# Preliminary User's Manual

# 78K0S/KB1+

## 8-Bit Single-Chip Microcontrollers

## μPD78F9232
## μPD78F9234

Document No. U17446EJ1V0UD00 (1st edition)
Date Published April 2005 NS 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.

### ⑤ POWER ON/OFF SEQUENCE

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

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

### ⑥ INPUT OF SIGNAL DURING POWER OFF STATE

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

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

Caution: This product uses SuperFlash® technology licensed from Silicon Storage Technology, inc.

- **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

# INTRODUCTION

**Target Readers**

This manual is intended for user engineers who wish to understand the functions of the 78K0S/KB1+ in order to design and develop its application systems and programs.
The target devices are the following subseries products.

- 78K0S/KB1+: μPD78F9232, 78F9234

**Purpose**

This manual is intended to give users on understanding of the functions described in the **Organization** below.

**Organization**

Two manuals are available for the 78K0S/KB1+: this manual and the Instruction Manual (common to the 78K/0S Series).

| 78K0S/KB1+<br>User's Manual | 78K/0S Series<br>Instructions<br>User's Manual |
|---|---|
| • Pin functions<br>• Internal block functions<br>• Interrupts<br>• Other internal peripheral functions<br>• Electrical specifications (target values) | • CPU function<br>• Instruction set<br>• Instruction description |

**How to Use This Manual**

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

◊ To understand the overall functions of 78K0S/KB1+
→ Read this manual in the order of the **CONTENTS**.

◊ How to read register formats
→ For a bit number enclosed in a square, the bit name is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S.

◊ To learn the detailed functions of a register whose register name is known
→ See **APPENDIX C REGISTER INDEX**.

◊ To learn the details of the instruction functions of the 78K/0S Series
→ Refer to **78K/0S Series Instructions User's Manual (U11047E)** separately available.

◊ To learn the electrical specifications (target) of the 78K0S/KB1+
→ See **CHAPTER 21 ELECTRICAL SPECIFICATIONS (TARGET VALUES)**.

**Conventions**

| | |
|---|---|
| Data significance: | Higher digits on the left and lower digits on the right |
| Active low representation: | $\overline{\times\times\times}$ (overscore over pin or signal name) |
| **Note**: | Footnote for item marked with **Note** in the text |
| **Caution**: | Information requiring particular attention |
| **Remark**: | Supplementary information |
| Numerical representation: | Binary ... ×××× or ××××B<br>Decimal ... ××××<br>Hexadecimal ... ××××H |

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

**Documents Related to Devices**

| Document Name | Document No. |
|---|---|
| 78K0S/KB1+ User's Manual | This manual |
| 78K/0S Series Instructions User's Manual | U11047E |

**Documents Related to Development Software Tools (User's Manuals)**

| Document Name | | Document No. |
|---|---|---|
| RA78K0S Assembler Package | Operation | U16656E |
| | Language | U14877E |
| | Structured Assembly Language | U11623E |
| CC78K0S C Compiler | Operation | U16654E |
| | Language | U14872E |
| ID78K0S-NS Ver. 2.52 Integrated Debugger | Operation | U16584E |
| ID78K0S-QB Ver. 2.81 Integrated Debugger | Operation | U17287E |
| PM plus Ver. 5.20 | | U16934E |

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

| Document Name | Document No. |
|---|---|
| IE-78K0S-NS In-Circuit Emulator | U13549E |
| IE-78K0S-NS-A In-Circuit Emulator | U15207E |
| QB-78K0SKX1MINI In-Circuit Emulator | U17272E |

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

### Documents Related to Flash Memory Writing

| Document Name | Document No. |
|---|---|
| PG-FP4 Flash Memory Programmer User’s Manual | U15260E |
| PG-FPL2 Flash Memory Programmer User’s Manual | U17307E |

### Other Related Documents

| Document Name | Document No. |
|---|---|
| SEMICONDUCTOR SELECTION GUIDE - Products and Packages - | X13769X |
| Semiconductor Device Mount Manual | **Note** |
| Quality Grades on NEC Semiconductor Devices | C11531E |
| NEC Semiconductor Device Reliability/Quality Control System | C10983E |
| Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD) | C11892E |

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

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

# CONTENTS

**CHAPTER 1 OVERVIEW** ..... 15

**1.1 Features** ..... 15

**1.2 Application Fields** ..... 15

**1.3 Ordering Information** ..... 16

**1.4 Pin Configuration (Top View)** ..... 17

**1.5 78K0S/Kx1+ Product Lineup** ..... 18

**1.6 Block Diagram** ..... 19

**1.7 Functional Outline** ..... 20

**CHAPTER 2 PIN FUNCTIONS** ..... 21

**2.1 Pin Function List** ..... 21

**2.2 Pin Functions** ..... 23

2.2.1 P00 to P03 (Port 0) ..... 23

2.2.2 P20 to P23 (Port 2) ..... 23

2.2.3 P30 to P34 (Port 3) ..... 23

2.2.4 P40 to P47 (Port 4) ..... 24

2.2.5 P121 to P123 (Port 12) ..... 24

2.2.6 P130 (Port 13) ..... 24

2.2.7 $\overline{\text{RESET}}$ ..... 24

2.2.8 X1 and X2 ..... 25

2.2.9 $AV_{REF}$ ..... 25

2.2.10 $AV_{SS}$ ..... 25

2.2.11 $V_{DD}$ ..... 25

2.2.12 $V_{SS}$ ..... 25

**2.3 Pin I/O Circuits and Connection of Unused Pins** ..... 26

**CHAPTER 3 CPU ARCHITECTURE** ..... 28

**3.1 Memory Space** ..... 28

3.1.1 Internal program memory space ..... 30

3.1.2 Internal data memory space ..... 30

3.1.3 Special function register (SFR) area ..... 31

3.1.4 Data memory addressing ..... 31

**3.2 Processor Registers** ..... 33

3.2.1 Control registers ..... 33

3.2.2 General-purpose registers ..... 35

3.2.3 Special function registers (SFRs) ..... 36

**3.3 Instruction Address Addressing** ..... 39

3.3.1 Relative addressing ..... 39

3.3.2 Immediate addressing ..... 40

3.3.3 Table indirect addressing ..... 40

3.3.4 Register addressing ..... 41

**3.4 Operand Address Addressing** ..... 42

3.4.1 Direct addressing ..... 42

3.4.2 Short direct addressing ..... 43

3.4.3 Special function register (SFR) addressing ..... 44

3.4.4 Register addressing ........ 45
3.4.5 Register indirect addressing ........ 46
3.4.6 Based addressing ........ 47
3.4.7 Stack addressing ........ 48

**CHAPTER 4 PORT FUNCTIONS ........ 49**
**4.1 Functions of Ports ........ 49**
**4.2 Port Configuration ........ 51**
4.2.1 Port 0 ........ 52
4.2.2 Port 2 ........ 53
4.2.3 Port 3 ........ 54
4.2.4 Port 4 ........ 57
4.2.5 Port 12 ........ 60
4.2.6 Port 13 ........ 62
**4.3 Registers Controlling Port Functions ........ 63**
**4.4 Operation of Port Function ........ 68**
4.4.1 Writing to I/O port ........ 68
4.4.2 Reading from I/O port ........ 68
4.4.3 Operations on I/O port ........ 68

**CHAPTER 5 CLOCK GENERATORS ........ 69**
**5.1 Functions of Clock Generators ........ 69**
5.1.1 System clock oscillators ........ 69
5.1.2 Clock oscillator for interval time generation ........ 69
**5.2 Configuration of Clock Generators ........ 70**
**5.3 Registers Controlling Clock Generators ........ 72**
**5.4 System Clock Oscillators ........ 75**
5.4.1 High-speed Ring-OSC oscillator ........ 75
5.4.2 Crystal/ceramic oscillator ........ 75
5.4.3 External clock input circuit ........ 77
5.4.4 Prescaler ........ 77
**5.5 Operation of CPU Clock Generator ........ 78**
**5.6 Operation of Clock Generator Supplying Clock to Peripheral Hardware ........ 83**

**CHAPTER 6 16-BIT TIMER/EVENT COUNTER 00 ........ 85**
**6.1 Functions of 16-Bit Timer/Event Counter 00 ........ 85**
**6.2 Configuration of 16-Bit Timer/Event Counter 00 ........ 86**
**6.3 Registers to Control 16-Bit Timer/Event Counter 00 ........ 90**
**6.4 Operation of 16-Bit Timer/Event Counter 00 ........ 96**
6.4.1 Interval timer operation ........ 96
6.4.2 External event counter operation ........ 99
6.4.3 Pulse width measurement operations ........ 102
6.4.4 Square-wave output operation ........ 110
6.4.5 PPG output operations ........ 112
6.4.6 One-shot pulse output operation ........ 115
**6.5 Cautions Related to 16-Bit Timer/Event Counter 00 ........ 120**

CHAPTER 7 8-BIT TIMER 80 ........ 126
7.1 Function of 8-Bit Timer 80 ........ 126
7.2 Configuration of 8-Bit Timer 80 ........ 127
7.3 Register Controlling 8-Bit Timer 80 ........ 129
7.4 Operation of 8-Bit Timer 80 ........ 130
7.4.1 Operation as interval timer ........ 130
7.5 Notes on 8-Bit Timer 80 ........ 132

CHAPTER 8 8-BIT TIMER H1 ........ 133
8.1 Functions of 8-Bit Timer H1 ........ 133
8.2 Configuration of 8-Bit Timer H1 ........ 133
8.3 Registers Controlling 8-Bit Timer H1 ........ 136
8.4 Operation of 8-Bit Timer H1 ........ 139
8.4.1 Operation as interval timer/square-wave output ........ 139
8.4.2 Operation as PWM output mode ........ 142

CHAPTER 9 WATCHDOG TIMER ........ 148
9.1 Functions of Watchdog Timer ........ 148
9.2 Configuration of Watchdog Timer ........ 150
9.3 Registers Controlling Watchdog Timer ........ 151
9.4 Operation of Watchdog Timer ........ 153
9.4.1 Watchdog timer operation when “low-speed Ring-OSC cannot be stopped” is selected by option byte ........ 153
9.4.2 Watchdog timer operation when “low-speed Ring-OSC can be stopped by software” is selected by option byte ........ 155
9.4.3 Watchdog timer operation in STOP mode (when “low-speed Ring-OSC can be stopped by software” is selected by option byte) ........ 157
9.4.4 Watchdog timer operation in HALT mode (when “low-speed Ring-OSC can be stopped by software” is selected by option byte) ........ 159

CHAPTER 10 A/D CONVERTER ........ 160
10.1 Functions of A/D Converter ........ 160
10.2 Configuration of A/D Converter ........ 163
10.3 Registers Used by A/D Converter ........ 164
10.4 A/D Converter Operations ........ 169
10.4.1 Basic operations of A/D converter ........ 169
10.4.2 Input voltage and conversion results ........ 171
10.4.3 A/D converter operation mode ........ 172
10.5 How to Read A/D Converter Characteristics Table ........ 174
10.6 Cautions for A/D Converter ........ 176

CHAPTER 11 SERIAL INTERFACE UART6 ........ 179
11.1 Functions of Serial Interface UART6 ........ 179
11.2 Configuration of Serial Interface UART6 ........ 183
11.3 Registers Controlling Serial Interface UART6 ........ 185
11.4 Operation of Serial Interface UART6 ........ 195
11.4.1 Operation stop mode ........ 195

11.4.2 Asynchronous serial interface (UART) mode ... 196
11.4.3 Dedicated baud rate generator ... 210

**CHAPTER 12 MULTIPLIER ... 217**
**12.1 Multiplier Function ... 217**
**12.2 Multiplier Configuration ... 217**
**12.3 Multiplier Control Register ... 219**
**12.4 Multiplier Operation ... 220**

**CHAPTER 13 INTERRUPT FUNCTIONS ... 221**
**13.1 Interrupt Function Types ... 221**
**13.2 Interrupt Sources and Configuration ... 222**
**13.3 Interrupt Function Control Registers ... 224**
**13.4 Interrupt Servicing Operation ... 229**
13.4.1 Maskable interrupt request acknowledgment operation ... 229
13.4.2 Multiple interrupt servicing ... 232
13.4.3 Interrupt request pending ... 233

**CHAPTER 14 STANDBY FUNCTION ... 234**
**14.1 Standby Function and Configuration ... 234**
14.1.1 Standby function ... 234
14.1.2 Registers used during standby ... 236
**14.2 Standby Function Operation ... 237**
14.2.1 HALT mode ... 237
14.2.2 STOP mode ... 240

**CHAPTER 15 RESET FUNCTION ... 244**
**15.1 Register for Confirming Reset Source ... 251**

**CHAPTER 16 POWER-ON-CLEAR CIRCUIT ... 252**
**16.1 Functions of Power-on-Clear Circuit ... 252**
**16.2 Configuration of Power-on-Clear Circuit ... 253**
**16.3 Operation of Power-on-Clear Circuit ... 253**
**16.4 Cautions for Power-on-Clear Circuit ... 254**

**CHAPTER 17 LOW-VOLTAGE DETECTOR ... 256**
**17.1 Functions of Low-Voltage Detector ... 256**
**17.2 Configuration of Low-Voltage Detector ... 256**
**17.3 Registers Controlling Low-Voltage Detector ... 257**
**17.4 Operation of Low-Voltage Detector ... 259**
**17.5 Cautions for Low-Voltage Detector ... 262**

**CHAPTER 18 OPTION BYTE ... 265**

**CHAPTER 19 FLASH MEMORY ... 268**
**19.1 Features ... 268**
**19.2 Memory Configuration ... 269**

**19.3 Functional Outline ... 270**
**19.4 Writing with Flash Programmer ... 271**
**19.5 Programming Environment ... 272**
**19.6 Pin Connection on Board ... 274**
19.6.1 X1 and X2 pins ... 274
19.6.2 $\overline{\text{RESET}}$ pin ... 275
19.6.3 Port pins ... 276
19.6.4 Power supply ... 276
**19.7 On-Board and Off-Board Flash Memory Programming ... 277**
19.7.1 Controlling flash memory ... 277
19.7.2 Flash memory programming mode ... 278
19.7.3 Communication commands ... 278
19.7.4 Security settings ... 279
**19.8 Flash Memory Programming by Self Writing ... 280**
19.8.1 Outline of self programming ... 280
19.8.2 Cautions on self programming function ... 283
19.8.3 Registers used for self programming function ... 283
19.8.4 Example of shifting normal mode to self programming mode ... 291
19.8.5 Example of shifting self programming mode to normal mode ... 294
19.8.6 Example of block erase operation in self programming mode ... 297
19.8.7 Example of block blank check operation in self programming mode ... 300
19.8.8 Example of byte write operation in self programming mode ... 303
19.8.9 Example of internal verify operation in self programming mode ... 306
19.8.10 Examples of operation when command execution time should be minimized in self programming mode ... 309
19.8.11 Examples of operation when interrupt-disabled time should be minimized in self programming mode ... 316

**CHAPTER 20 INSTRUCTION SET OVERVIEW ... 327**
**20.1 Operation ... 327**
20.1.1 Operand identifiers and description methods ... 327
20.1.2 Description of “Operation” column ... 328
20.1.3 Description of “Flag” column ... 328
**20.2 Operation List ... 329**
**20.3 Instructions Listed by Addressing Type ... 334**

**CHAPTER 21 ELECTRICAL SPECIFICATIONS (TARGET VALUES) ... 337**

**CHAPTER 22 PACKAGE DRAWING ... 349**

**APPENDIX A DEVELOPMENT TOOLS ... 350**
**A.1 Software Package ... 353**
**A.2 Language Processing Software ... 353**
**A.3 Control Software ... 354**
**A.4 Flash Memory Writing Tools ... 354**
**A.5 Debugging Tools (Hardware) ... 355**
A.5.1 When using in-circuit emulator IE-78K0S-NS or IE-78K0S-NS-A ... 355
A.5.2 When using in-circuit emulator QB-78K0SKX1MINI ... 355

A.6 Debugging Tools (Software) ....................356

APPENDIX B NOTES ON TARGET SYSTEM DESIGN ....................357

APPENDIX C REGISTER INDEX ....................360
C.1 Register Index (Register Name) ....................360
C.2 Register Index (Symbol) ....................363

APPENDIX D LIST OF CAUTIONS ....................366

# CHAPTER 1 OVERVIEW

## 1.1 Features

O Minimum instruction execution time selectable from high speed (0.2 μs) to low speed (3.2 μs) (with CPU clock of 10 MHz)

O General-purpose registers: 8 bits × 8 registers

O ROM and RAM capacities

| Item<br>Part number | Program Memory (Flash Memory) | Memory (Internal High-Speed RAM) |
|---|---|---|
| μPD78F9232 | 4 KB | 256 bytes |
| μPD78F9234 | 8 KB | 256 bytes |

O On-chip power-on clear (POC) circuit and low voltage detector (LVI)

O On-chip watchdog timer (operable on internal low-speed Ring-OSC clock)

O I/O ports: 26

O Timer: 4 channels

| | |
|---|---|
| • 16-bit timer/event counter: | 1 channel |
| • 8-bit timer: | 2 channels |
| • Watchdog timer: | 1 channel |

O Serial interface: UART (LIN (Local Interconnect Network) bus supported) 1 channel

O On-chip multiplier: 8 bits x 8 bits = 16 bits

O 10-bit resolution A/D converter: 4 channels

O Supply voltage: $V_{DD}$ = 2.0 to 5.5 V[Note]

O Operating temperature range: $T_A$ = –40 to +85°C

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

## 1.2 Application Fields

O Automotive electronics

- System control of body instrumentation system (such as power windows and keyless entry reception)
- Sub-microcontroller of control system

O Household appliances

- Electric toothbrushes
- Electric shavers

O Toys

O Industrial equipment

- Sensor and switch control
- Power tools

## 1.3 Ordering Information

| Part Number | Package | Quality Grade |
|---|---|---|
| μ PD78F9232MC(T)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9232MC(S)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9232MC(R)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9232MC(T)-5A4-A | 30-pin plastic SSOP | Standard |
| μPD78F9232MC(S)-5A4-A | 30-pin plastic SSOP | Standard |
| μPD78F9232MC(R)-5A4-A | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(T)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(S)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(R)-5A4 | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(T)-5A4-A | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(S)-5A4-A | 30-pin plastic SSOP | Standard |
| μPD78F9234MC(R)-5A4-A | 30-pin plastic SSOP | Standard |

**Remark** The μPD78F9232MC(xx)- 5A4-A and 78F9234MC(xx)-5A4-A are lead-free products.
xx : T, S, R

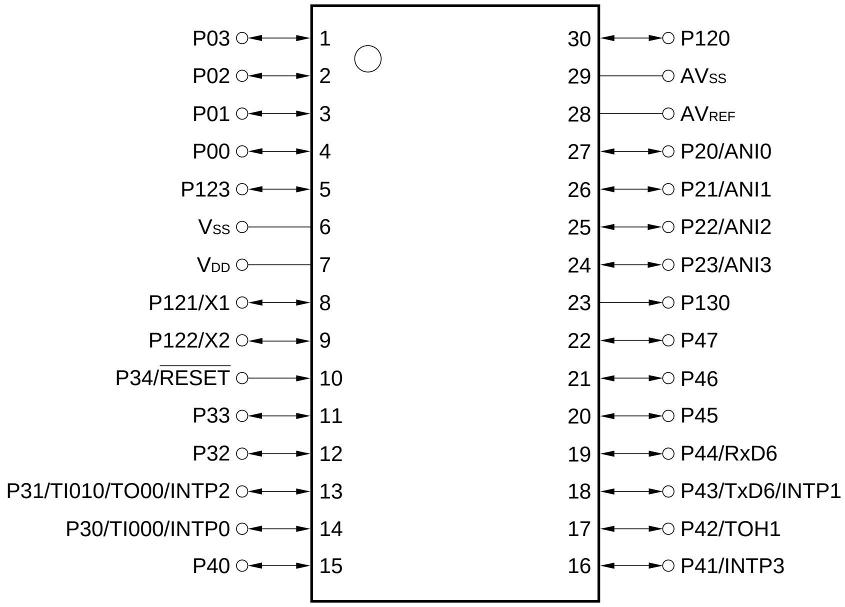
Please refer to "Quality Grades on NEC Semiconductor Devices" (Document No. C11531E) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

The 78K0S/KB1+ standard grade products are further classified as follows.

| | |
|---|---|
| (T): | General management |
| (S): | Management based on individual contract |
| (R): | Management for automotive accessories |

## 1.4 Pin Configuration (Top View)

30-pin plastic SSOP



**Caution Connect the AV$_{SS}$ pin to V$_{SS}$.**

| | | | |
|---|---|---|---|
| ANI0 to ANI3: | Analog input | P130: | Port 13 |
| AV$_{REF}$: | Analog reference voltage | $\overline{\text{RESET}}$: | Reset |
| AV$_{SS}$: | Analog ground | RxD6: | Receive data |
| INTP0 to INTP3: | External interrupt input | TI000, TI010: | Timer input |
| P00 to P03: | Port 0 | TO00, TOH1: | Timer output |
| P20 to P23: | Port 2 | TxD6: | Transmit data |
| P30 to P34: | Port 3 | V$_{DD}$: | Power supply |
| P40 to P47: | Port 4 | V$_{SS}$: | Ground |
| P120 to P123: | Port 12 | X1, X2: | Crystal oscillator (X1 input clock) |

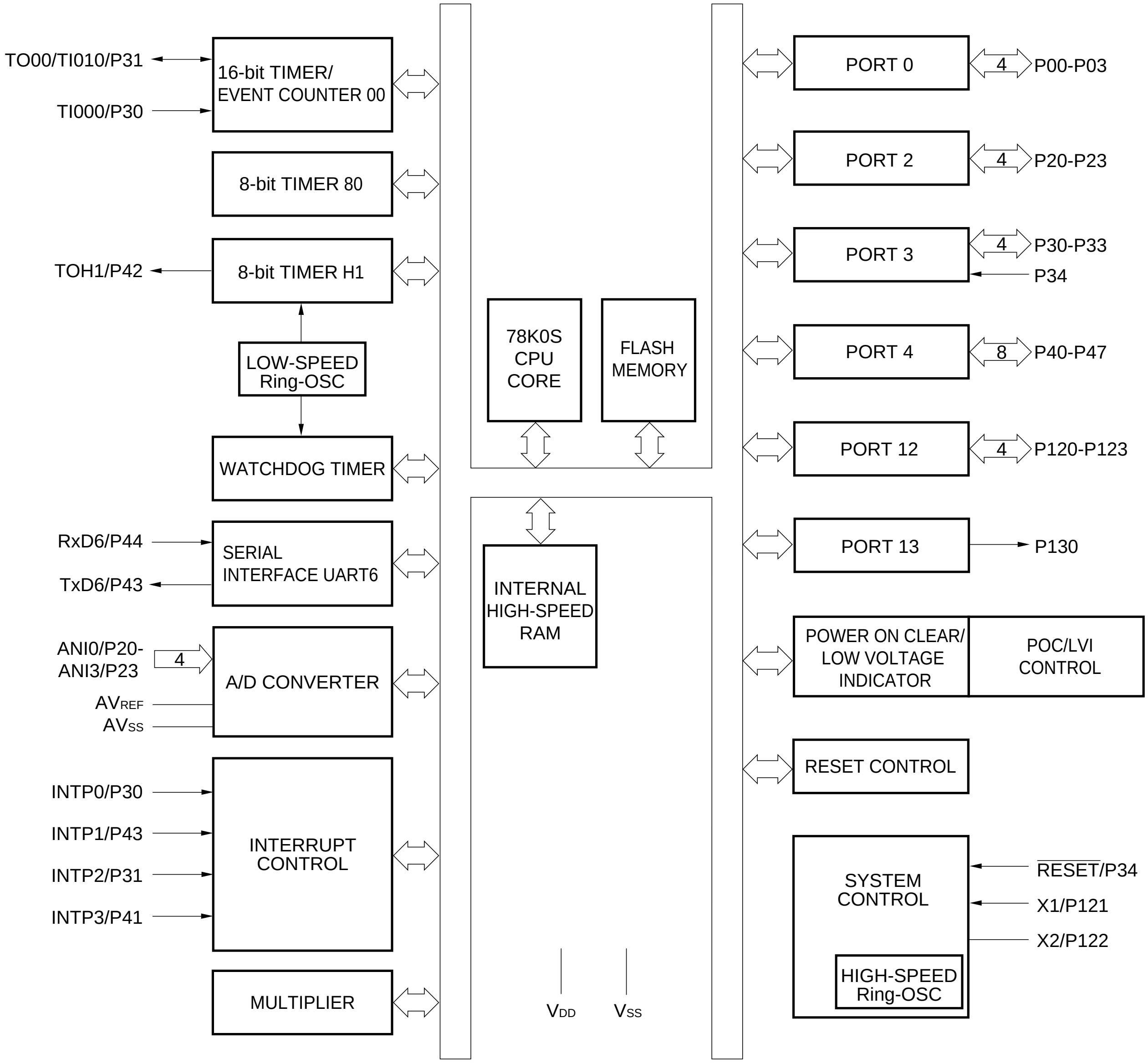
## 1.5 78K0S/Kx1+ Product Lineup

The following table shows the product lineup of the 78K0S/Kx1+.

| Part Number<br>Item | | 78K0S/KY1+ | 78K0S/KA1+ | | 78K0S/KB1+ |
|---|---|---|---|---|---|
| Number of pins | | 16 pins | 20 pins | | 30 pins |
| Internal memory | Flash memory | 1 KB, 2 KB, 4 KB | 2 KB | 4 KB | 4 KB, 8 KB |
| | RAM | 128 bytes | 128 bytes | 256 bytes | 256 bytes |
| Supply voltage | | $V_{DD}$ = 2.0 to 5.5 V[Note] | | | |
| Minimum instruction execution time | | 0.20 µs (10 MHz, $V_{DD}$ = 4.0 to 5.5 V)<br>0.33 µs (6 MHz, $V_{DD}$ = 3.0 to 5.5 V)<br>0.40 µs (5 MHz, $V_{DD}$ = 2.7 to 5.5 V)<br>1.0 µs (2 MHz, $V_{DD}$ = 2.0 to 5.5 V) | | | |
| System clock (oscillation frequency) | | Internal high-speed Ring-OSC oscillation (8 MHz (TYP.))<br>Crystal/ceramic oscillation (1 to 10 MHz)<br>External clock input oscillation (1 to 10 MHz) | | | |
| Clock for TMH1 and WDT (oscillation frequency) | | Internal low-speed Ring-OSC oscillation (240 kHz (TYP.)) | | | |
| Port | CMOS I/O | 13 | 15 | | 24 |
| | CMOS input | 1 | 1 | | 1 |
| | CMOS output | – | 1 | | 1 |
| Timer | 16-bit (TM0) | 1 ch | | | |
| | 8-bit (TMH) | 1 ch | | | |
| | 8-bit (TM8) | – | 1 ch | | |
| | WDT | 1 ch | | | |
| Serial interface | | – | LIN-Bus-supporting UART: 1 ch | | |
| A/D converter | | 10 bits: 4 ch (2.7 to 5.5V) | | | |
| Multiplier (8 bits x 8 bits) | | – | | | Provided |
| Interrupts | External | 2 | 4 | | |
| | Internal | 5 | 9 | | |
| Reset | $\overline{\text{RESET}}$ pin | Provided | | | |
| | POC | 2.1 V ±0.1 V | | | |
| | LVI | Provided (selectable by software) | | | |
| | WDT | Provided | | | |
| Operating temperature range | | –40 to +85°C | | | |

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

## 1.6 Block Diagram



## 1.7 Functional Outline

| Item | | µPD78F9232 | µPD78F9234 |
|---|---|---|---|
| Internal memory | Flash memory | 4 KB | 8 KB |
| | High-speed RAM | 256 bytes | |
| Memory space | | 64 KB | |
| X1 input clock (oscillation frequency) | | Crystal/ceramic/external clock input:<br>10 MHz ($V_{DD}$ = 2.0 to 5.5 V) | |
| Ring-OSC clock | High speed (oscillation frequency) | Internal Ring oscillation: 8 MHz (TYP.) | |
| | Low speed (for TMH1 and WDT) | Internal Ring oscillation: 240 kHz (TYP.) | |
| General-purpose registers | | 8 bits × 8 registers | |
| Minimum instruction execution time | | 0.2 µs/0.4 µs/0.8 µs/1.6 µs/3.2 µs (X1 input clock: $f_X$ = 10 MHz) | |
| Multiplier | | 8 bits x 8 bits = 16 bits | |
| I/O port | | Total: 26 pins<br>CMOS I/O: 24 pins<br>CMOS input: 1 pin<br>CMOS output: 1 pin | |
| Timer | | • 16-bit timer/event counter: 1 channel<br>• 8-bit timer (timer H1): 1 channel<br>• 8-bit timer (timer 80): 1 channel<br>• Watchdog timer: 1 channel | |
| | Timer output | 2 pins (PWM: 1 pin) | |
| A/D converter | | 10-bit resolution × 4 channels | |
| Serial interface | | LIN-bus-supporting UART mode: 1 channel | |
| Vectored interrupt sources | External | 4 | |
| | Internal | 9 | |
| Reset | | • Reset by $\overline{\text{RESET}}$ pin<br>• Internal reset by watchdog timer<br>• Internal reset by power-on clear<br>• Internal reset by low-voltage detector | |
| Supply voltage | | $V_{DD}$ = 2.0 to 5.5 V[Note] | |
| Operating temperature range | | $T_A$ = –40 to +85°C | |
| Package | | 30-pin plastic SSOP | |

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

# CHAPTER 2 PIN FUNCTIONS

## 2.1 Pin Function List

### (1) Port pins

| Pin Name | I/O | Function | | After Reset | Alternate-Function Pin |
|---|---|---|---|---|---|
| P00 to P03 | I/O | Port 0.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>An on-chip pull-up resistor can be connected by setting software. | | Input | – |
| P20 to P23 | I/O | Port 2.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>An on-chip pull-up resistor can be connected by setting software. | | Input | ANI0 to ANI3 |
| P30 | I/O | Port 3 | Can be set to input or output mode in 1-bit units.<br>An on-chip pull-up resistor can be connected by setting software. | Input | TI000/INTP0 |
| P31 | | | | | TI010/TO00/INTP2 |
| P32 | | | | | – |
| P33 | | | | | – |
| P34[Note] | Input | | Input only | Input | $\overline{\text{RESET}}$[Note] |
| P40 | I/O | Port 4.<br>8-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>An on-chip pull-up resistor can be connected by setting software. | | Input | – |
| P41 | | | | | INTP3 |
| P42 | | | | | TOH1 |
| P43 | | | | | TxD6/INTP1 |
| P44 | | | | | RxD6 |
| P45 | | | | | – |
| P46 | | | | | – |
| P47 | | | | | – |
| P120 | I/O | Port 12.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>An on-chip pull-up resistor can be connected only to P120 and P123 by setting software. | | Input | – |
| P121[Note] | | | | | X1[Note] |
| P122[Note] | | | | | X2[Note] |
| P123 | | | | | – |
| P130 | Output | Port 13.<br>1-bit output-only port | | Output | – |

**Note** For settings of alternate function, refer to **CHAPTER 18 OPTION BYTE**.

**Caution The P121/X1 and P122/X2 pins are pulled down during reset.**

### (2) Non-port pins

| Pin Name | I/O | Function | After Reset | Alternate-Function Pin |
|---|---|---|---|---|
| INTP0 | Input | External interrupt input for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified | Input | P30/TI000 |
| INTP1 | | | | P43/TxD6 |
| INTP2 | | | | P31/TI010/TO00 |
| INTP3 | | | | P41 |
| RxD6 | Input | Serial data input for asynchronous serial interface | Input | P44 |
| TxD6 | Output | Serial data output for asynchronous serial interface | Input | P43/INTP1 |
| TI000 | | External count clock input to 16-bit timer/event counter 00.<br>Capture trigger input to capture registers (CR000 and CR010) of 16-bit timer/event counter 00 | | P30/INTP0 |
| TI010 | | | | |

## 2.2 Pin Functions

### 2.2.1 P00 to P03 (Port 0)

P00 to P03 function as a 4-bit I/O port. P00 to P03 can be set to input or output in 1-bit units using port mode register 0 (PM0). Use of an on-chip pull-up resistor can be specified by pull-up resistor option register 0 (PU0).

### 2.2.2 P20 to P23 (Port 2)

P20 to P23 constitute a 4-bit I/O port, port 2. In addition to I/O port pins, these pins also have a function to input analog signals to the A/D converter. These pins can be set to the following operation modes in 1-bit units.

**(1) Port mode**

P20 to P23 function as a 4-bit I/O port. Each bit of this port can be set to the input or output mode by using port mode register 2 (PM2). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 2 (PU2).

**(2) Control mode**

P20 to P23 function as the analog input pins (ANI0 to ANI3) of the A/D converter. When using these pins as analog input pins, refer to **10.6 Cautions for A/D converter (5) ANI0/P20 to ANI3/P23**.

### 2.2.3 P30 to P34 (Port 3)

P30 to P33 constitute a 4-bit I/O port, port 3. In addition to I/O port pins, P30 and p31 also have functions to input/output a timer signal, and input an external interrupt request signal.

P34 is a 1-bit input-only port. This pin is also used as a $\overline{\text{RESET}}$ pin. For settings of alternate function, refer to **CHAPTER 18 OPTION BYTE**. When using P34 as input port, pull up the P34 pin by using external resistor.

P30 to P33 can be set to the following operation modes in 1-bit units.

**(1) Port mode**

P30 to P33 function as a 4-bit I/O port. Each bit of this port can be set to the input or output mode by using port mode register 3 (PM3). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 3 (PU3).

P34 functions as a 1-bit input-only port.

**(2) Control mode**

P30 and P31 function to input/output signals to/from internal timers, and to input an external interrupt request signal.

**(a) INTP0 and INTP2**

These are external interrupt request input pins for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified.

**(b) TI000**

This pin inputs an external count clock to 16-bit timer/event counter 00, or a capture trigger signal to the capture registers (CR000 and CR010) of 16-bit timer/event counter 00.

**(c) TI010**

This pin inputs a capture trigger signal to the capture register (CR000) of 16-bit timer/event counter 00.

#### (d) TO00

This pin outputs a signal from 16-bit timer/event counter 00.

### 2.2.4 P40 to P47 (Port 4)

P40 to P47 constitute a 8-bit I/O port, port 4. In addition to I/O port pins, P41 to P44 also have functions to output a timer signal, input external interrupt request signals, and input/output the data of the serial interface.

These pins can be set to the following operation modes in 1-bit units.

#### (1) Port mode

P40 to P47 function as a 8-bit I/O port. Each bit of this port can be set to the input or output mode by using port mode register 4 (PM4). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 4 (PU4).

#### (2) Control mode

P41 to 44 function to output a signal from an internal timer, input external interrupt request signals, and input/output data of the serial interface.

#### (a) INTP1 and INTP3

These are external interrupt request input pins for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified.

#### (b) TOH1

This is the output pin of 8-bit timer H1.

#### (c) TxD6

This pin outputs serial data from the asynchronous serial interface.

#### (d) RxD6

This pin inputs serial data to the asynchronous serial interface.

### 2.2.5 P120 to P123 (Port 12)

P120 to P123 constitute a 4-bit I/O port, port 12.

Each bit of this port can be set to the input or output mode by using port mode register 12 (PM12). An on-chip pull-up resistor can be connected to P120 and P123 by using pull-up resistor option register 12 (PU12).

P121 and P122 also function as the X1 and X2 pins, respectively. For settings of alternate function, refer to **CHAPTER 18 OPTION BYTE**.

**Caution The P121/X1 and P122/X2 pins are pulled down during reset.**

### 2.2.6 P130 (Port 13)

This is a 1-bit output-only port.

### 2.2.7 $\overline{\text{RESET}}$

This pin inputs an active-low system reset signal.

### 2.2.8 X1 and X2

These pins connect an oscillator to oscillate the X1 input clock.

X1 and X2 also function as the P121 and P122 pins, respectively. For settings of alternate function, refer to **CHAPTER 18 OPTION BYTE**.

Supply an external clock to X1.

**Caution The P121/X1 and P122/X2 pins are pulled down during reset.**

### 2.2.9 $AV_{REF}$

This pin inputs a reference voltage to the internal A/D converter. When the A/D converter is not used, connect this pin to $V_{DD}$.

### 2.2.10 $AV_{SS}$

This is the A/D converter ground potential pin. Even when the A/D converter is not used, always use this pin with the same potential as the $V_{SS}$ pin.

### 2.2.11 $V_{DD}$

This is the positive power supply pin.

### 2.2.12 $V_{SS}$

This is the ground pin.

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

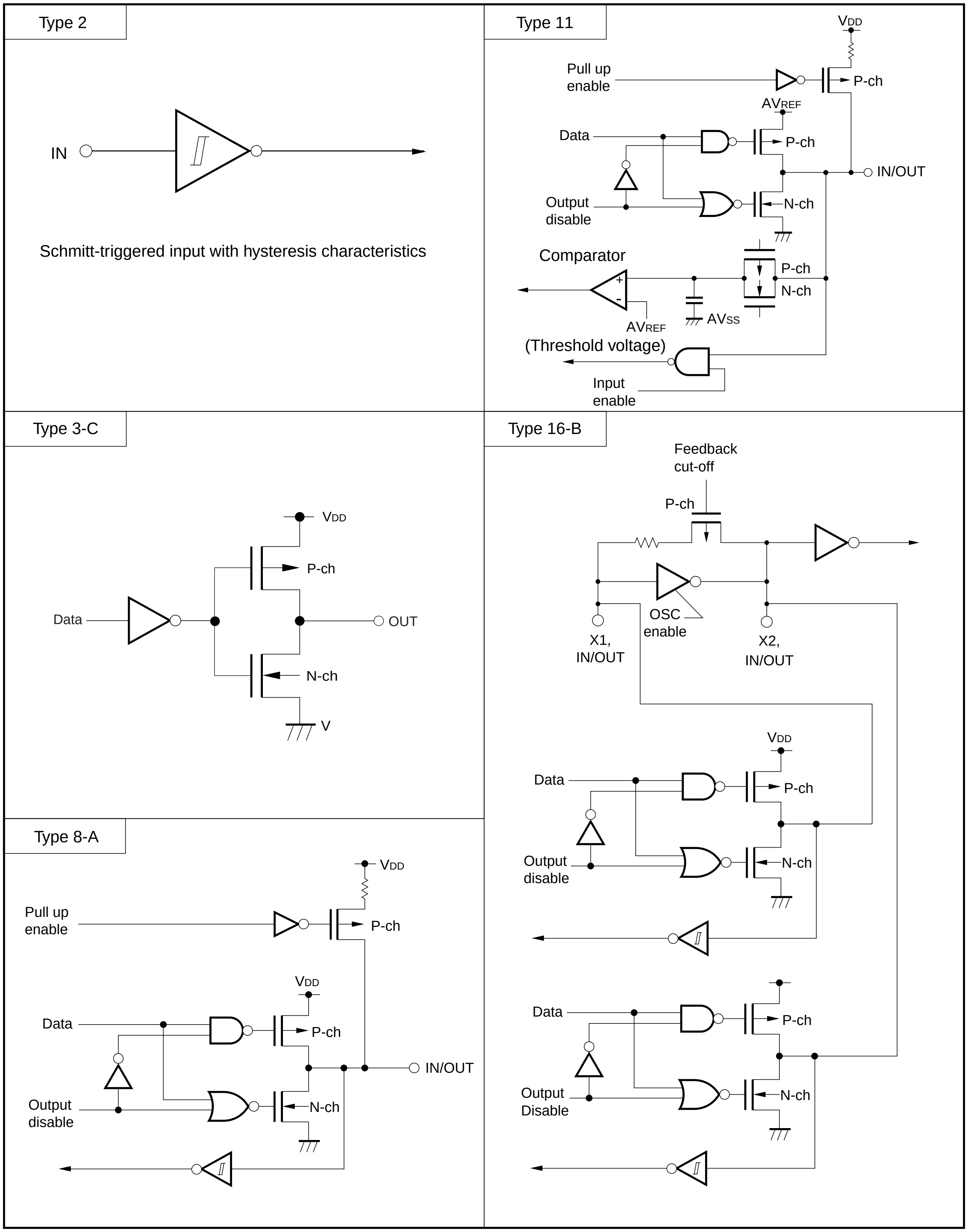
Table 2-1 shows I/O circuit type of each pin and the connections of unused pins.

For the configuration of the I/O circuit of each type, refer to **Figure 2-1**.

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

| Pin Name | I/O Circuit Type | I/O | Recommended Connection of Unused Pin |
|---|---|---|---|
| P00 to P03 | 8-A | I/O | Input: Independently connect to $V_{DD}$ or $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P20/ANI0 to P23/ANI3 | 11 | | Input: Independently connect to $AV_{REF}$ or $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P30/TI000/INTP0 | 8-A | | Input: Independently connect to $V_{DD}$ or $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P31/TI010/TO00/INTP2 | | | |
| P32 and P33 | | | |
| P34/$\overline{\text{RESET}}$ | 2 | Input | Connect to $V_{DD}$ via a resistor. |
| P40 | 8-A | I/O | Input: Independently connect to $V_{DD}$ or $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P41/INTP3 | | | |
| P42/TOH1 | | | |
| P43/TxD6/INTP1 | | | |
| P44/RxD6 | | | |
| P45 to P47 | | | |
| P120 | | | |
| P121/X1 | 16-B | | Input: Independently connect to $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P122/X2 | | | |
| P123 | 8-A | | Input: Independently connect to $V_{DD}$ or $V_{SS}$ via a resistor.<br>Output: Leave open. |
| P130 | 3-C | Output | Leave open. |
| $AV_{REF}$ | – | – | Directly connect to $V_{DD}$. |
| $AV_{SS}$ | – | – | Connect directly to $V_{SS}$. |

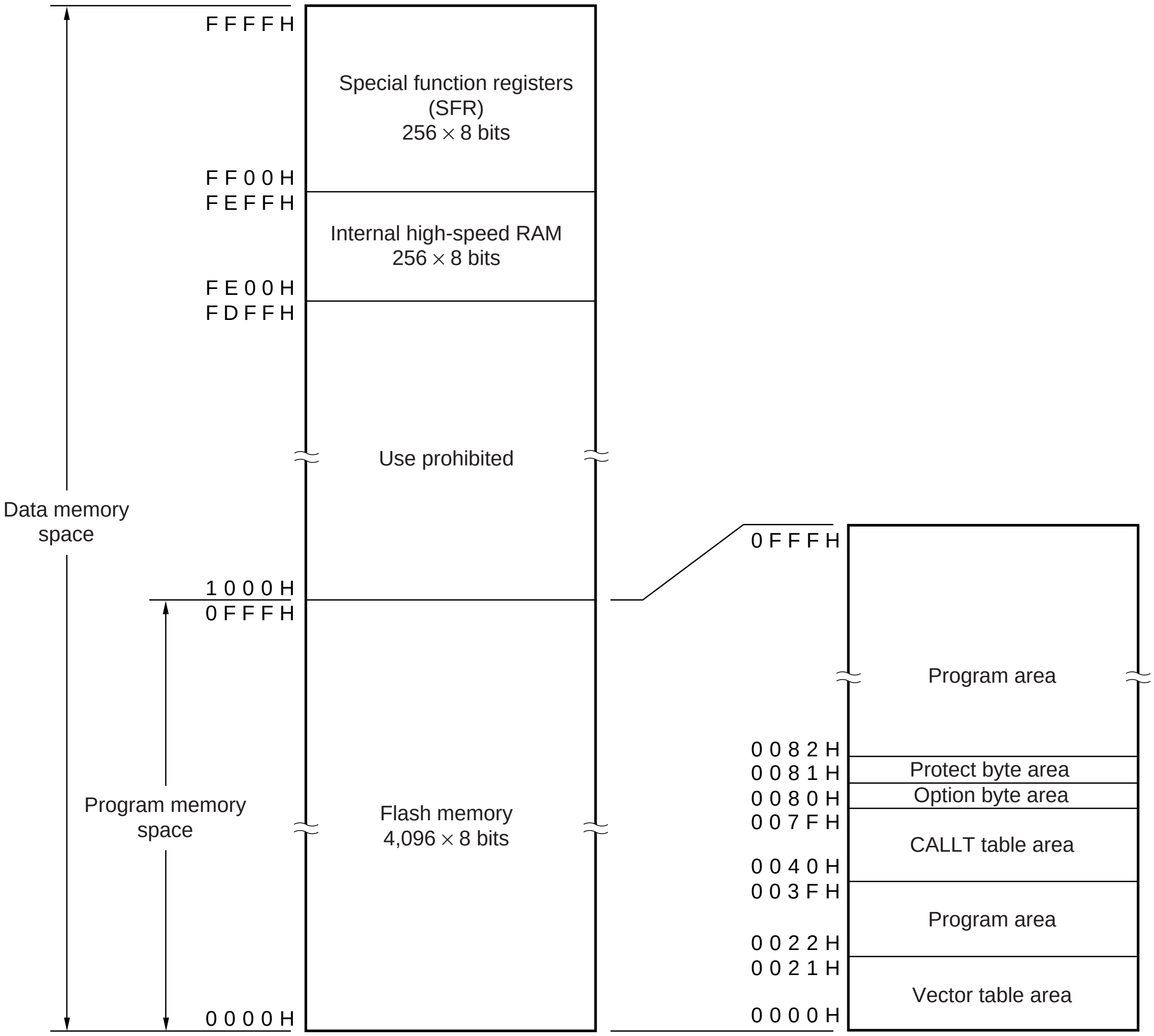
**Figure 2-1. Pin I/O Circuits**



## 3.1 Memory Space

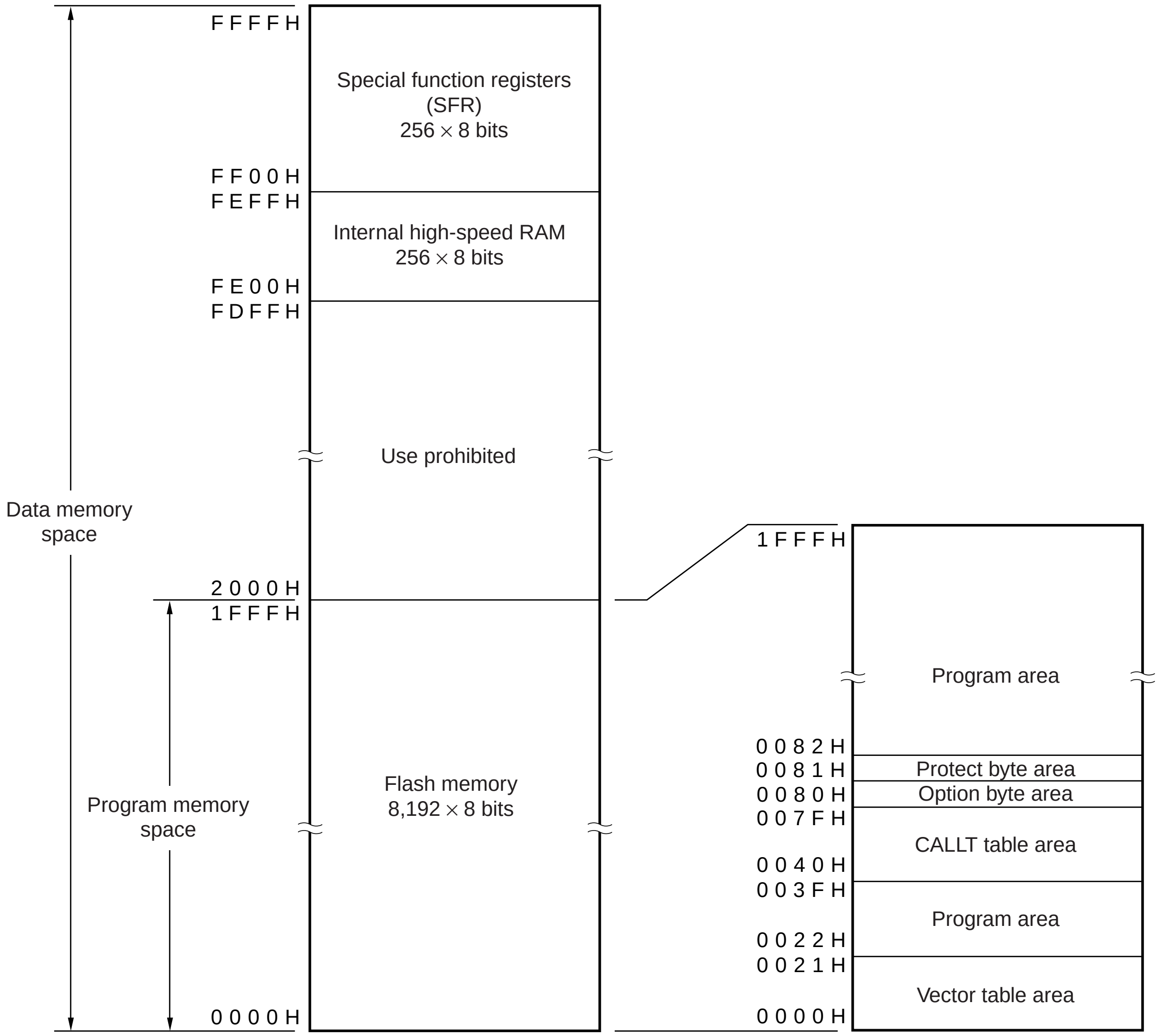
The 78K0S/KB1+ can access up to 64 KB of memory space. Figures 3-1 and 3-2 show the memory maps.

**Figure 3-1. Memory Map (μPD78F9232)**



**Remark** The option byte and protect byte are 1 byte each.

**Figure 3-2. Memory Map (μPD78F9234)**



**Remark** The option byte and protect byte are 1 byte each.

### 3.1.1 Internal program memory space

The internal program memory space stores programs and table data. This space is usually addressed by the program counter (PC).

The 78K0S/KB1+ provides the following internal ROMs (or flash memory) containing the following capacities.

**Table 3-1. Internal ROM Capacity**

| Part Number | Internal ROM | |
|---|---|---|
| | Structure | Capacity |
| μPD78F9232 | Flash memory | 4,096 × 8 bits |
| μPD78F9234 | | 8,192 × 8 bits |

The following areas are allocated to the internal program memory space.

**(1) Vector table area**

The 34-byte area of addresses 0000H to 0021H is reserved as a vector table area. This area stores program start addresses to be used when branching by $\overline{\text{RESET}}$ input or interrupt request generation. Of a 16-bit address, the lower 8 bits are stored in an even address, and the higher 8 bits are stored in an odd address.

**Table 3-2. Vector Table**

| Vector Table Address | Interrupt Request | Vector Table Address | Interrupt Request |
|---|---|---|---|
| 0000H | Reset input | 0012H | INTAD |
| 0006H | INTLVI | 0016H | INTP2 |
| 0008H | INTP0 | 0018H | INTP3 |
| 000AH | INP1 | 001AH | INTTM80 |
| 000CH | INTTMH1 | 001CH | INTSRE6 |
| 000EH | INTTM000 | 001EH | INTSR6 |
| 0010H | INTTM010 | 0020H | INTST6 |

**Caution No interrupt sources correspond to the vector table address 0014H.**

**(2) CALLT instruction table area**

The subroutine entry address of a 1-byte call instruction (CALLT) can be stored in the 64-byte area of addresses 0040H to 007FH.

**(3) Option byte area**

The option byte area is the 1-byte area of address 0080H. For details, refer to **CHAPTER 18 OPTION BYTE**.

**(4) Protect byte area**

The protect byte area is the 1-byte area of address 0081H. For details, refer to **CHAPTER 19 FLASH MEMORY**.

### 3.1.2 Internal data memory space

μPD78F9232 and μPD78F9234 provide 256-byte internal high-speed RAM.

The internal high-speed RAM can also be used as a stack memory.

### 3.1.3 Special function register (SFR) area

Special function registers (SFRs) of on-chip peripheral hardware are allocated to the area of FF00H to FFFFH (see **Table 3-3**).

### 3.1.4 Data memory addressing

The 78K0S/KB1+ is provided with a wide range of addressing modes to make memory manipulation as efficient as possible. The area (FE00H to FEFFH) which contains a data memory and the special function register area (SFR) can be accessed using a unique addressing mode in accordance with each function. Figures 3-3 and 3-4 illustrate the data memory addressing.

**Figure 3-3. Data Memory Addressing (μPD78F9232)**

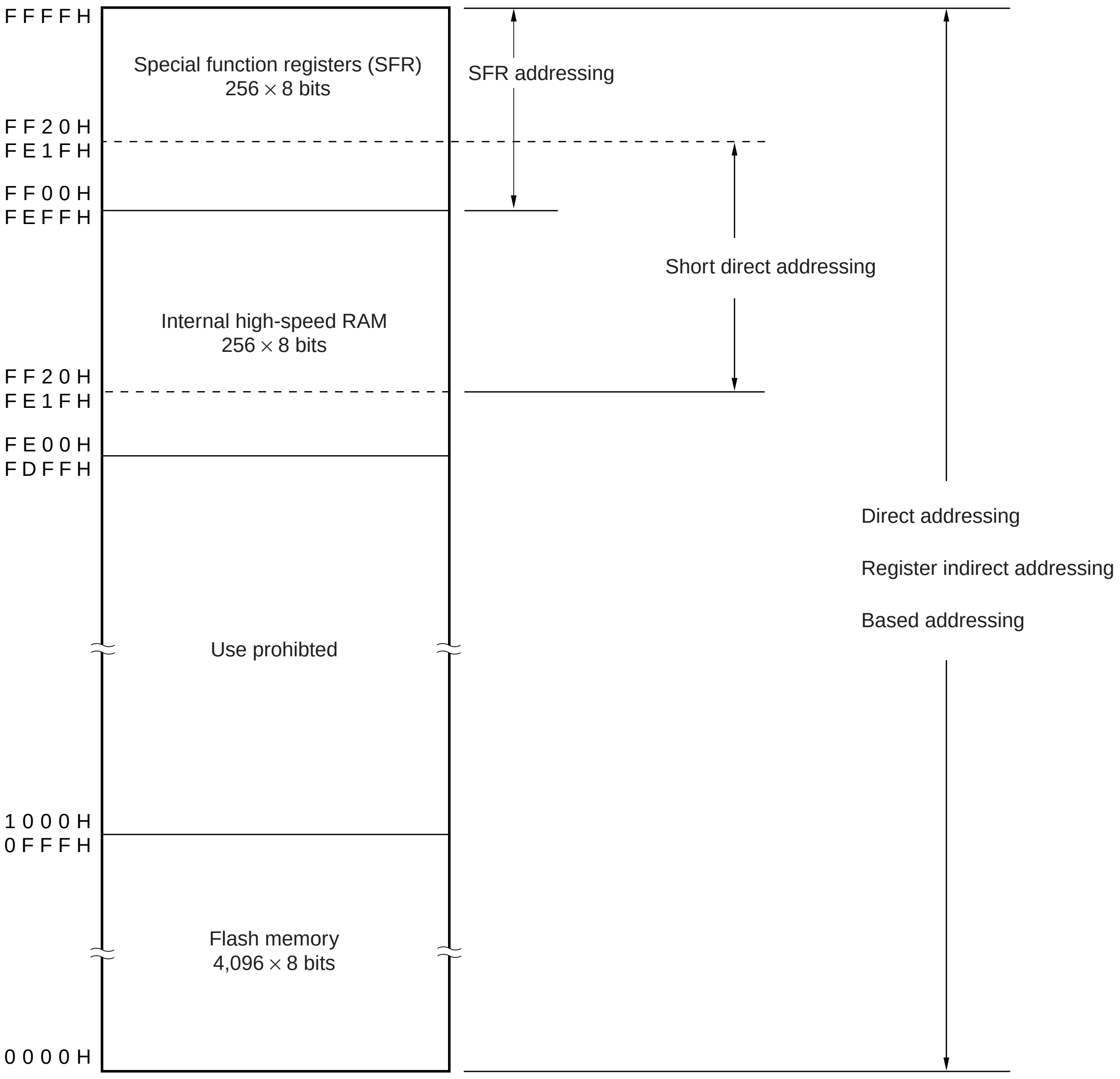
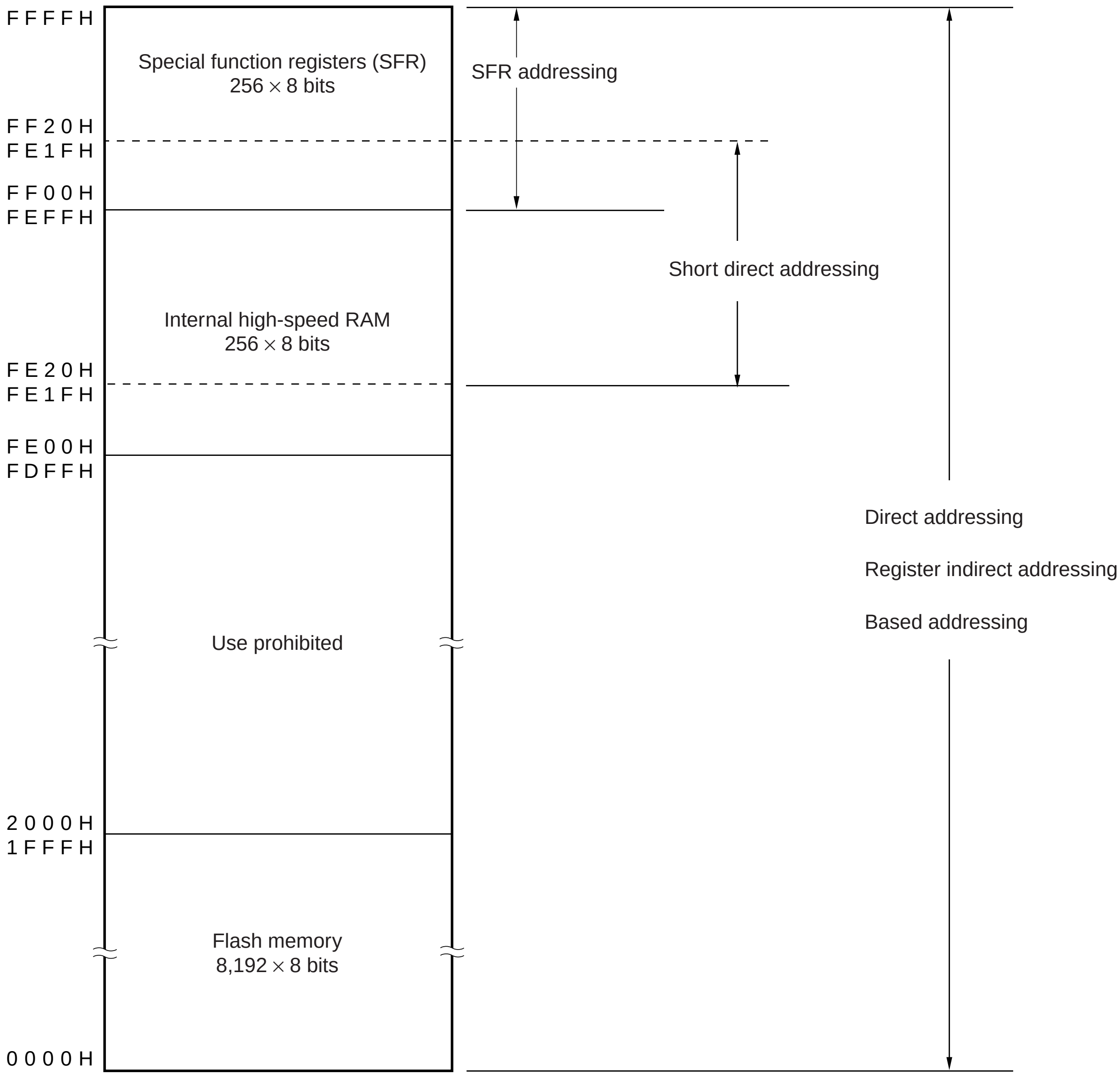


**Figure 3-4. Data Memory Addressing (μPD78F9234)**



## 3.2 Processor Registers

The 78K0S/KB1+ provides the following on-chip processor registers.

### 3.2.1 Control registers

The control registers have special functions to control the program sequence statuses and stack memory. The control registers include a program counter, a program status word, and a stack pointer.

**(1) Program counter (PC)**

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

In normal operation, the PC is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data or register contents are set.

$\overline{\text{RESET}}$ input sets the reset vector table values at addresses 0000H and 0001H to the program counter.

**Figure 3-5. Program Counter Configuration**

| | 15 | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PC | PC15 | PC14 | PC13 | PC12 | PC11 | PC10 | PC9 | PC8 | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |

**(2) Program status word (PSW)**

The program status word is an 8-bit register consisting of various flags to be set/reset by instruction execution.

Program status word contents are automatically stacked upon interrupt request generation or PUSH PSW instruction execution and are automatically restored upon execution of the RETI and POP PSW instructions.

$\overline{\text{RESET}}$ input sets PSW to 02H.

**Figure 3-6. Program Status Word Configuration**

| | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| PSW | IE | Z | 0 | AC | 0 | 0 | 1 | CY |

**(a) Interrupt enable flag (IE)**

This flag controls interrupt request acknowledge operations of the CPU.

When IE = 0, the interrupt disabled (DI) status is set. All interrupt requests are disabled.

When IE = 1, the interrupt enabled (EI) status is set. Interrupt request acknowledgment is controlled with an interrupt mask flag for various interrupt sources.

This flag is reset to 0 upon DI instruction execution or interrupt acknowledgment and is set to 1 upon EI instruction execution.

**(b) Zero flag (Z)**

When the operation result is zero, this flag is set to 1. It is reset to 0 in all other cases.

**(c) Auxiliary carry flag (AC)**

If the operation result has a carry from bit 3 or a borrow at bit 3, this flag is set to 1. It is reset to 0 in all other cases.

**(d) Carry flag (CY)**

This flag stores overflow and underflow that have occurred upon add/subtract instruction execution. It stores the shift-out value upon rotate instruction execution and functions as a bit accumulator during bit operation instruction execution.

**(3) Stack pointer (SP)**

This is a 16-bit register to hold the start address of the memory stack area. Only the internal high-speed RAM area can be set as the stack area (the stack area cannot be set except internal high-speed RAM area).

**Figure 3-7. Stack Pointer Configuration**

| | 15 | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SP | SP15 | SP14 | SP13 | SP12 | SP11 | SP10 | SP9 | SP8 | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |

The SP is decremented before writing (saving) to the stack memory and is incremented after reading (restoring) from the stack memory.

Each stack operation saves/restores data as shown in Figures 3-8 and 3-9.

**Caution Since generation of reset signal makes the SP contents undefined, be sure to initialize the SP before using the stack memory.**

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

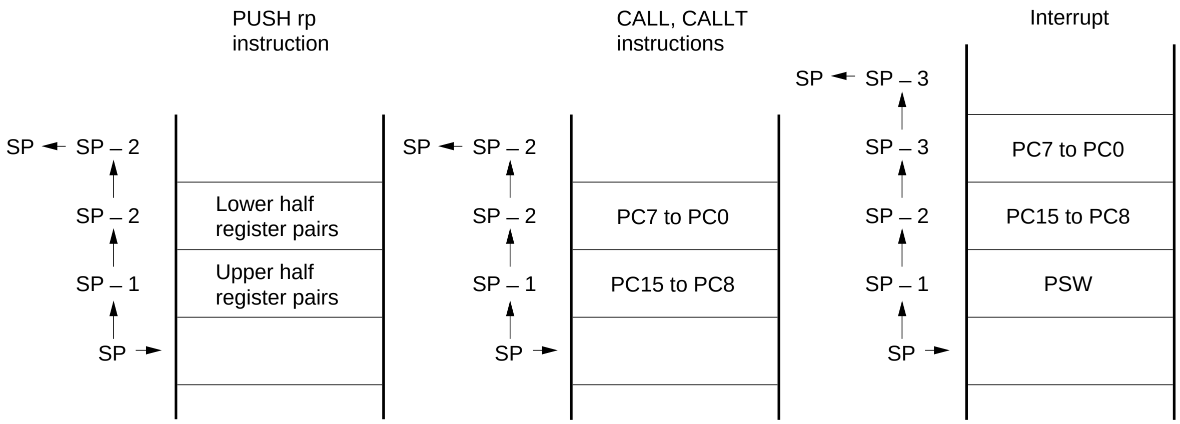
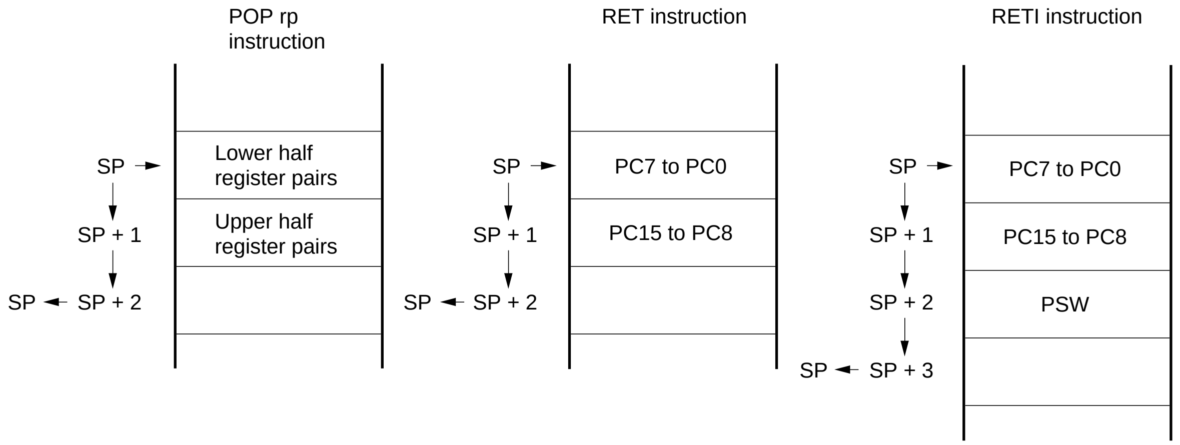


**Figure 3-9. Data to Be Restored from Stack Memory**



### 3.2.2 General-purpose registers

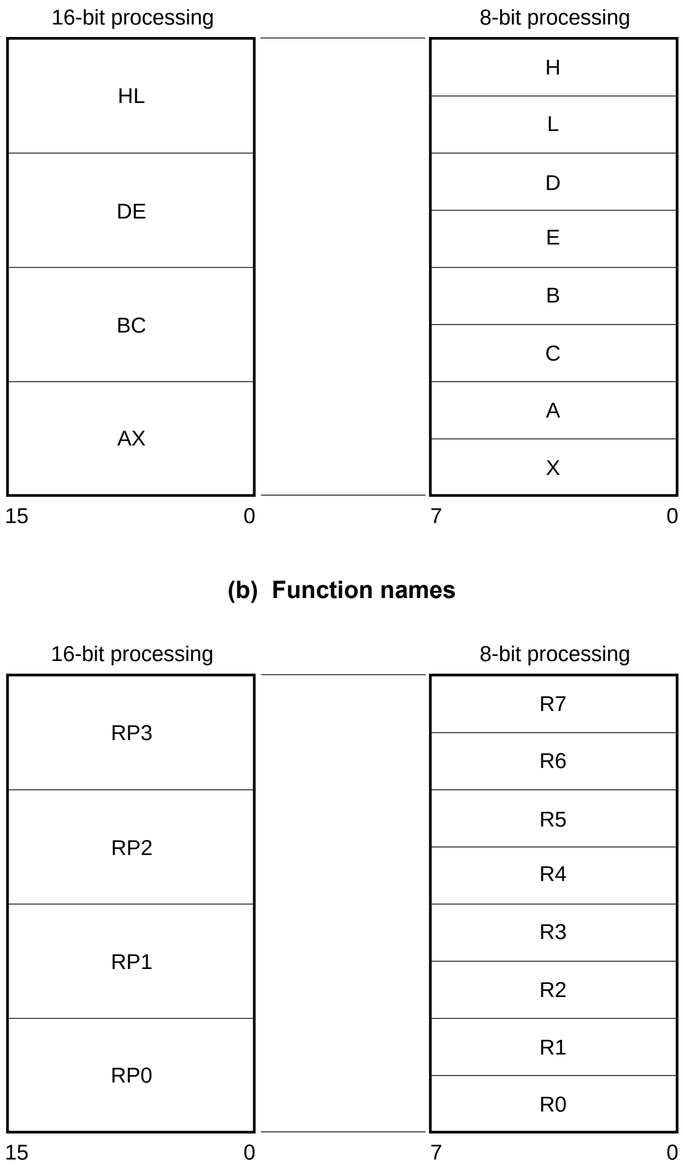
A general-purpose register consists of eight 8-bit registers (X, A, C, B, E, D, L, and H).

In addition each register being used as an 8-bit register, two 8-bit registers in pairs can be used as a 16-bit register (AX, BC, DE, and HL).

Registers can be described in terms of function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL) and absolute names (R0 to R7 and RP0 to RP3).

**Figure 3-10. General-Purpose Register Configuration**

**(a) Absolute names**



**(b) Function names**

### 3.2.3 Special function registers (SFRs)

Unlike the general-purpose registers, each special function register has a special function.

The special function registers are allocated to the 256-byte area FF00H to FFFFH.

The special function registers can be manipulated, like the general-purpose registers, with operation, transfer, and bit manipulation instructions. Manipulatable bit units (1, 8, and 16) differ depending on the special function register type.

Each manipulation bit unit can be specified as follows.

- 1-bit manipulation
  Describes a symbol reserved by the assembler for the 1-bit manipulation instruction operand (sfr.bit). This manipulation can also be specified with an address.

- 8-bit manipulation
  Describes a symbol reserved by the assembler for the 8-bit manipulation instruction operand (sfr). This manipulation can also be specified with an address.

- 16-bit manipulation
  Describes a symbol reserved by the assembler for the 16-bit manipulation instruction operand. When specifying an address, describe an even address.

Table 3-3 lists the special function registers. The meanings of the symbols in this table are as follows:

- Symbol
  Indicates the addresses of the implemented special function registers. It is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S. Therefore, these symbols can be used as instruction operands if an assembler or integrated debugger is used.

- R/W
  Indicates whether the special function register can be read or written.
  R/W: Read/write
  R: Read only
  W: Write only

- Number of bits manipulated simultaneously
  Indicates the bit units (1, 8, and 16) in which the special function register can be manipulated.

- After reset
  Indicates the status of the special function register when a reset signal is generated.

### Table 3-3. Special Function Registers (1/3)

| Address | Special Function Register (SFR) Name | Symbol | | R/W | Number of Bits Manipulated Simultaneously | | | After Reset |
|---|---|---|---|---|---|---|---|---|
| | | | | | 1 Bit | 8 Bits | 16 Bits | |
| FF00H | Port register 0 | P0 | | R/W Note 1 | √ | √ | – | 00H |
| FF02H | Port register 2 | P2 | | | √ | √ | – | |
| FF03H | Port register 3 | P3 | | | √ | √ | – | |
| FF04H | Port register 4 | P4 | | | √ | √ | – | |
| FF0CH | Port register 12 | P12 | | | √ | √ | – | |
| FF0DH | Port register 13 | P13 | | | √ | √ | – | |
| FF0EH | 8-bit timer H compare register 01 | CMP01 | | R/W | – | √ | – | |
| FF0FH | 8-bit timer H compare register 11 | CMP11 | | | – | √ | – | |
| FF10H | 16-bit Multiplication result storage register L | MUL0L | MUL0 | R | – | √ | √ | Undefined |
| FF11H | 16-bit Multiplication result storage register H | MUL0H | | | – | √ | | |
| FF12H | 16-bit timer counter 00 | TM00 | | | – | – | √Note 2 | 0000H |
| FF13H | | | | | | | | |
| FF14H | 16-bit timer capture/compare register 000 | CR000 | | R/W | – | – | √Note 2 | 0000H |
| FF15H | | | | | | | | |
| FF16H | 16-bit timer capture/compare register 010 | CR010 | | | – | – | √Note 2 | 0000H |
| FF17H | | | | | | | | |
| FF18H | 10-bit A/D conversion result register | ADCR | | R | – | – | √Note 2 | Undefined |
| FF19H | | | | | | | | |
| FF1AH | 8-bit A/D conversion result register | ADCRH | | | – | √ | – | |
| FF20H | Port mode register 0 | PM0 | | R/W | √ | √ | – | FFH |
| FF22H | Port mode register 2 | PM2 | | | √ | √ | – | |
| FF23H | Port mode register 3 | PM3 | | | √ | √ | – | |
| FF24H | Port mode register 4 | PM4 | | | √ | √ | – | |
| FF2CH | Port mode register 12 | PM12 | | | √ | √ | – | |
| FF30H | Pull-up resistance option register 0 | PU0 | | | √ | √ | – | 00H |
| FF32H | Pull-up resistance option register 2 | PU2 | | | √ | √ | – | |
| FF33H | Pull-up resistance option register 3 | PU3 | | | √ | √ | – | |
| FF34H | Pull-up resistance option register 4 | PU4 | | | √ | √ | – | |
| FF3CH | Pull-up resistance option register 12 | PU12 | | | √ | √ | – | |
| FF48H | Watchdog timer mode register | WDTM | | | – | √ | – | 67H |
| FF49H | Watchdog timer enable register | WDTE | | | – | √ | – | 9AH |
| FF50H | Low voltage detect register | LVIM | | | √ | √ | – | 00HNote 3 |
| FF51H | Low voltage detection level select register | LVIS | | | – | √ | – | |
| FF54H | Reset control flag register | RESF | | R | – | √ | – | 00HNote 4 |
| FF58H | Low-speed Ring-OSC mode register | LSRCM | | R/W | √ | √ | – | 00H |

**Notes 1.** Only P34 is an input-only port.

**2.** A 16-bit access is possible only by the short direct addressing.

**3.** Retained only after a reset by LVI.

**4.** Varies depending on the reset cause.

**Table 3-3. Special Function Registers (2/3)**

| Address | Special Function Register (SFR) Name | Symbol | R/W | Number of Bits Manipulated Simultaneously | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 Bit | 8 Bits | 16 Bits | |
| FF60H | 16-bit timer mode control register 00 | TMC00 | R/W | √ | √ | – | 00H |
| FF61H | Prescaler mode register 00 | PRM00 | | √ | √ | – | |
| FF62H | Capture/compare control register 00 | CRC00 | | √ | √ | – | |
| FF63H | 16-bit timer output control register 00 | TOC00 | | √ | √ | – | |
| FF70H | 8-bit timer H mode register 1 | TMHMD1 | | √ | √ | – | |
| FF80H | A/D converter mode register | ADM | | √ | √ | – | |
| FF81H | Analog input channel specify register | ADS | | √ | √ | – | |
| FF84H | Port mode control register 2 | PMC2 | | √ | √ | – | |
| FF8CH | Input switching control register | ISC | | √ | √ | – | |
| FF90H | Asynchronous serial interface operation mode register 6 | ASIM6 | | √ | √ | – | 01H |
| FF92H | Reception buffer register 6 | RXB6 | R | – | √ | – | FFH |
| FF93H | Asynchronous serial interface reception error status register 6 | ASIS6 | | – | √ | – | 00H |
| FF94H | Transmission buffer register 6 | TXB6 | R/W | – | √ | – | FFH |
| FF95H | Asynchronous serial interface transmission status register 6 | ASIF6 | R | – | √ | – | 00H |
| FF96H | Clock selection register 6 | CKSR6 | R/W | – | √ | – | |
| FF97H | Baud rate generator control register 6 | BRGC6 | | – | √ | – | FFH |
| FF98H | Asynchronous serial interface control register 6 | ASICL6 | | √ | √ | – | 16H |
| FFA0H | Flash protect command register | PFCMD | W | – | √ | – | Undefined |
| FFA1H | Flash status register | PFS | R/W | √ | √ | – | 00H |
| FFA2H | Flash programming mode control register | FLPMC | | – | √ | – | Undefined |
| FFA3H | Flash programming command register | FLCMD | | √ | √ | – | 00H |
| FFA4H | Flash address pointer L | FLAPL | | √ | √ | – | Undefined |
| FFA5H | Flash address pointer H | FLAPH | | √ | √ | – | |
| FFA6H | Flash address pointer H compare register | FLAPHC | | √ | √ | – | 00H |
| FFA7H | Flash address pointer L compare register | FLAPLC | | √ | √ | – | |
| FFA8H | Flash write buffer register | FLW | | – | √ | – | |
| FFCCH | 8-bit timer mode control register 80 | TMC80 | | √ | √ | – | |
| FFCDH | 8-bit compare register 80 | CR80 | W | – | √ | – | Undefined |
| FFCEH | 8-bit timer counter 80 | TM80 | R | – | √ | – | 00H |
| FFD0H | Multiplication data register A | MRA0 | W | – | √ | – | Undefined |
| FFD1H | Multiplication data register B | MRB0 | | – | √ | – | |
| FFD2H | Multiplier control register 0 | MULC0 | R/W | √ | √ | – | 00H |
| FFE0H | Interrupt request flag register 0 | IF0 | | √ | √ | – | |
| FFE1H | Interrupt request flag register 1 | IF1 | | √ | √ | – | |
| FFE4H | Interrupt mask flag register 0 | MK0 | | √ | √ | – | FFH |
| FFE5H | Interrupt mask flag register 1 | MK1 | | √ | √ | – | |
| FFECH | External interrupt mode register 0 | INTM0 | | – | √ | – | 00H |

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

| Address | Special Function Register (SFR) Name | Symbol | R/W | Number of Bits Manipulated Simultaneously | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 Bit | 8 Bits | 16 Bits | |
| FFEDH | External interrupt mode register 1 | INTM1 | | – | √ | – | 00H |
| FFF3H | Preprocessor clock control register | PPCC | | √ | √ | – | 02H |
| FFF4H | Oscillation stabilization time selection register | OSTS | | – | √ | – | Undefined[Note] |
| FFFBH | Processor clock control register | PCC | | √ | √ | – | 02H |

**Note** The oscillation stabilization time that elapses after release of reset is selected by the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**.

## 3.3 Instruction Address Addressing

An instruction address is determined by the program counter (PC) contents. The PC contents are normally incremented (+1 for each byte) automatically according to the number of bytes of an instruction to be fetched each time another instruction is executed. When a branch instruction is executed, the branch destination address information is set to the PC to branch by the following addressing (for details of each instruction, refer to **78K/0S Series Instructions User's Manual (U11047E)**).

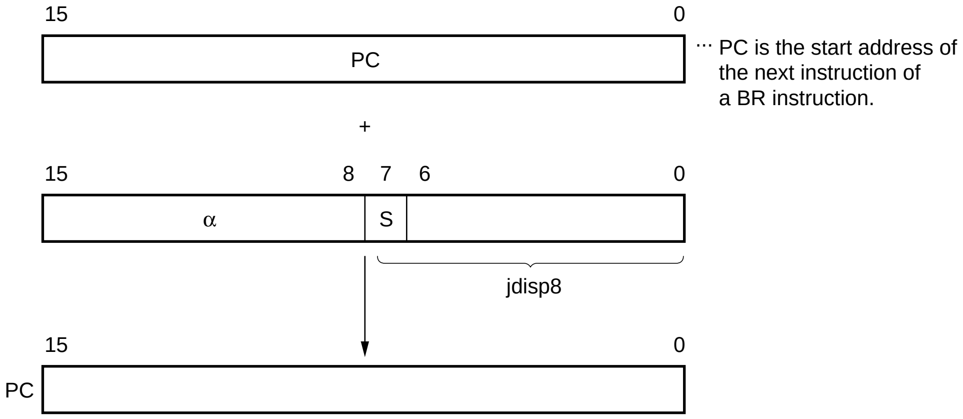
### 3.3.1 Relative addressing

**[Function]**

The value obtained by adding 8-bit immediate data (displacement value: jdisp8) of an instruction code to the start address of the following instruction is transferred to the program counter (PC) to branch. The displacement value is treated as signed two's complement data (–128 to +127) and bit 7 becomes the sign bit. In other words, the range of branch in relative addressing is between –128 and +127 of the start address of the following instruction.

This function is carried out when the BR $addr16 instruction or a conditional branch instruction is executed.

**[Illustration]**



··· PC is the start address of the next instruction of a BR instruction.

When S = 0, α indicates that all bits are “0”.
When S = 1, α indicates that all bits are “1”.

### 3.3.2 Immediate addressing

**[Function]**

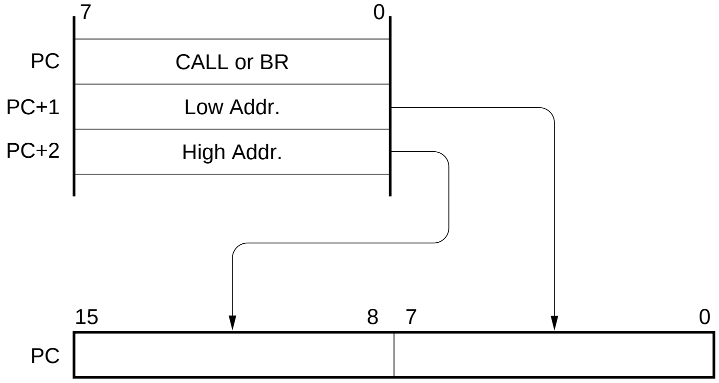
Immediate data in the instruction word is transferred to the program counter (PC) to branch.

This function is carried out when the CALL !addr16 and BR !addr16 instructions are executed.

CALL !addr16 and BR !addr16 instructions can be used to branch to all the memory spaces.

**[Illustration]**

In case of CALL !addr16 and BR !addr16 instructions



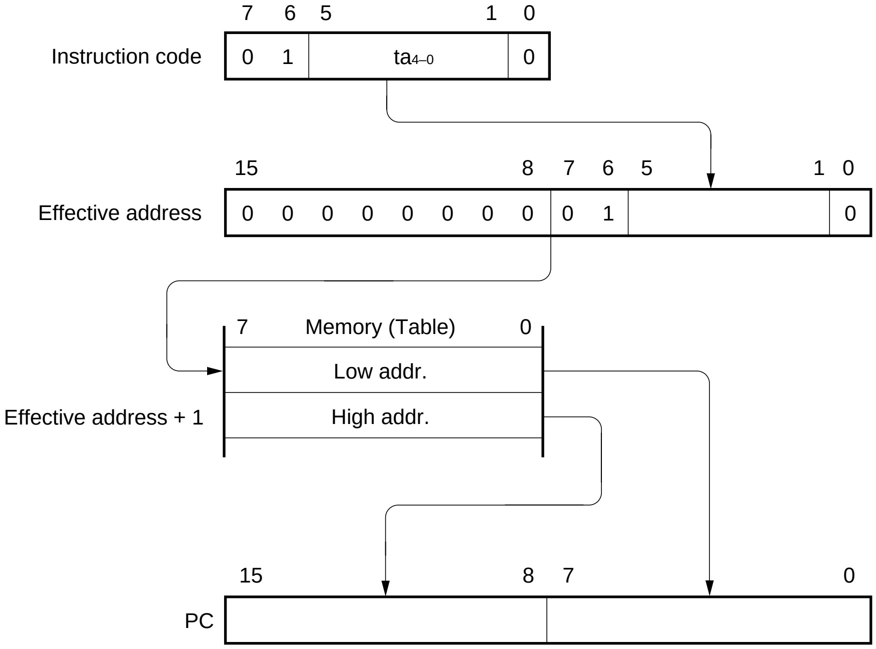
### 3.3.3 Table indirect addressing

**[Function]**

The table contents (branch destination address) of the particular location to be addressed by the immediate data of an instruction code from bit 1 to bit 5 are transferred to the program counter (PC) to branch.

Table indirect addressing is carried out when the CALLT [addr5] instruction is executed. This instruction can be used to branch to all the memory spaces according to the address stored in the memory table 40H to 7FH.

**[Illustration]**



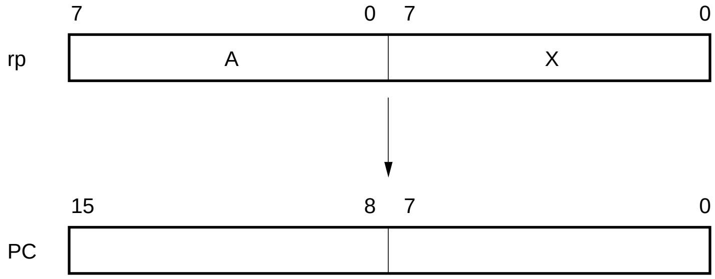
### 3.3.4 Register addressing

**[Function]**

The register pair (AX) contents to be specified with an instruction word are transferred to the program counter (PC) to branch.

This function is carried out when the BR AX instruction is executed.

**[Illustration]**



## 3.4 Operand Address Addressing

The following methods (addressing) are available to specify the register and memory to undergo manipulation during instruction execution.

### 3.4.1 Direct addressing

**[Function]**

The memory indicated by immediate data in an instruction word is directly addressed.

**[Operand format]**

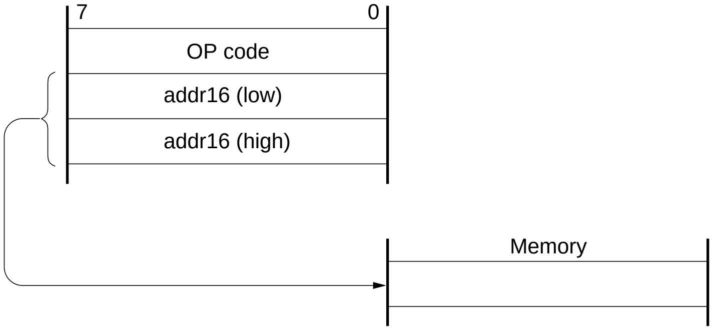
| Identifier | Description |
|---|---|
| addr16 | Label or 16-bit immediate data |

**[Description example]**

MOV A, !0FE80H; When setting !addr16 to FE80H

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction code | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | OP code |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 80H |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | FEH |

**[Illustration]**



### 3.4.2 Short direct addressing

**[Function]**

The memory to be manipulated in the fixed space is directly addressed with the 8-bit data in an instruction word.

The fixed space where this addressing is applied is the 256-byte space FE20H to FF1FH. An internal high-speed RAM is mapped at FE20H to FEFFH and the special function registers (SFR) are mapped at FF00H to FF1FH.

The SFR area where short direct addressing is applied (FF00H to FF1FH) is a part of the total SFR area. In this area, ports which are frequently accessed in a program and a compare register of the timer counter are mapped, and these SFRs can be manipulated with a small number of bytes and clocks.

When 8-bit immediate data is at 20H to FFH, bit 8 of an effective address is cleared to 0. When it is at 00H to 1FH, bit 8 is set to 1. See **[Illustration]** below.

**[Operand format]**

| Identifier | Description |
|---|---|
| saddr | Label or FE20H to FF1FH immediate data |
| saddrp | Label or FE20H to FF1FH immediate data (even address only) |

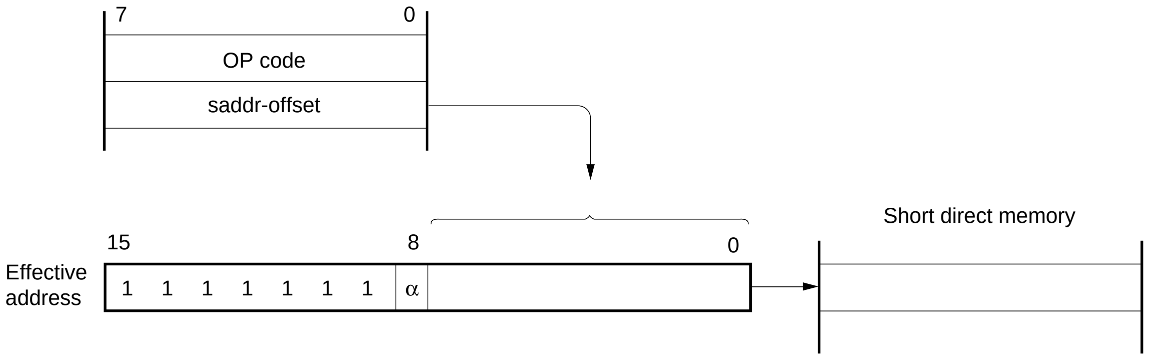
**[Description example]**

EQU DATA1 0FE90H; DATA1 indicates FE90H in saddr area

MOV DATA1, #50H; When the immediate data to 50H

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction code | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | OP code |
| | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 90H (saddr-offset) |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 50H (immediate data) |

**[Illustration]**



When 8-bit immediate data is 20H to FFH, $\alpha = 0$.
When 8-bit immediate data is 00H to 1FH, $\alpha = 1$.

### 3.4.3 Special function register (SFR) addressing

**[Function]**

A memory-mapped special function register (SFR) is addressed with the 8-bit immediate data in an instruction word.

This addressing is applied to the 256-byte space FF00H to FFFFH. However, SFRs mapped at FF00H to FF1FH are accessed with short direct addressing.

**[Operand format]**

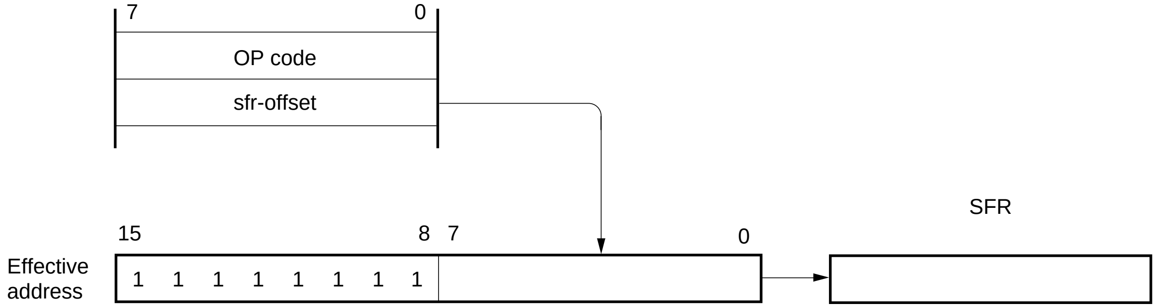
| Identifier | Description |
|---|---|
| sfr | Special function register name |

**[Description example]**

MOV PM0, A; When selecting PM0 for sfr

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instruction code | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**[Illustration]**



### 3.4.4 Register addressing

**[Function]**

A general-purpose register is accessed as an operand.

The general-purpose register to be accessed is specified with the register specify code and functional name in the instruction code.

Register addressing is carried out when an instruction with the following operand format is executed. When an 8-bit register is specified, one of the eight registers is specified with 3 bits in the instruction code.

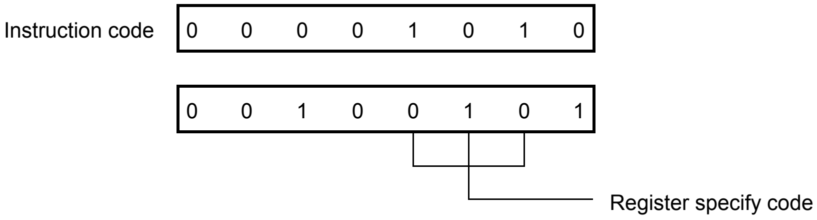
**[Operand format]**

| Identifier | Description |
|---|---|
| r | X, A, C, B, E, D, L, H |
| rp | AX, BC, DE, HL |

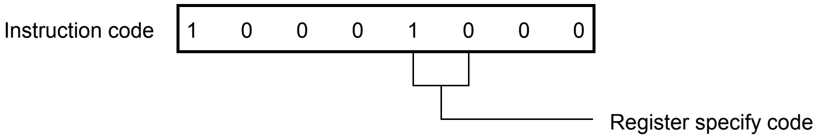
‘r’ and ‘rp’ can be described with absolute names (R0 to R7 and RP0 to RP3) as well as function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL).

**[Description example]**

MOV A, C; When selecting the C register for r



INCW DE; When selecting the DE register pair for rp



### 3.4.5 Register indirect addressing

**[Function]**

The memory is addressed with the contents of the register pair specified as an operand. The register pair to be accessed is specified with the register pair specify code in the instruction code. This addressing can be carried out for all the memory spaces.

**[Operand format]**

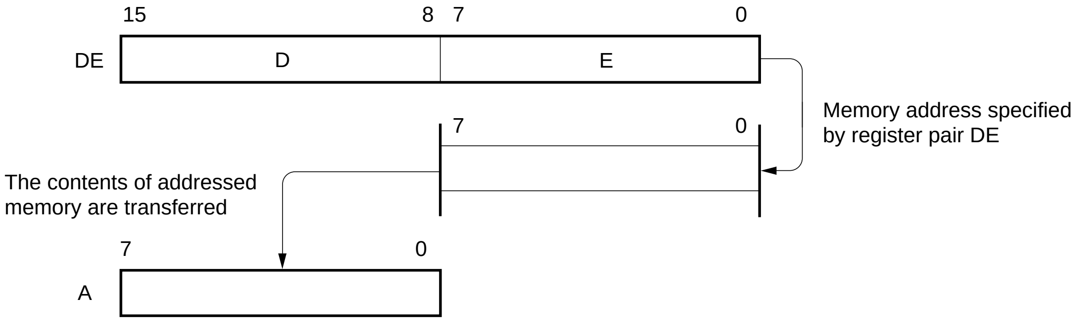
| Identifier | Description |
|---|---|
| – | [DE], [HL] |

**[Description example]**

MOV A, [DE]; When selecting register pair [DE]

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instruction code | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

**[Illustration]**



### 3.4.6 Based addressing

**[Function]**

8-bit immediate data is added to the contents of the base register, that is, the HL register pair, and the sum is used to address the memory. Addition is performed by expanding the offset data as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

**[Operand format]**

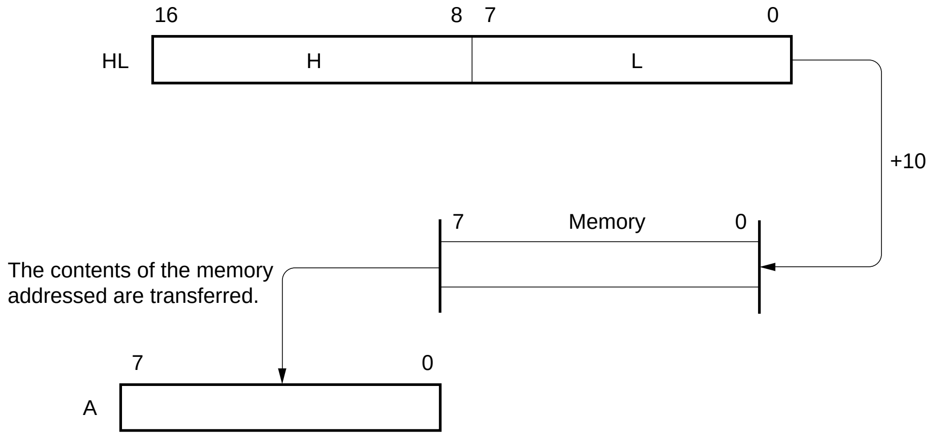
| Identifier | Description |
|---|---|
| – | [HL+byte] |

**[Description example]**

MOV A, [HL+10H]; When setting byte to 10H

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instruction code | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**[Illustration]**



### 3.4.7 Stack addressing

**[Function]**

The stack area is indirectly addressed with the stack pointer (SP) contents.

This addressing method is automatically employed when the PUSH, POP, subroutine call, and return instructions are executed or the register is saved/restored upon interrupt request generation.

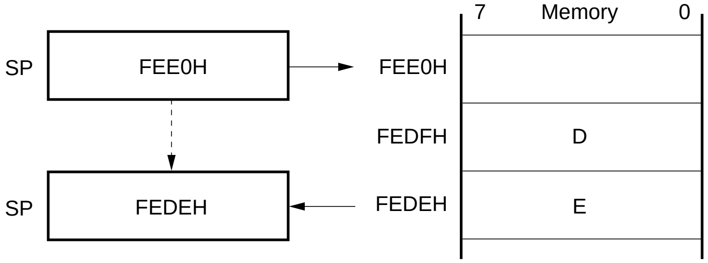
Stack addressing can be used to access the internal high-speed RAM area only.

**[Description example]**

In the case of PUSH DE

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instruction code | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

**[Illustration]**



# CHAPTER 4 PORT FUNCTIONS

## 4.1 Functions of Ports

The 78K0S/KB1+ has the ports shown in Figure 4-1, which can be used for various control operations. Table 4-1 shows the functions of each port.

In addition to digital I/O port functions, each of these ports has an alternate function. For details, refer to **CHAPTER 2 PIN FUNCTIONS**.

**Figure 4-1. Port Functions**

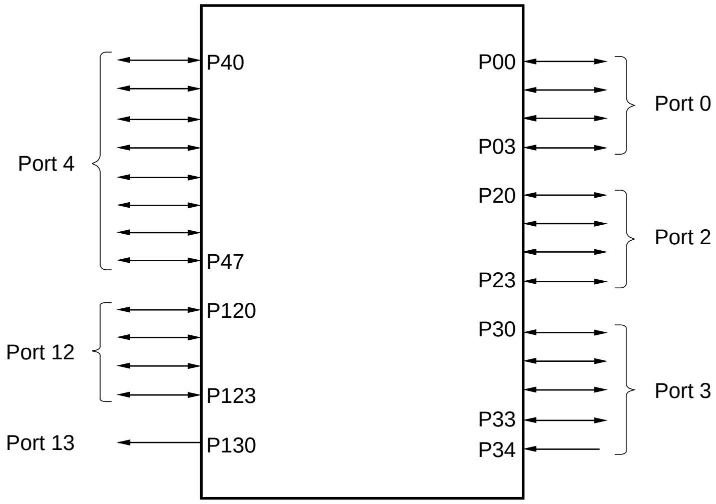


**Table 4-1. Port Functions**

| Pin Name | I/O | Function | | After Reset | Alternate-Function Pin |
|---|---|---|---|---|---|
| P00 to P03 | I/O | Port 0.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>On-chip pull-up resistor can be connected by setting software. | | Input | – |
| P20 to P23 | I/O | Port 2.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>On-chip pull-up resistor can be connected by setting software. | | Input | ANI0 to ANI3 |
| P30 | I/O | Port 3 | Can be set to input or output mode in 1-bit units.<br>On-chip pull-up resistor can be connected by setting software. | Input | TI000/INTP0 |
| P31 | | | | | TI010/TO00/INTP2 |
| P32 | | | | | – |
| P33 | | | | | – |
| P34[Note] | Input | | Input only | Input | $\overline{\text{RESET}}$[Note] |
| P40 | I/O | Port 4.<br>8-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>On-chip pull-up resistor can be connected setting software. | | Input | – |
| P41 | | | | | INTP3 |
| P42 | | | | | TOH1 |
| P43 | | | | | TxD6/INTP1 |
| P44 | | | | | RxD6 |
| P45 | | | | | – |
| P46 | | | | | – |
| P47 | | | | | – |
| P120 | I/O | Port 12.<br>4-bit I/O port.<br>Can be set to input or output mode in 1-bit units.<br>On-chip pull-up resistor can be connected only to P120 and P123 by setting software. | | Input | – |
| P121[Note] | | | | | X1[Note] |
| P122[Note] | | | | | X2[Note] |
| P123 | | | | | – |
| P130 | Output | Port 13.<br>1-bit output-only port. | | Output | – |

**Note** For settings of alternate function, refer to **CHAPTER 18 OPTION BYTE**.

**Caution The P121/X1 and P122/X2 pins are pulled down during reset.**

**Remarks 1.** P121 and P122 can be allocated when the high-speed Ring-OSC is selected as the system clock.

**2.** P122 can be allocated when an external clock is selected as the system clock.

## 4.2 Port Configuration

Ports consist of the following hardware units.

**Table 4-2. Configuration of Ports**

| Item | Configuration |
|---|---|
| Control registers | Port mode registers (PM0, PM2, PM3, PM4, PM12)<br>Port registers (P0, P2, P3, P4, P12, P13)<br>Port mode control register 2 (PMC2)<br>Pull-up resistor option registers (PU0, PU2, PU3, PU4, PU12) |
| Ports | Total: 26 (CMOS I/O: 24, CMOS input: 1, CMOS output: 1) |
| Pull-up resistor | Total: 22 |

### 4.2.1 Port 0

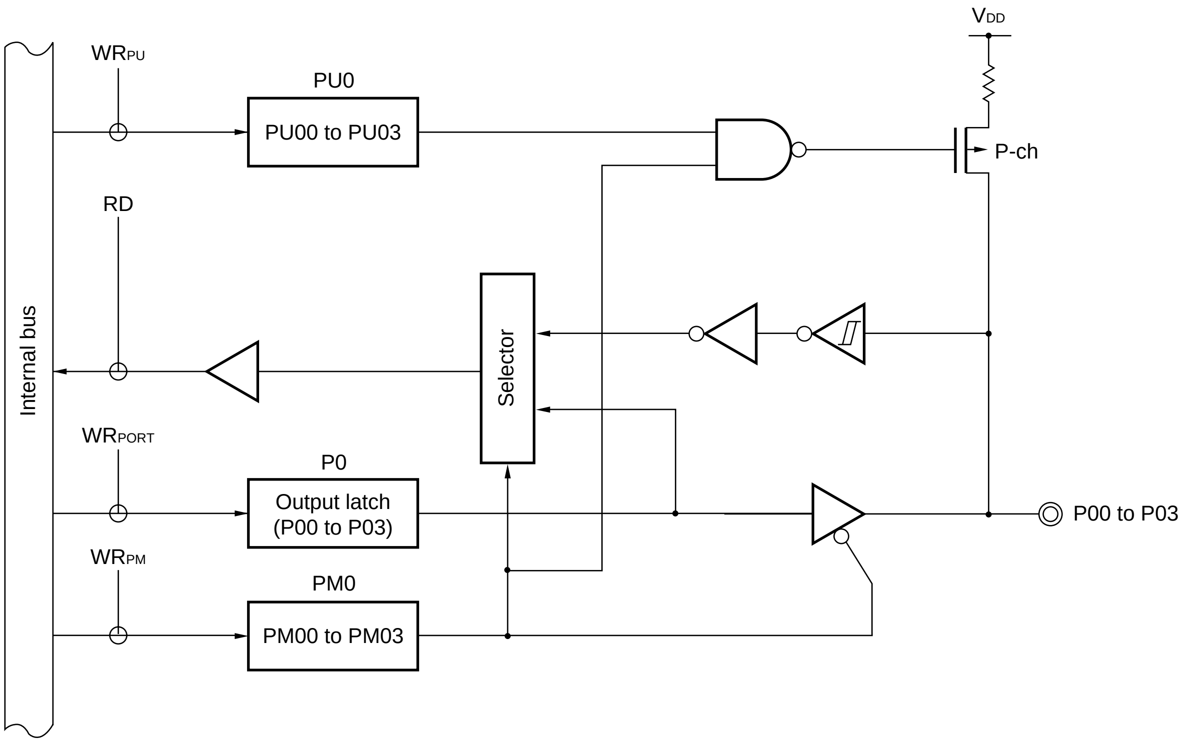
Port 0 is a 4-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 0 (PM0). When the P00 to P03 pins are used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 0 (PU0).

This port is also used as the analog input pins of the internal A/D converter.

Generation of reset signal sets port 0 to the input mode.

Figure 4-2 shows the block diagram of port 0.

**Figure 4-2. Block Diagram of P00 to P03**



PU0: Pull-up resistor option register 0

P0: Port register 0

PM0: Port mode register 0

RD: Read signal

WR××: Write signal

### 4.2.2 Port 2

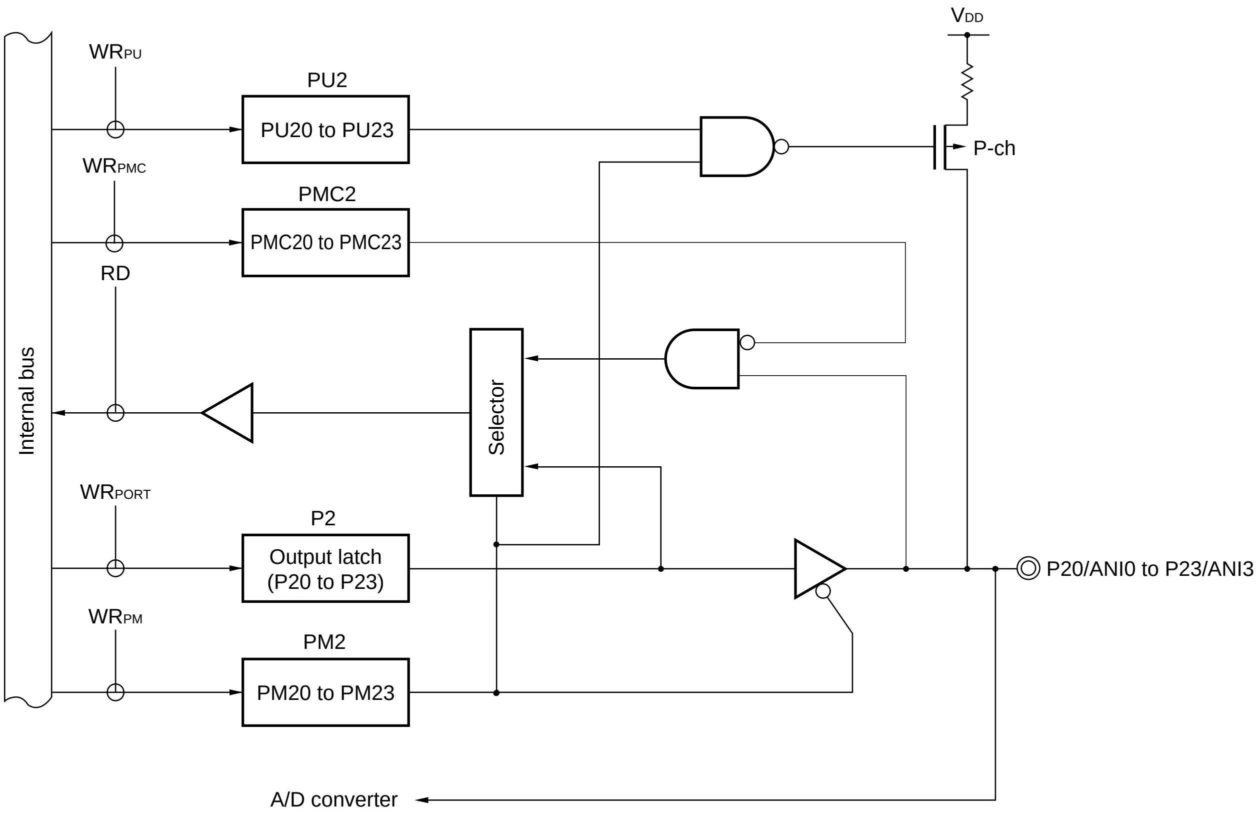
Port 2 is a 4-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 2 (PM2). When the P20 to P23 pins are used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 2 (PU2).

This port is also used as the analog input pins of the internal A/D converter.

Generation of reset signal sets port 2 to the input mode.

Figure 4-2 shows the block diagram of port 2.

**Figure 4-3. Block Diagram of P20 to P23**

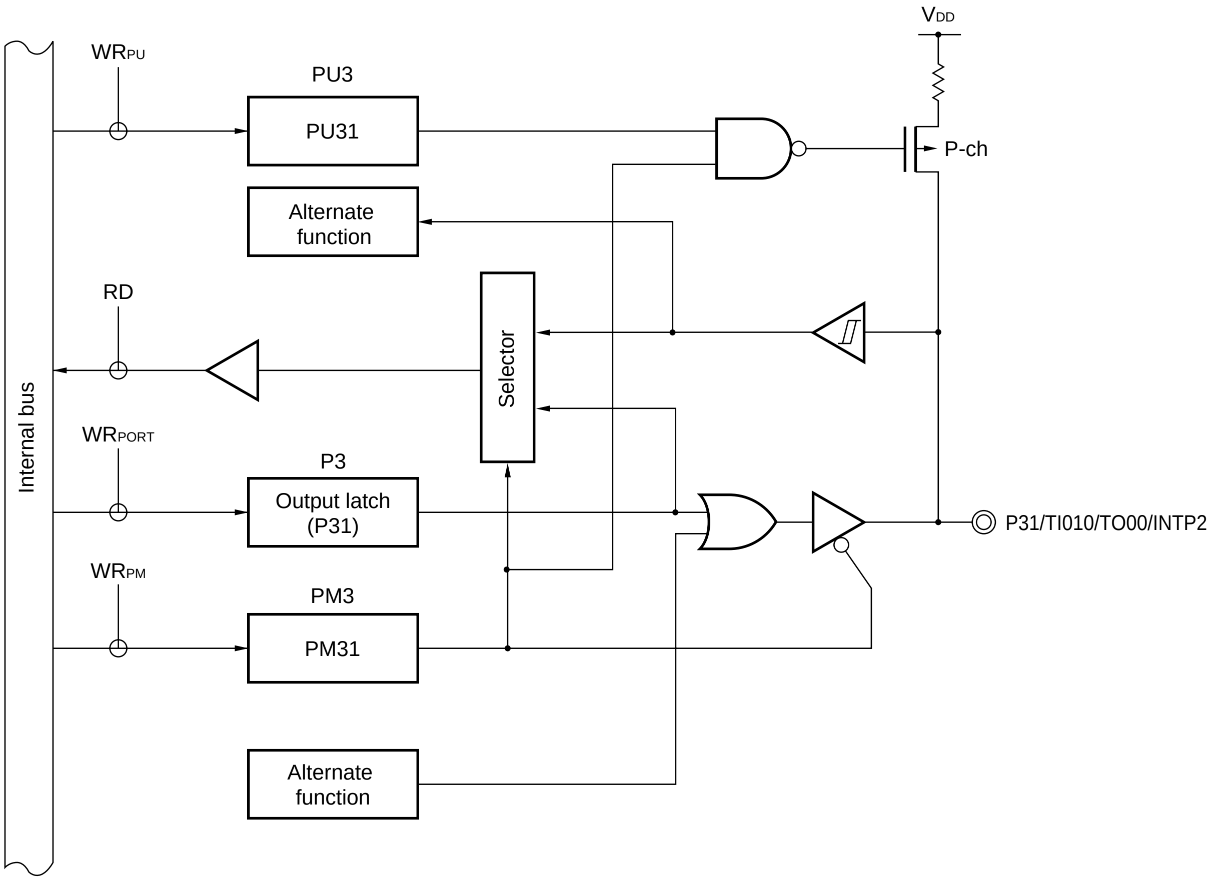


| | |
|---|---|
| PU2: | Pull-up resistor option register 2 |
| P2: | Port register 2 |
| PM2: | Port mode register 2 |
| PMC2: | Port mode control register 2 |
| RD: | Read signal |
| WR××: | Write signal |

### 4.2.3 Port 3

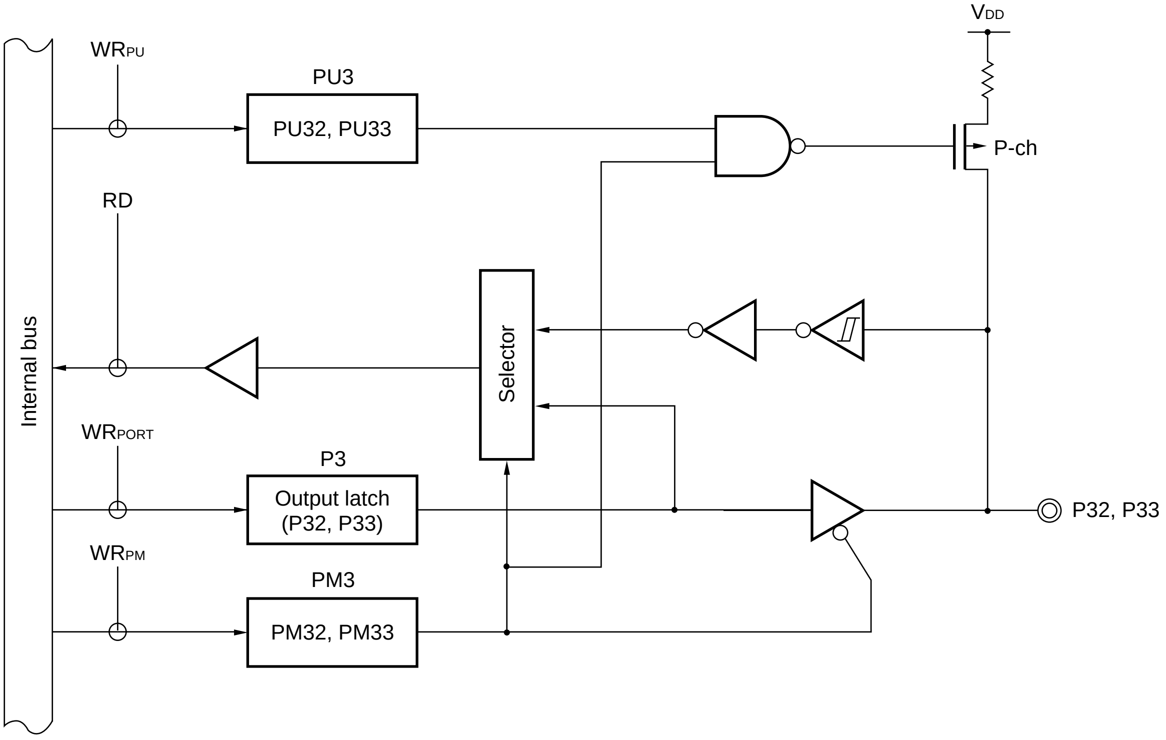
Pins P30 to P33 constitute a 4-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register

**Figure 4-5. Block Diagram of P31**



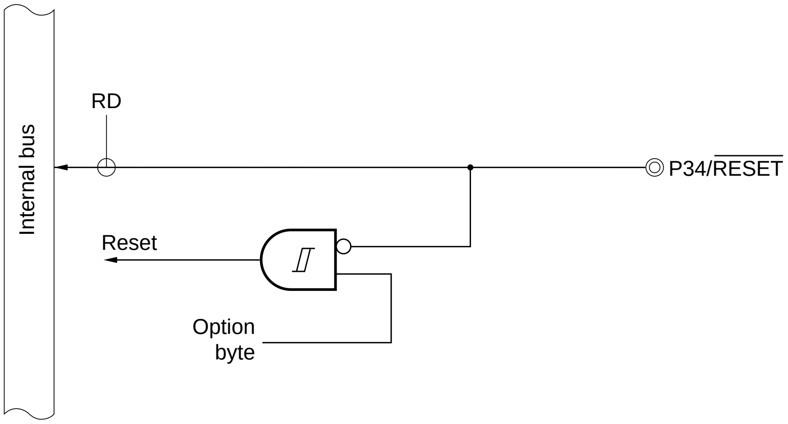
PU3: Pull-up resistor option register 3
P3: Port register 3
PM3: Port mode register 3
RD: Read signal
WR××: Write signal

**Figure 4-6. Block Diagram of P32 and P33**



| | |
|---|---|
| PU3: | Pull-up resistor option register 3 |
| P3: | Port register 3 |
| PM3: | Port mode register 3 |
| RD: | Read signal |
| WR××: | Write signal |

**Figure 4-7. Block Diagram of P34**



RD: Read signal

**Caution Because the P34 pin functions alternately as the $\overline{\text{RESET}}$ pin, if it is used as an input port pin, the function to input an external reset signal to the $\overline{\text{RESET}}$ pin cannot be used. The function of the port is selected by the option byte. For details, refer to CHAPTER 18 OPTION BYTE.**

**If a low level is input to the RESET pin before the option byte is referenced again after reset is released by the POC circuit, the 78K0S/KB1+ is reset and is held in the reset state until a high level is input to the RESET pin.**

### 4.2.4 Port 4

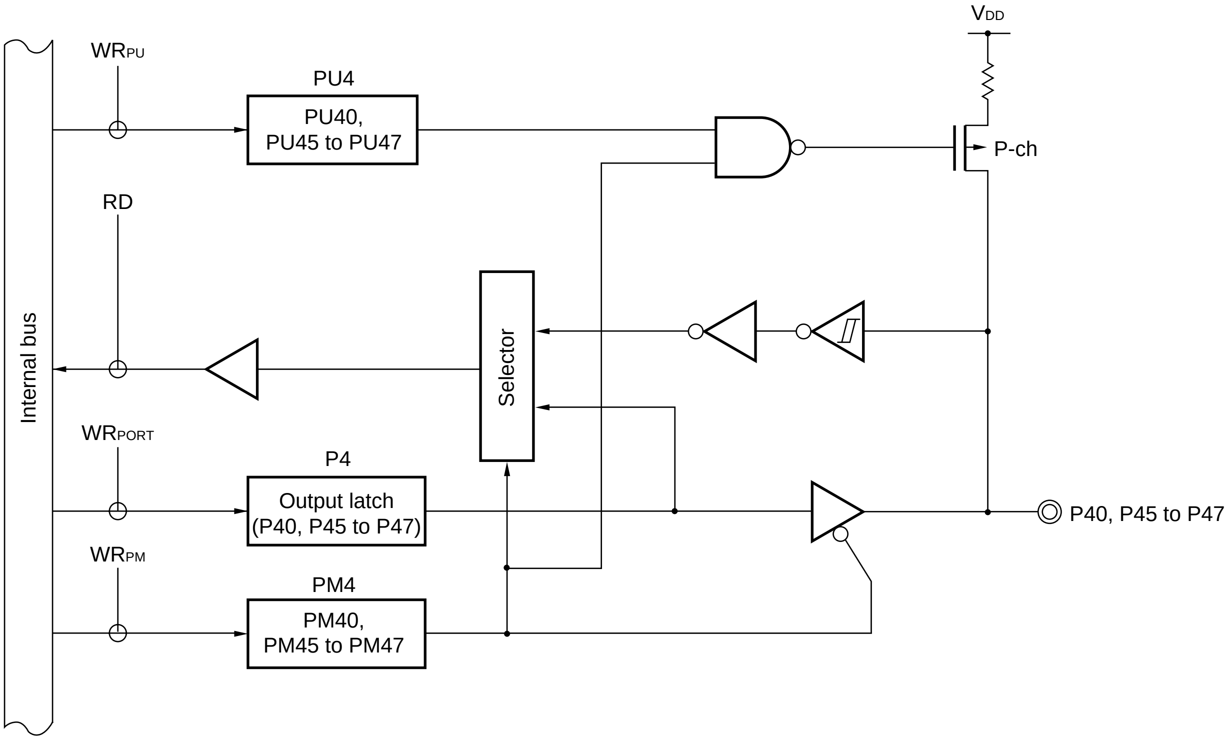
Port 4 is a 8-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 4 (PM4). When the P40 to P47 pins are used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 4 (PU4).

The P41 to P44 pins can also be used for external interrupt request input, serial interface data I/O, and timer output.

Generation of reset signal sets port 4 to the input mode.

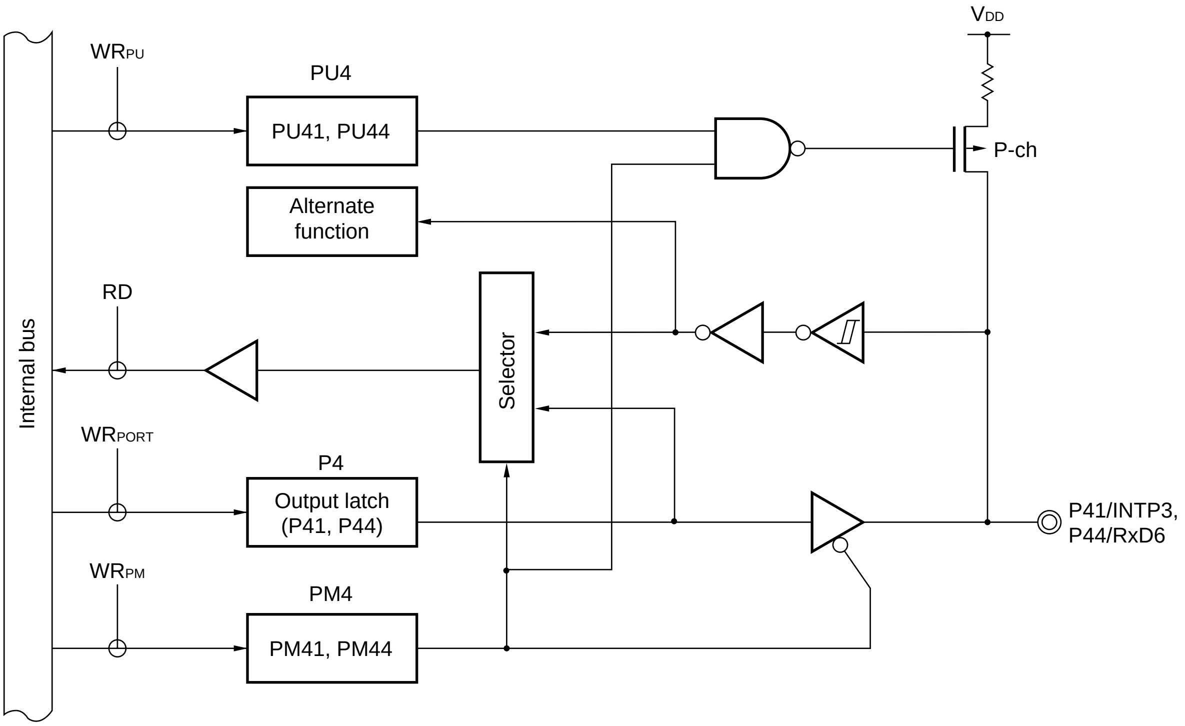
Figures 4-8 to 4-11 show the block diagrams of port 4.

**Figure 4-8. Block Diagram of P40, P45 to P47**



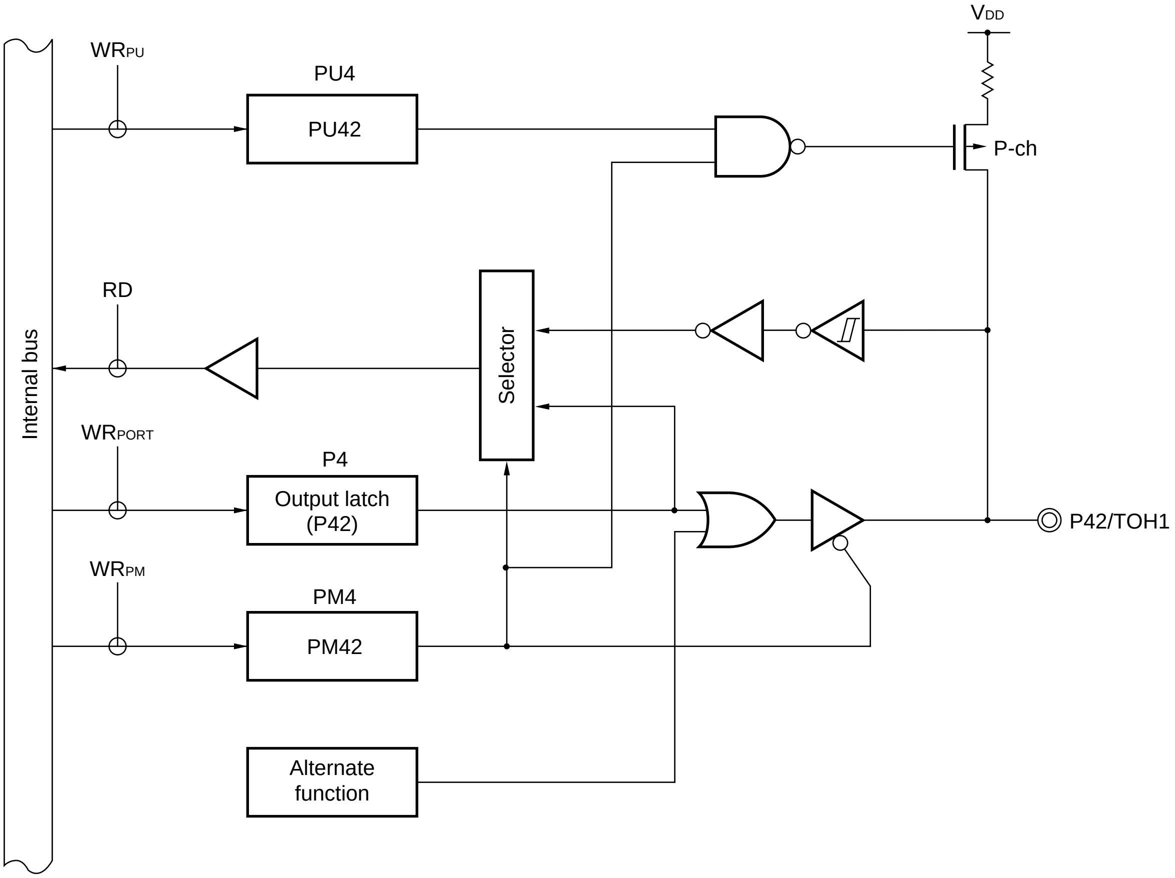
| | |
|---|---|
| PU4: | Pull-up resistor option register 4 |
| P4: | Port register 4 |
| PM4: | Port mode register 4 |
| RD: | Read signal |
| WR××: | Write signal |

Figure 4-9. Block Diagram of P41 and P44



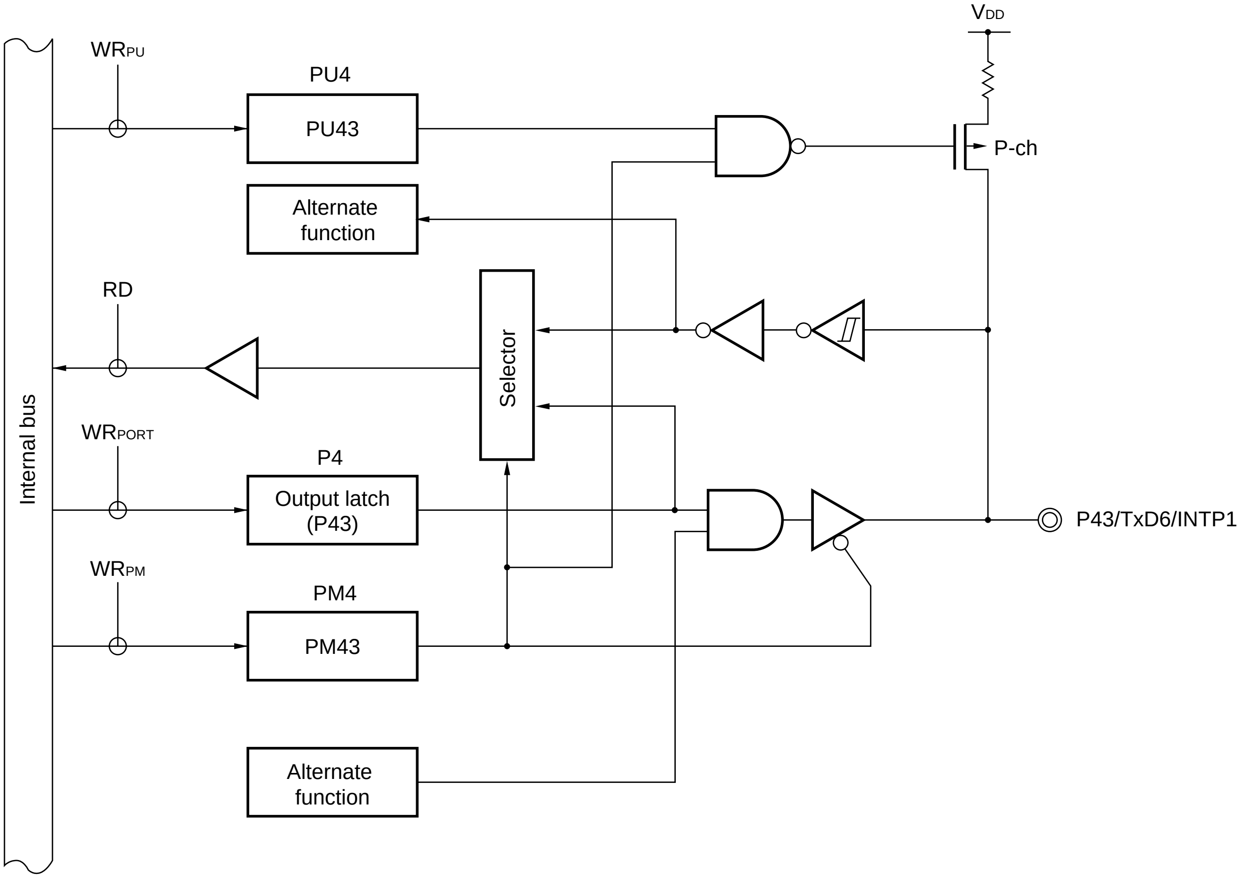
| | |
|---|---|
| PU4: | Pull-up resistor option register 4 |
| P4: | Port register 4 |
| PM4: | Port mode register 4 |
| RD: | Read signal |
| WR××: | Write signal |

**Figure 4-10. Block Diagram of P42**



| | |
|---|---|
| PU4: | Pull-up resistor option register 4 |
| P4: | Port register 4 |
| PM4: | Port mode register 4 |
| RD: | Read signal |
| WR××: | Write signal |

**Figure 4-11. Block Diagram of P43**



| | |
|---|---|
| PU4: | Pull-up resistor option register 4 |
| P4: | Port register 4 |
| PM4: | Port mode register 4 |
| RD: | Read signal |
| WR××: | Write signal |

### 4.2.5 Port 12

Port 12 is a 4-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 12 (PM12). When the P120 and P123 pins are used as an input port, an on-chip pull-up resistor can be connected by using pull-up resistor option register 12 (PU12).

The P121 and P122 pins are also used as the X1 and X2 pins of the system clock oscillator. The functions of the P121 and P122 pins differ, therefore, depending on the selected system clock oscillator. The following three system clock oscillators can be used.

**(1) High-speed Ring-OSC oscillator**

The P121 and P122 pins can be used as I/O port pins.

**(2) Crystal/ceramic oscillator**

The P121 and P122 pins cannot be used as I/O port pins because they are used as the X1 and X2 pins.

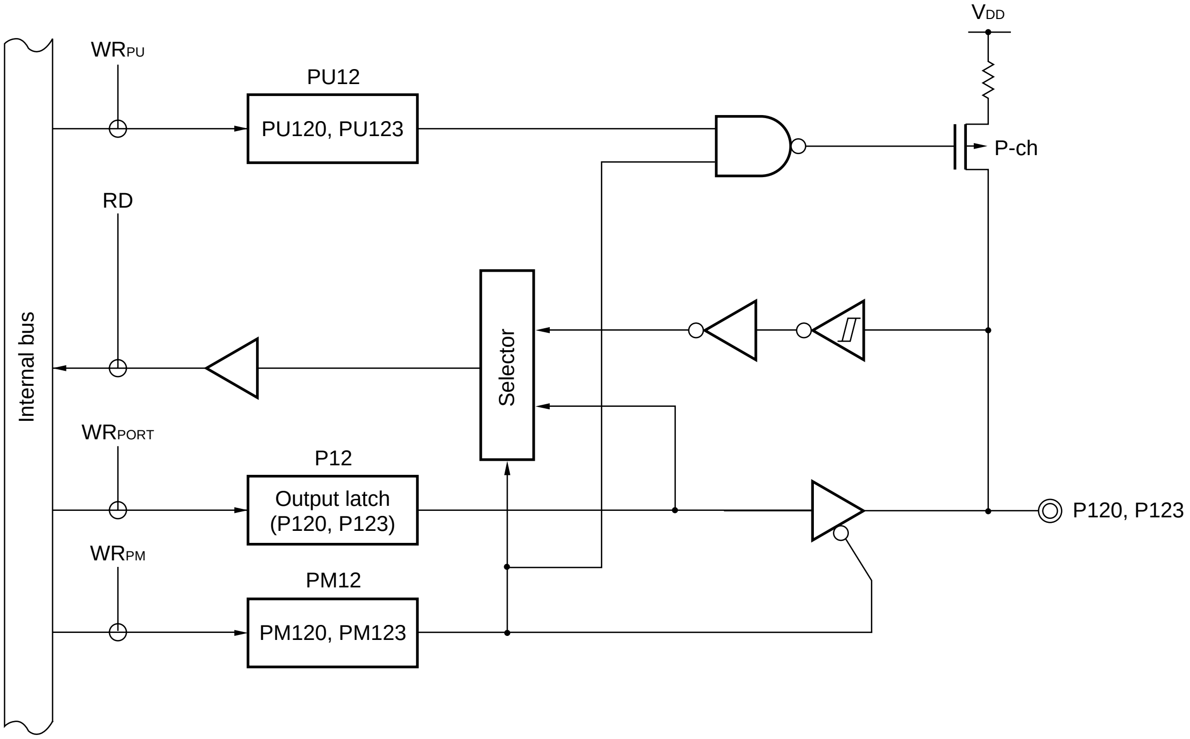
### (3) External clock input

The P121 pin is used as the X1 pin to input an external clock, and therefore it cannot be used as an I/O port pin. The P122 pin can be used as an I/O port pin.

The system clock oscillation is selected by the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**. Generation of reset signal sets port 12 to the input mode.

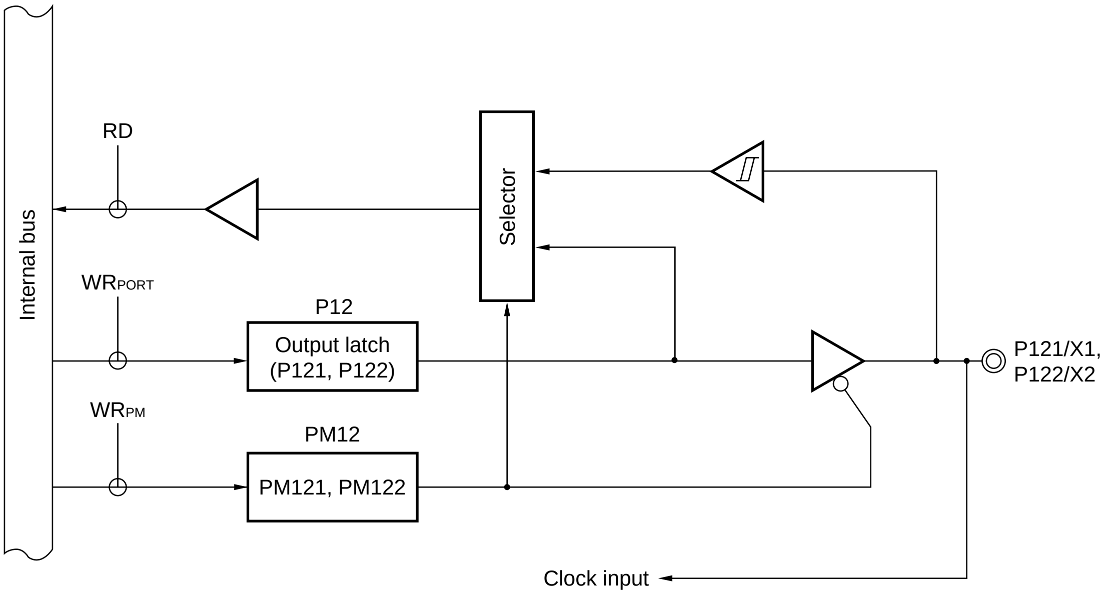
Figures 4-12 and 4-13 show the block diagrams of port 12.

**Figure 4-12. Block Diagram of P120 and P123**



| | |
|---|---|
| PU12: | Pull-up resistor option register 12 |
| P12: | Port register 12 |
| PM12: | Port mode register 12 |
| RD: | Read signal |
| WR××: | Write signal |

**Figure 4-13. Block Diagram of P121 and P122**



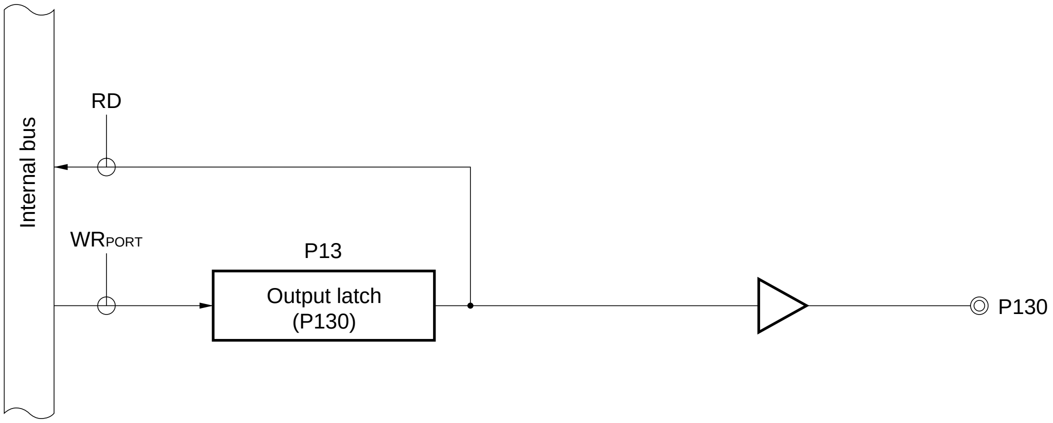
PM12: Port mode register 12
P12: Port register 12
RD: Read signal
WR××: Write signal

### 4.2.6 Port 13

This is a 1-bit output-only port.

Figure 4-14 shows the block diagram of port 13.

**Figure 4-14. Block Diagram of P130**



P13: Port register 13
RD: Read signal
WR××: Write signal

**Remark** When a reset is input, P130 outputs a low level. If P130 outputs a high level immediately after reset is released, the output signal of P130 can be used as a dummy CPU reset signal.

## 4.3 Registers Controlling Port Functions

The ports are controlled by the following four types of registers.

- Port mode registers (PM0, PM2, PM3, PM4, PM12)
- Port registers (P0, P2, P3, P4, P12, P13)
- Port mode control register 2 (PMC2)
- Pull-up resistor option registers (PU0, PU2, PU3, PU4, PU12)

### (1) Port mode registers (PM0, PM2, PM3, PM4, PM12)

These registers are used to set the corresponding port to the input or output mode in 1-bit units.

Each port mode register can be set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets these registers to FFH.

When a port pin is used as an alternate-function pin, set its port mode register and output latch as shown in Table 4-3.

**Caution Because P30, P31, and P43 are also used as external interrupt pins, the corresponding interrupt request flag is set if each of these pins is set to the output mode and its output level is changed. To use the port pin in the output mode, therefore, set the corresponding interrupt mask flag to 1 in advance.**

**Figure 4-15. Format of Port Mode Register**

Address: FF20H After reset: FFH R/W

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

Address: FF22H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM2 | 1 | 1 | 1 | 1 | PM23 | PM22 | PM21 | PM20 |

Address: FF23H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM3 | 1 | 1 | 1 | 1 | PM33 | PM32 | PM31 | PM30 |

Address: FF24H After reset: FFH R/W

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

Address: FF2CH After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM12 | 1 | 1 | 1 | 1 | PM123 | PM122 | PM121 | PM120 |

| PMmn | Selection of I/O mode of Pmn pin (m = 0, 2, 3, 4, or 12; n = 0 to 7) |
|---|---|
| 0 | Output mode (output buffer ON) |
| 1 | Input mode (output buffer OFF) |

### (2) Port registers (P0, P2, P3, P4, P12, P13)

These registers are used to write data to be output from the corresponding port pin to an external device connected to the chip.

When a port register is read, the pin level is read in the input mode, and the value of the output latch of the port is read in the output mode.

P00 to P03, P20 to P23, P30 to P34, P40 to P47, P120 to P123, and P130 are set by using a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets these registers to 00H.

**Figure 4-16. Format of Port Register**

Address: FF00H After reset: 00H (Output latch) R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P0 | 0 | 0 | 0 | 0 | P03 | P02 | P01 | P00 |

Address: FF02H After reset: 00H (Output latch) R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P2 | 0 | 0 | 0 | 0 | P23 | P22 | P21 | P20 |

Address: FF03H After reset: 00H[Note] (Output latch) R/W[Note]

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P3 | 0 | 0 | 0 | P34 | P33 | P32 | P31 | P30 |

Address: FF04H After reset: 00H (Output latch) R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P4 | P47 | P46 | P45 | P44 | P43 | P42 | P41 | P40 |

Address: FF0CH After reset: 00H (Output latch) R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P12 | 0 | 0 | 0 | 0 | P123 | P122 | P121 | P120 |

Address: FF0DH After reset: 00H (Output latch) R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P130 |

| Pmn | m = 0, 2, 3, 4, 12, or 13; n = 0-7 | |
|---|---|---|
| | Controls of output data (in output mode) | Input data read (in input mode) |
| 0 | Output 0 | Input low level |
| 1 | Output 1 | Input high level |

**Note** Because P34 is read-only, its reset value is undefined.

### (3) Port mode control register 2 (PMC2)

This register specifies the port mode or A/D converter mode.

Each bit of the PMC2 register corresponds to each pin of port 2 and can be specified in 1-bit units.

PMC2 is set by using a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets PMC2 to 00H.

**Figure 4-17. Format of Port Mode Control Register 2**

Address: FF84H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMC2 | 0 | 0 | 0 | 0 | PMC23 | PMC22 | PMC21 | PMC20 |

| PMC2n | Specification of operation mode (n = 0 to 3) |
|---|---|
| 0 | Port mode |
| 1 | A/D converter mode |

**Table 4-3. Setting of Port Mode Register, Port Register (Output Latch), and Port Mode Control Register When Alternate Function Is Used**

| Port Name | Alternate-Function Pin | | PM×× | P×× | PMC2n (n = 0 to 3) |
|---|---|---|---|---|---|
| | Name | I/O | | | |
| P20 to P23 | ANI0 to ANI3 | Input | 1 | × | 1 |
| P30 | TI000 | Input | 1 | × | – |
| | INTP0 | Input | 1 | × | – |
| P31 | TO00 | Output | 0 | 0 | – |
| | TI010 | Input | 1 | × | – |
| | INTP2 | Input | 1 | × | – |
| P41 | INTP3 | Input | 1 | × | – |
| P42 | TOH1 | Output | 0 | 0 | – |
| P43 | TxD6 | Output | 0 | 1 | – |
| | INTP1 | Input | 1 | × | – |
| P44 | RxD6 | Input | 1 | × | – |

**Remark** ×: don’t care

PM××: Port mode register, P××: Port register (output latch of port)

PMC2×: Port mode control register

### (4) Pull-up resistor option registers (PU0, PU2, PU3, PU4, and PU12)

These registers are used to specify whether an on-chip pull-up resistor is connected to P00 to P03, P20 to P23, P30 to P33, P40 to P47, P120, and P123. By setting PU0, PU2, PU3, PU4, or PU12, an on-chip pull-up resistor can be connected to the port pin corresponding to the bit of PU0, PU2, PU3, PU4, or PU12.

PU0, PU2, PU3, PU4, and PU12 are set by using a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal set these registers to 00H.

**Figure 4-18. Format of Pull-up Resistor Option Register**

Address: FF30H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU0 | 0 | 0 | 0 | 0 | PU03 | PU02 | PU01 | PU00 |

Address: FF32H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU2 | 0 | 0 | 0 | 0 | PU23 | PU22 | PU21 | PU20 |

Address: FF33H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU3 | 0 | 0 | 0 | 0 | PU33 | P32 | PU31 | PU30 |

Address: FF34H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU4 | PU47 | PU46 | PU45 | PU44 | PU43 | PU42 | PU41 | PU40 |

Address: FF3CH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU12 | 0 | 0 | 0 | 0 | PU123 | 0 | 0 | PU120 |

| PUmn | Selection of connection of on-chip pull-up resistor of Pmn (m = 0, 2, 3, 4, or 12; n = 0 to 7) |
|---|---|
| 0 | Does not connect on-chip pull-up resistor |
| 1 | Connects on-chip pull-up resistor |

## 4.4 Operation of Port Function

The operation of a port differs, as follows, depending on the setting of the I/O mode.

**Caution Although a 1-bit memory manipulation instruction manipulates 1 bit, it accesses a port in 8-bit units. Therefore, the contents of the output latch of a pin in the input mode, even if it is not subject to manipulation by the instruction, are undefined in a port with a mixture of inputs and outputs.**

### 4.4.1 Writing to I/O port

**(1) In output mode**

A value can be written to the output latch by a transfer instruction. In addition, the contents of the output latch are output from the pin. Once data is written to the output latch, it is retained until new data is written to the output latch.

Reset signal generation clears the data in the output latch.

**(2) In input mode**

A value can be written to the output latch by a transfer instruction. Because the output buffer is off, however, the pin status remains unchanged.

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

Reset signal generation clears the data in the output latch.

### 4.4.2 Reading from I/O port

**(1) In output mode**

The contents of the output latch can be read by a transfer instruction. The contents of the output latch remain unchanged.

**(2) In input mode**

The pin status can be read by a transfer instruction. The contents of the output latch remain unchanged.

### 4.4.3 Operations on I/O port

**(1) In output mode**

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

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

Reset signal generation clears the data in the output latch.

**(2) In input mode**

The pin level is read and an operation is performed on its contents. The operation result is written to the output latch. However, the pin status remains unchanged because the output buffer is off.

Reset signal generation clears the data in the output latch.

# CHAPTER 5 CLOCK GENERATORS

## 5.1 Functions of Clock Generators

The clock generators include a circuit that generates a clock (system clock) to be supplied to the CPU and peripheral hardware, and a circuit that generates a clock (interval time generation clock) to be supplied to the watchdog timer and 8-bit timer H1 (TMH1).

### 5.1.1 System clock oscillators

The following three types of system clock oscillators are used.

- High-speed Ring-OSC oscillator
  This circuit internally oscillates a clock of 8 MHz (TYP.). Its oscillation can be stopped by execution of the STOP instruction.
  If the high-speed Ring-OSC oscillator is selected to supply the system clock, the X1 and X2 pins can be used as I/O port pins.

- Crystal/ceramic oscillator
  This circuit oscillates a clock with a crystal/ceramic oscillator connected across the X1 and X2 pins. It can oscillate a clock of 1 to 10 MHz. Oscillation of this circuit can be stopped by execution of the STOP instruction.

- External clock input circuit
  This circuit supplies a clock from an external IC to the X1 pin. A clock of 1 to 10 MHz can be supplied. Internal clock supply can be stopped by execution of the STOP instruction.
  If the external clock input is selected as the system clock, the X2 pin can be used as an I/O port pin.

The system clock source is selected by using the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**.
When using the X1 and X2 pins as I/O port pins, refer to **CHAPTER 4 PORT FUNCTIONS** for details.

### 5.1.2 Clock oscillator for interval time generation

The following circuit is used as a clock oscillator for interval time generation.

- Low-speed Ring-OSC oscillator
  This circuit oscillates a clock of 240 kHz (TYP.). Its oscillation can be stopped by using the low-speed Ring-OSC mode register (LSRCM) when it is specified by the option byte that its oscillation can be stopped by software.

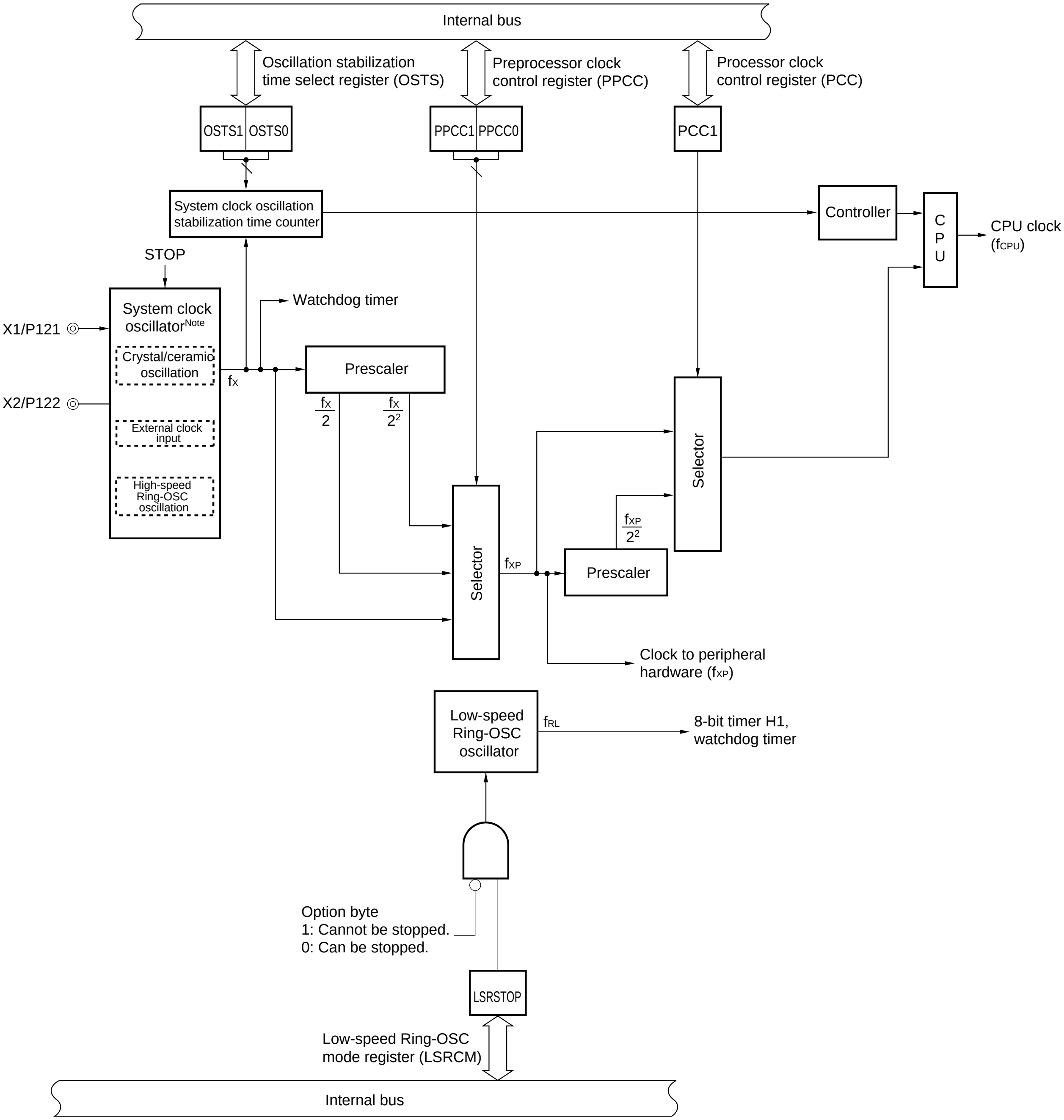
## 5.2 Configuration of Clock Generators

The clock generators consist of the following hardware.

**Table 5-1. Configuration of Clock Generators**

| Item | Configuration |
|---|---|
| Control registers | Processor clock control register (PCC)<br>Preprocessor clock control register (PPCC)<br>Low-speed Ring-OSC mode register (LSRCM)<br>Oscillation stabilization time select register (OSTS) |
| Oscillators | Crystal/ceramic oscillator<br>High-speed Ring-OSC oscillator<br>External clock input circuit<br>Low-speed Ring-OSC oscillator |

**Figure 5-1. Block Diagram of Clock Generators**


Internal bus
Oscillation stabilization time select register (OSTS)
Preprocessor clock control register (PPCC)
Processor clock control register (PCC)
OSTS1
OSTS0
PPCC1
PPCC0
PCC1
System clock oscillation stabilization time counter
Controller
C P U
CPU clock ($f_{CPU}$)
STOP
System clock oscillator[Note]
Crystal/ceramic oscillation
External clock input
High-speed Ring-OSC oscillation
X1/P121
X2/P122
Watchdog timer
$f_X$
Prescaler
$\frac{f_X}{2}$
$\frac{f_X}{2^2}$
Selector
$f_{XP}$
Prescaler
$\frac{f_{XP}}{2^2}$
Selector
Clock to peripheral hardware ($f_{XP}$)
Low-speed Ring-OSC oscillator
$f_{RL}$
8-bit timer H1, watchdog timer
Option byte
1: Cannot be stopped.
0: Can be stopped.
LSRSTOP
Low-speed Ring-OSC mode register (LSRCM)
Internal bus

**Note** Select the high-speed Ring-OSC oscillator, crystal/ceramic oscillator, or external clock input as the system clock source by using the option byte.

## 5.3 Registers Controlling Clock Generators

The clock generators are controlled by the following four registers.

- Processor clock control register (PCC)
- Preprocessor clock control register (PPCC)
- Low-speed Ring-OSC mode register (LSRCM)
- Oscillation stabilization time select register (OSTS)

### (1) Processor clock control register (PCC) and preprocessor clock control register (PPCC)

These registers are used to specify the division ratio of the system clock.

PCC and PPCC are set by using a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets PCC and PPCC to 02H.

**Figure 5-2. Format of Processor Clock Control Register (PCC)**

Address: FFFBH After reset: 02H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCC | 0 | 0 | 0 | 0 | 0 | 0 | PCC1 | 0 |

**Caution Bits 7 to 2, and 0 must be set to 0.**

**Figure 5-3. Format of Preprocessor Clock Control Register (PPCC)**

Address: FFF3H After reset: 02H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PPCC | 0 | 0 | 0 | 0 | 0 | 0 | PPCC1 | PPCC0 |

| PPCC1 | PPCC0 | PCC1 | Selection of CPU clock ($f_{CPU}$) |
|---|---|---|---|
| 0 | 0 | 0 | $f_X$ |
| 0 | 1 | 0 | $f_X/2$ **Note 1** |
| 0 | 0 | 1 | $f_X/2^2$ |
| 1 | 0 | 0 | $f_X/2^2$ **Note 2** |
| 0 | 1 | 1 | $f_X/2^3$ **Note 1** |
| 1 | 0 | 1 | $f_X/2^4$ **Note 2** |
| Other than above | | | Setting prohibited |

**Notes 1.** If PPCC = 01H, the clock ($f_{XP}$) supplied to the peripheral hardware is $f_X/2$.

**2.** If PPCC = 02H, the clock ($f_{XP}$) supplied to the peripheral hardware is $f_X/2^2$.

The fastest instruction of the 78K0S/KB1+ is executed in two CPU clocks. Therefore, the relationship between the CPU clock ($f_{CPU}$) and the minimum instruction execution time is as shown in Table 5-2.

**Table 5-2. Relationship Between CPU Clock and Minimum Instruction Execution Time**

| CPU Clock ($f_{CPU}$) Note | Minimum Instruction Execution Time: 2/$f_{CPU}$ | |
|---|---|---|
| | High-speed Ring-OSC clock (at 8.0 MHz (TYP.)) | Crystal/ceramic oscillation clock or external clock input (at 10.0 MHz) |
| $f_X$ | 0.25 $\mu$s | 0.2 $\mu$s |
| $f_X/2$ | 0.5 $\mu$s | 0.4 $\mu$s |
| $f_X/2^2$ | 1.0 $\mu$s | 0.8 $\mu$s |
| $f_X/2^3$ | 2.0 $\mu$s | 1.6 $\mu$s |
| $f_X/2^4$ | 4.0 $\mu$s | 3.2 $\mu$s |

**Note** The CPU clock (high-speed Ring-OSC clock, crystal/ceramic oscillation clock, or external clock input) is selected by the option byte.

### (2) Low-speed Ring-OSC mode register (LSRCM)

This register is used to select the operation mode of the low-speed Ring-OSC oscillator (240 kHz (TYP.)).

This register is valid when it is specified by the option byte that the low-speed Ring-OSC oscillator can be stopped by software. If it is specified by the option byte that the low-speed Ring-OSC oscillator cannot be stopped by software, setting of this register is invalid, and the low-speed Ring-OSC oscillator continues oscillating. In addition, the source clock of WDT is fixed to the low-speed Ring-OSC oscillator. For details, refer to **CHAPTER 9 WATCHDOG TIMER**.

LSRCM can be set by using a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets LSRCM to 00H.

**Figure 5-4. Format of Low-Speed Ring-OSC Mode Register (LSRCM)**

Address: FF58H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| LSRCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | LSRSTOP |

| LSRSTOP | Oscillation/stop of low-speed Ring-OSC |
|---|---|
| 0 | Low-speed Ring-OSC oscillates |
| 1 | Low-speed Ring-OSC stops |

### (3) Oscillation stabilization time select register (OSTS)

This register is used to select oscillation stabilization time of the clock supplied from the oscillator when the STOP mode is released. The wait time set by OSTS is valid only when the crystal/ceramic oscillation clock is selected as the system clock and after the STOP mode is released. If the high-speed Ring-OSC oscillator or external clock input is selected as the system clock source, no wait time elapses.

The system clock oscillator and the oscillation stabilization time that elapses after power application or release of reset are selected by the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**.

OSTS is set by using an 8-bit memory manipulation instruction.

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

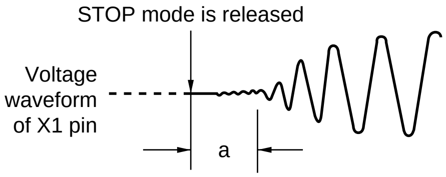
Address: FFF4H After reset: Undefined R/W

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

| OSTS1 | OSTS0 | Selection of oscillation stabilization time |
|---|---|---|
| 0 | 0 | $2^{10}$/fx (102.4 μs) |
| 0 | 1 | $2^{12}$/fx (409.6 μs) |
| 1 | 0 | $2^{15}$/fx (3.27 ms) |
| 1 | 1 | $2^{17}$/fx (13.1 ms) |

**Cautions 1. To set and then release the STOP mode, set the oscillation stabilization time as follows.**
**Expected oscillation stabilization time of resonator ≤ Oscillation stabilization time set by OSTS**

**2. The wait time after the STOP mode is released does not include the time from the release of the STOP mode to the start of clock oscillation ("a" in the figure below), regardless of whether STOP mode was released by reset signal generation or interrupt generation.**



**Caution 3. The oscillation stabilization time that elapses on power application or after release of reset is selected by the option byte. For details, refer to CHAPTER 18 OPTION BYTE.**

**Remarks 1.** ( ): fx = 10 MHz

**2.** Determine the oscillation stabilization time of the resonator by checking the characteristics of the resonator to be used.

## 5.4 System Clock Oscillators

The following three types of system clock oscillators are available.

| | |
|---|---|
| • High-speed Ring-OSC oscillator: | Internally oscillates a clock of 8 MHz (TYP.). |
| • Crystal/ceramic oscillator: | Oscillates a clock of 1 to 10 MHz. |
| • External clock input circuit: | Supplies a clock of 1 to 10 MHz to the X1 pin. |

### 5.4.1 High-speed Ring-OSC oscillator

The 78K0S/KB1+ includes a high-speed Ring-OSC oscillator (8 MHz (TYP.)).

If the high-speed Ring-OSC is selected by the option byte as the clock source, the X1 and X2 pins can be used as I/O port pins.

For details of the option byte, refer to **CHAPTER 18 OPTION BYTE**. For details of I/O ports, refer to **CHAPTER 4 PORT FUNCTIONS**.

### 5.4.2 Crystal/ceramic oscillator

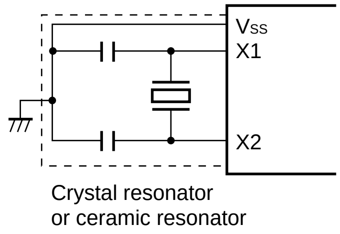
The crystal/ceramic oscillator oscillates using a crystal or ceramic resonator connected between the X1 and X2 pins.

If the crystal/ceramic oscillator is selected by the option byte as the system clock source, the X1 and X2 pins are used as crystal or ceramic resonator connection pins.

For details of the option byte, refer to **CHAPTER 18 OPTION BYTE**. For details of I/O ports, refer to **CHAPTER 4 PORT FUNCTIONS**.

Figure 5-6 shows the external circuit of the crystal/ceramic oscillator.

**Figure 5-6. External Circuit of Crystal/Ceramic Oscillator**

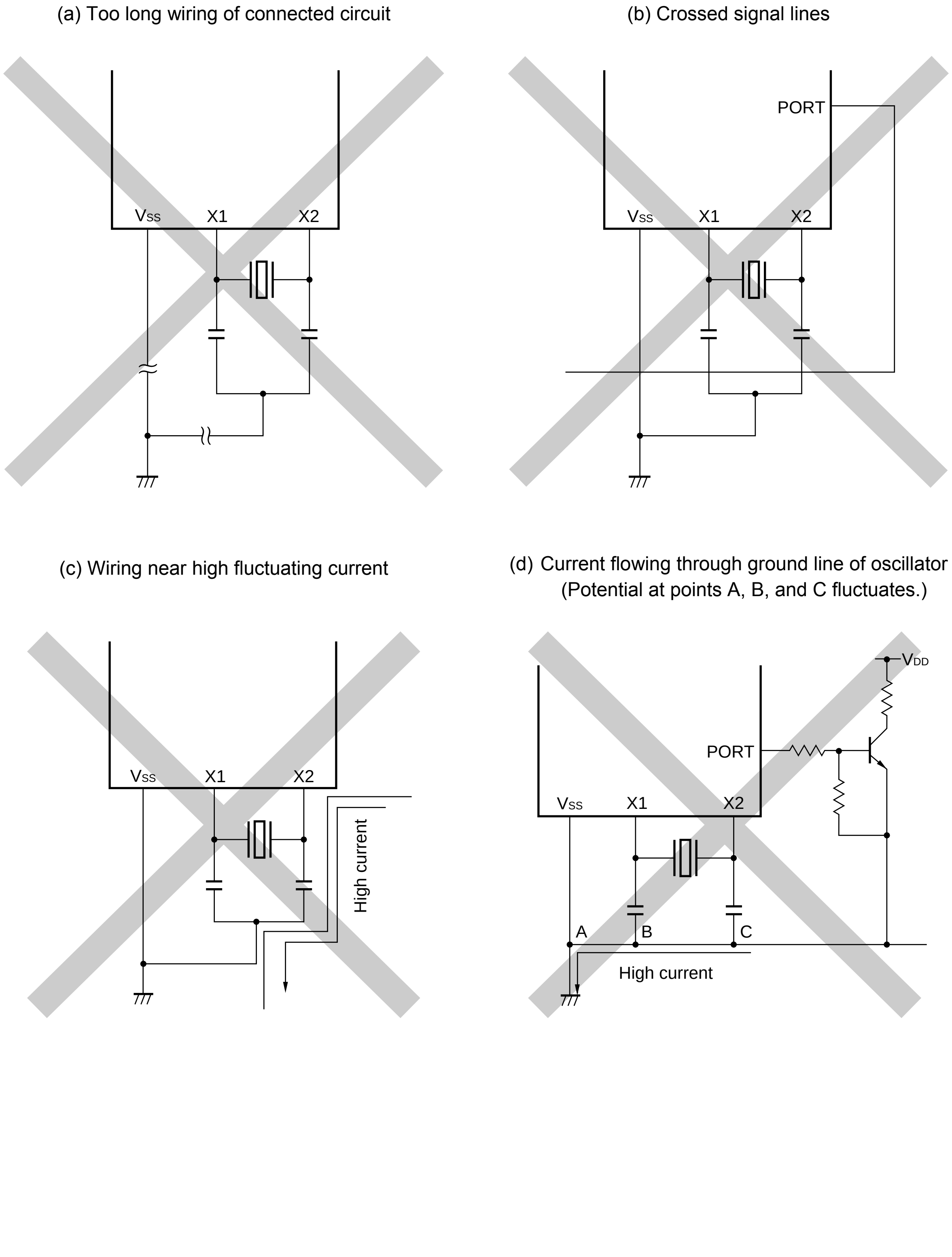


**Caution When using the crystal/ceramic oscillator, wire as follows in the area enclosed by the broken lines in Figure 5-6 to avoid an adverse effect from wiring capacitance.**

- **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.**

Figure 5-7 shows examples of incorrect resonator connection.

**Figure 5-7. Examples of Incorrect Resonator Connection (1/2)**



(a) Too long wiring of connected circuit

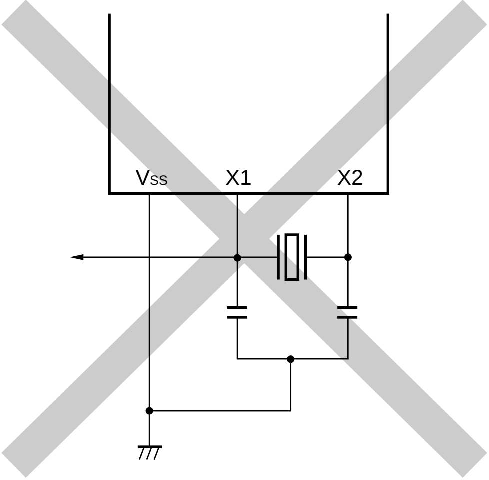
(b) Crossed signal lines

(c) Wiring near high fluctuating current

(d) Current flowing through ground line of oscillator
(Potential at points A, B, and C fluctuates.)

Figure 5-7. Examples of Incorrect Resonator Connection (2/2)

(e) Signals are fetched



### 5.4.3 External clock input circuit

This circuit supplies a clock from an external IC to the X1 pin.

If external clock input is selected by the option byte as the system clock source, the X2 pin can be used as an I/O port pin.

For details of the option byte, refer to **CHAPTER 18 OPTION BYTE**. For details of I/O ports, refer to **CHAPTER 4 PORT FUNCTIONS**.

### 5.4.4 Prescaler

The prescaler divides the clock ($f_X$) output by the system clock oscillator to generate a clock ($f_{XP}$) to be supplied to the peripheral hardware. It also divides the clock to peripheral hardware ($f_{XP}$) to generate a clock to be supplied to the CPU.

**Remark** The clock output by the oscillator selected by the option byte (high-speed Ring-OSC oscillator, crystal/ceramic oscillator, or external clock input circuit) is divided. For details of the option byte, refer to **CHAPTER 18 OPTION BYTE**.

## 5.5 Operation of CPU Clock Generator

A clock ($f_{CPU}$) is supplied to the CPU from the system clock ($f_X$) oscillated by one of the following three types of oscillators.

| | |
|---|---|
| • High-speed Ring-OSC oscillator: | Internally oscillates a clock of 8 MHz (TYP.). |
| • Crystal/ceramic oscillator: | Oscillates a clock of 1 to 10 MHz. |
| • External clock input circuit: | Supplies a clock of 1 to 10 MHz to X1 pin. |

The system clock oscillator is selected by the option byte. For details of the option byte, refer to **CHAPTER 18 OPTION BYTE**.

### (1) High-speed Ring-OSC oscillator

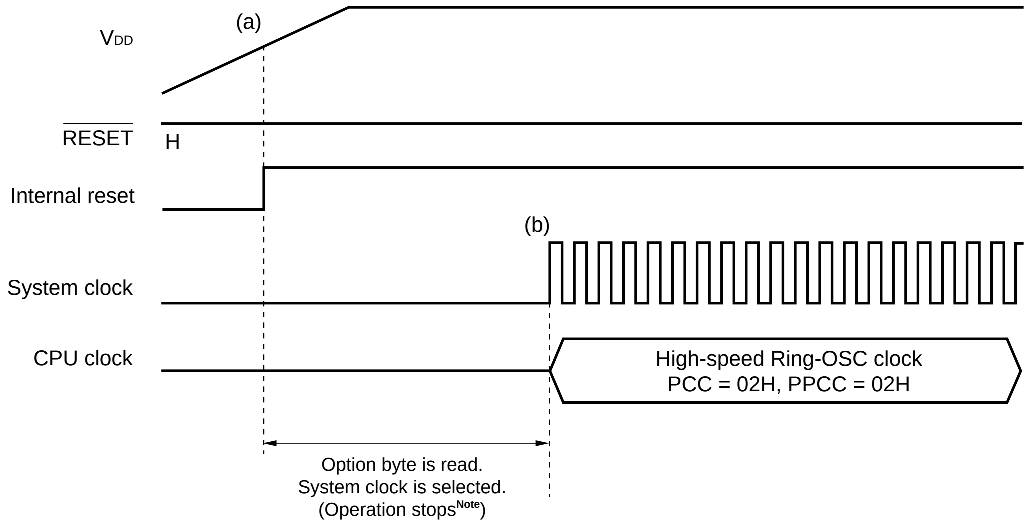
When the high-speed Ring-OSC oscillator is selected by the option byte, the following is possible.

- Shortening of start time
  If the high-speed Ring-OSC oscillator is selected as the oscillator, the CPU can be started without having to wait for the oscillation stabilization time of the system clock. Therefore, the start time can be shortened.
- Improvement of expandability
  If the high-speed Ring-OSC oscillator is selected as the oscillator, the X1 and X2 pins can be used as I/O port pins. For details, refer to **CHAPTER 4 PORT FUNCTIONS**.

Figures 5-8 and 5-9 show the timing chart and status transition diagram of the default start by the high-speed Ring-OSC oscillator.

**Remark** When the high-speed Ring-OSC oscillator is used, the clock accuracy is ±5%.

**Figure 5-8. Timing Chart of Default Start by High-Speed Ring-OSC Oscillator**

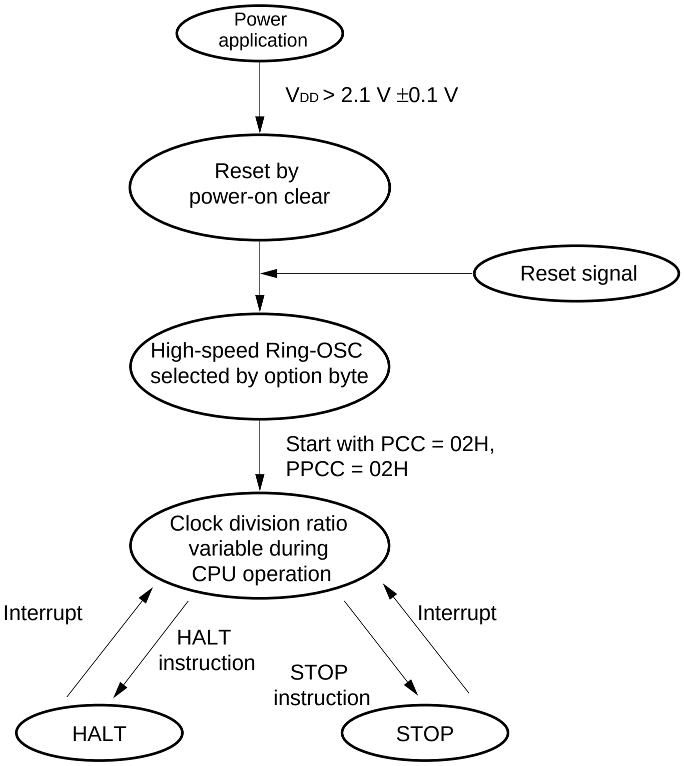


**Note** Operation stop time is 277 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.075 ms (MAX.).

(a) The internal reset signal is generated by the power-on clear function on power application, the option byte is referenced after reset, and the system clock is selected.

(b) The option byte is referenced and the system clock is selected. Then the high-speed Ring-OSC clock operates as the system clock.

**Figure 5-9. Status Transition of Default Start by High-Speed Ring-OSC**



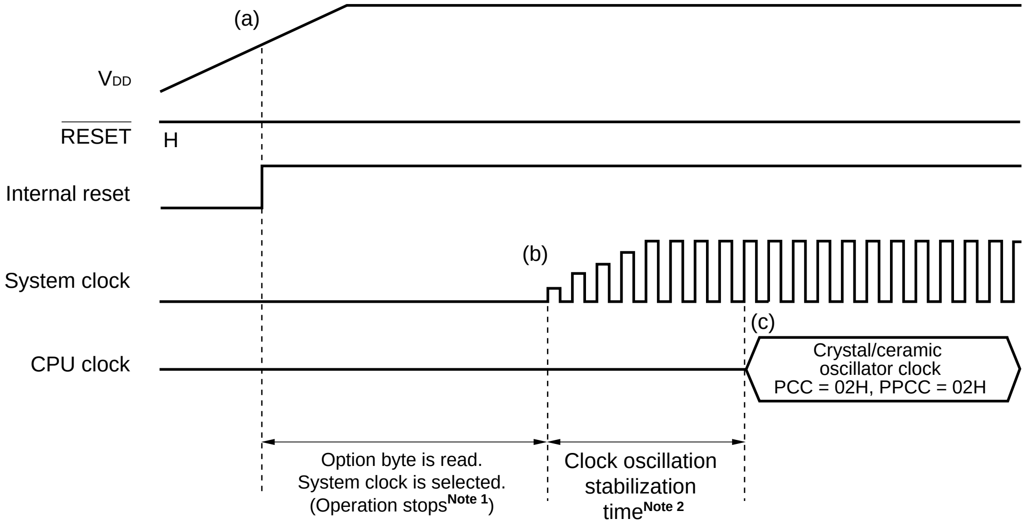
**Remark** PCC: Processor clock control register
PPCC: Preprocessor clock control register

**(2) Crystal/ceramic oscillator**

If crystal/ceramic oscillation is selected by the option byte, a clock frequency of 1 to 10 MHz can be selected and the accuracy of processing is improved because the frequency deviation is small, as compared with high-speed Ring-OSC oscillation (8 MHz (TYP.)).

Figures 5-10 and 5-11 show the timing chart and status transition diagram of default start by the crystal/ceramic oscillator.

Figure 5-10. Timing Chart of Default Start by Crystal/Ceramic Oscillator


(a)
VDD
RESET
H
Internal reset
(b)
System clock
(c)
CPU clock
Crystal/ceramic oscillator clock PCC = 02H, PPCC = 02H
Option byte is read. System clock is selected. (Operation stopsNote 1)
Clock oscillation stabilization timeNote 2

**Notes 1.** Operation stop time is 276 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.074 ms (MAX.).

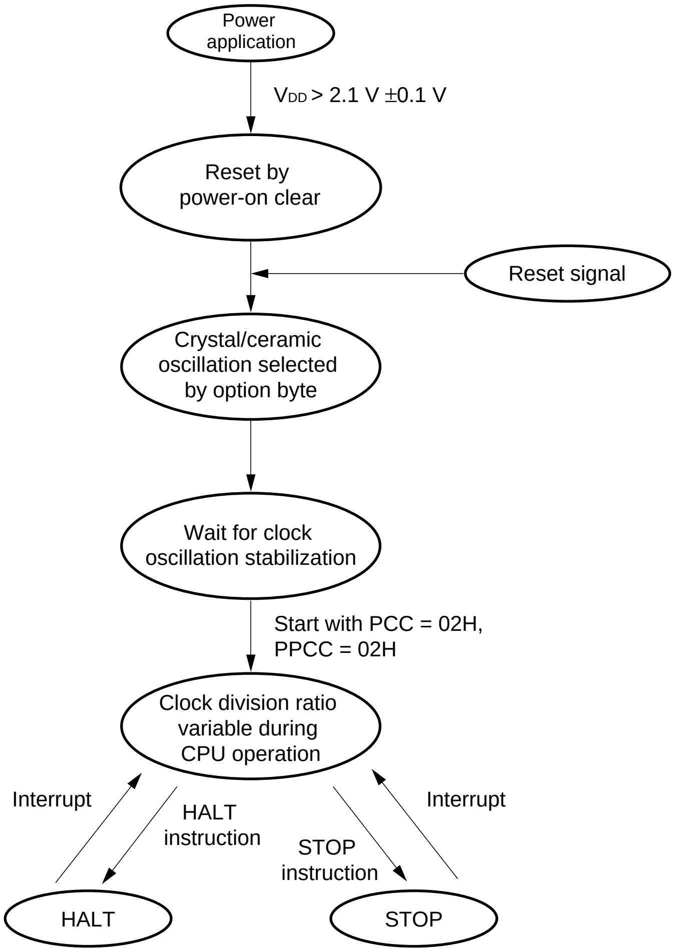
**2.** The clock oscillation stabilization time for default start is selected by the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**. The oscillation stabilization time that elapses after the STOP mode is released is selected by the oscillation stabilization time select register (OSTS).

(a) The internal reset signal is generated by the power-on clear function on power application, the option byte is referenced after reset, and the system clock is selected.

(b) After high-speed Ring-OSC clock is generated, the option byte is referenced and the system clock is selected. In this case, the crystal/ceramic oscillator clock is selected as the system clock.

(c) If the system clock is the crystal/ceramic oscillator clock, it starts operating as the CPU clock after clock oscillation is stabilized. The wait time is selected by the option byte. For details, refer to **CHAPTER 18 OPTION BYTE**.

**Figure 5-11. Status Transition of Default Start by Crystal/Ceramic Oscillation**


Power application
VDD > 2.1 V ±0.1 V
Reset by power-on clear
Reset signal
Crystal/ceramic oscillation selected by option byte
Wait for clock oscillation stabilization
Start with PCC = 02H, PPCC = 02H
Clock division ratio variable during CPU operation
Interrupt
HALT instruction
STOP instruction
Interrupt
HALT
STOP

**Remark** PCC: Processor clock control register
PPCC: Preprocessor clock control register

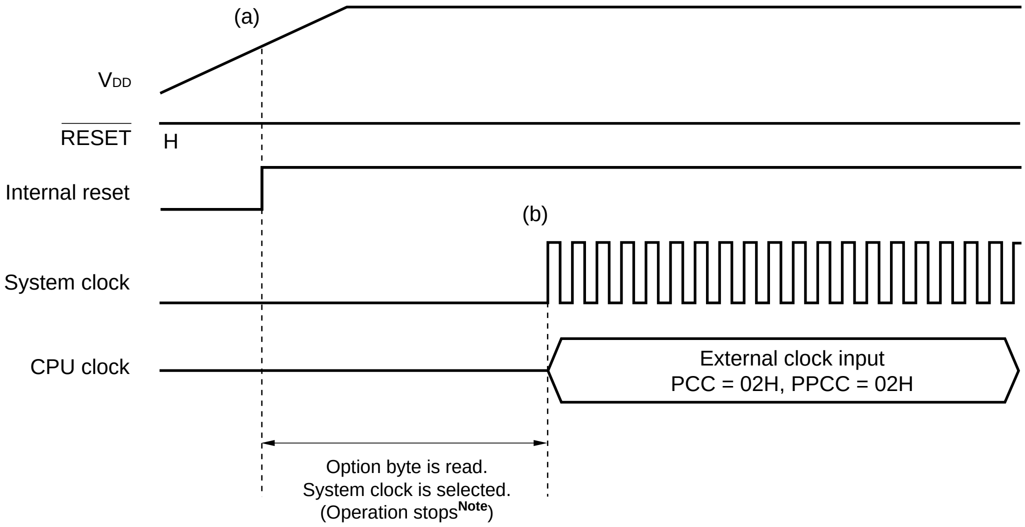
### (3) External clock input circuit

If external clock input is selected by the option byte, the following is possible.

- High-speed operation
  The accuracy of processing is improved as compared with high-speed Ring-OSC oscillation (8 MHz (TYP.)) because an oscillation frequency of 1 to 10 MHz can be selected and an external clock with a small frequency deviation can be supplied.
- Improvement of expandability
  If the external clock input circuit is selected as the oscillator, the X2 pin can be used as an I/O port pin. For details, refer to **CHAPTER 4 PORT FUNCTIONS**.

Figures 5-12 and 5-13 show the timing chart and status transition diagram of default start by external clock input.

**Figure 5-12. Timing of Default Start by External Clock Input**

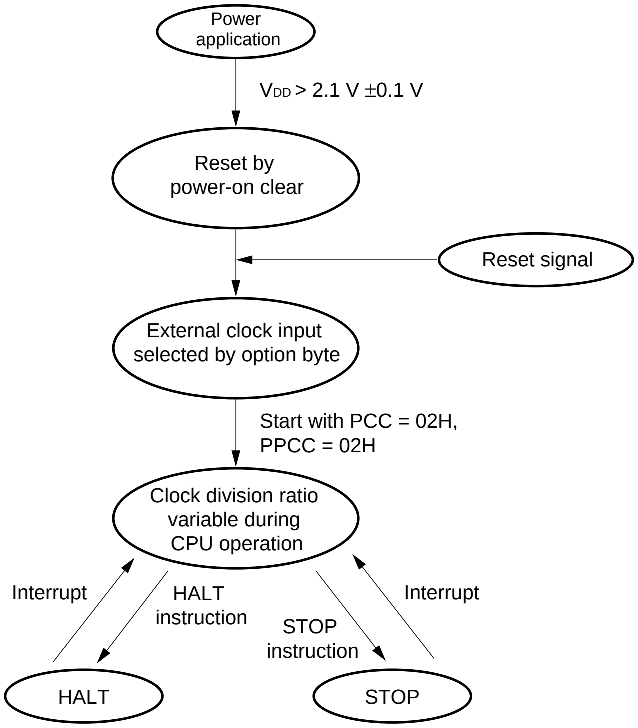


**Note** Operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

(a) The internal reset signal is generated by the power-on clear function on power application, the option byte is referenced after reset, and the system clock is selected.

(b) The option byte is referenced and the system clock is selected. Then the external clock operates as the system clock.

**Figure 5-13. Status Transition of Default Start by External Clock Input**



**Remark** PCC: Processor clock control register
PPCC: Preprocessor clock control register

## 5.6 Operation of Clock Generator Supplying Clock to Peripheral Hardware

The following two types of clocks are supplied to the peripheral hardware.

- Clock to peripheral hardware ($f_{XP}$)
- Low-speed Ring-OSC clock ($f_{RL}$)

### (1) Clock to peripheral hardware

The clock to the peripheral hardware is supplied by dividing the system clock ($f_X$). The division ratio is selected by the pre-processor clock control register (PPCC).

Three types of frequencies are selectable: “$f_X$”, “$f_X/2$”, and “$f_X/2^2$”. Table 5-3 lists the clocks supplied to the peripheral hardware.

**Table 5-3. Clocks to Peripheral Hardware**

| PPCC1 | PPCC0 | Selection of clock to peripheral hardware ($f_{XP}$) |
|---|---|---|
| 0 | 0 | $f_X$ |
| 0 | 1 | $f_X/2$ |
| 1 | 0 | $f_X/2^2$ |
| 1 | 1 | Setting prohibited |

### (2) Low-speed Ring-OSC clock

The low-speed Ring-OSC oscillator of the clock oscillator for interval time generation is always started after release of reset, and oscillates at 240 kHz (TYP.).

It can be specified by the option byte whether the low-speed Ring-OSC oscillator can or cannot be stopped by software. If it is specified that the low-speed Ring-OSC oscillator can be stopped by software, oscillation can be started or stopped by using the low-speed Ring-OSC mode register (LSRCM). If it is specified that it cannot be stopped by software, the clock source of WDT is fixed to the low-speed Ring-OSC clock ($f_{RL}$).

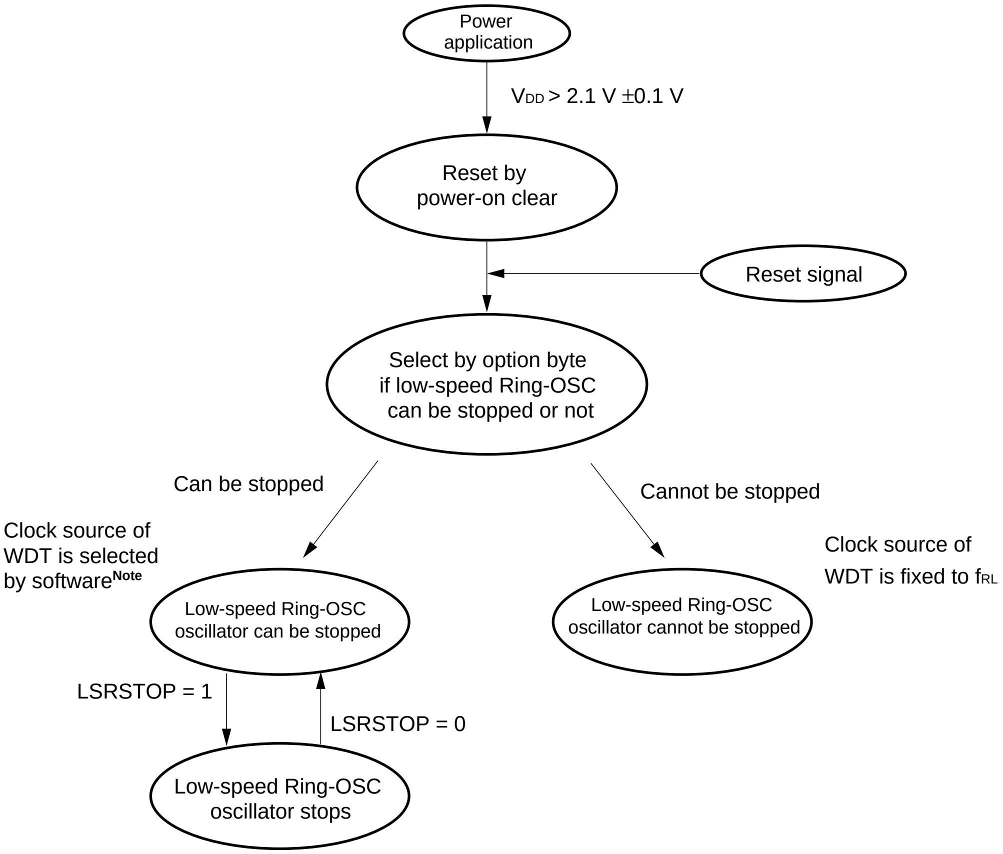
The low-speed Ring-OSC oscillator is independent of the CPU clock. If it is used as the source clock of WDT, therefore, a hang-up can be detected even if the CPU clock is stopped. If the low-speed Ring-OSC oscillator is used as a count clock source of 8-bit timer H1, 8-bit timer H1 can operate even in the standby status.

Table 5-4 shows the operation status of the low-speed Ring-OSC oscillator when it is selected as the source clock of WDT and the count clock of 8-bit timer H1. Figure 5-14 shows the status transition of the low-speed Ring-OSC oscillator.

**Table 5-4. Operation Status of Low-Speed Ring-OSC Oscillator**

| Option Byte Setting | | CPU Status | WDT Status | TMH1 Status |
|---|---|---|---|---|
| Can be stopped by software | LSRSTOP = 1 | Operation mode | Stopped | Stopped |
| | LSRSTOP = 0 | | Operates | Operates |
| | LSRSTOP = 1 | Standby | Stopped | Stopped |
| | LSRSTOP = 0 | | Stopped | Operates |
| Cannot be stopped | | Operation mode | Operates | |
| | | Standby | | |

**Figure 5-14. Status Transition of Low-Speed Ring-OSC Oscillator**


Power application
VDD > 2.1 V ±0.1 V
Reset by power-on clear
Reset signal
Select by option byte if low-speed Ring-OSC can be stopped or not
Can be stopped
Cannot be stopped
Clock source of WDT is selected by softwareNote
Clock source of WDT is fixed to fRL
Low-speed Ring-OSC oscillator can be stopped
Low-speed Ring-OSC oscillator cannot be stopped
LSRSTOP = 1
LSRSTOP = 0
Low-speed Ring-OSC oscillator stops

**Note** The clock source of the watchdog timer (WDT) is selected from $f_X$ or $f_{RL}$, or it may be stopped. For details, refer to **CHAPTER 9 WATCHDOG TIMER**.

## 6.1 Functions of 16-Bit Timer/Event Counter 00

16-bit timer/event counter 00 has the following functions.

**(1) Interval timer**

16-bit timer/event counter 00 generates interrupt requests at the preset time interval.

- Number of counts: 2 to 65536

**(2) External event counter**

16-bit timer/event counter 00 can measure the number of pulses with a high-/low-level width of a signal input externally.

- Valid level pulse width: 16/$f_{XP}$ or more

**(3) Pulse width measurement**

16-bit timer/event counter 00 can measure the pulse width of an externally input signal.

- Valid level pulse width: 2/$f_{XP}$ or more

**(4) Square-wave output**

16-bit timer/event counter 00 can output a square wave with any selected frequency.

- Cycle: (2 × 2 to 65536 × 2) × count clock cycle

**(5) PPG output**

16-bit timer/event counter 00 can output a square wave that have arbitrary cycle and pulse width.

- 1 < Pulse width < Cycle ≤ (FFFF + 1) H

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

16-bit timer/event counter 00 can output a one-shot pulse for which output pulse width can be set to any desired value.

## 6.2 Configuration of 16-Bit Timer/Event Counter 00

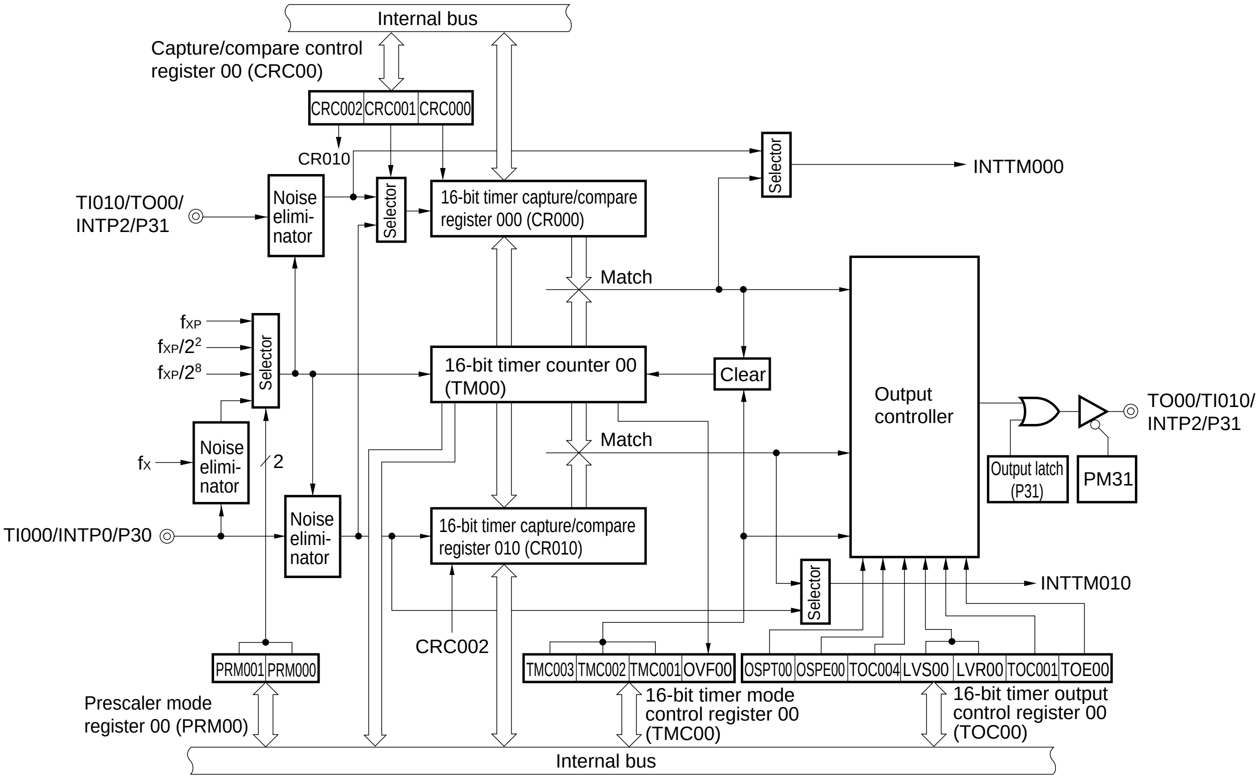
16-bit timer/event counter 00 consists of the following hardware.

Table 6-1. Configuration of 16-Bit Timer/Event Counter 00

| Item | Configuration |
|---|---|
| Timer counter | 16-bit timer counter 00 (TM00) |
| Register | 16-bit timer capture/compare registers 000, 010 (CR000, CR010) |
| Timer input | TI000, TI010 |
| Timer output | TO00, output controller |
| Control registers | 16-bit timer mode control register 00 (TMC00)<br>Capture/compare control register 00 (CRC00)<br>16-bit timer output control register 00 (TOC00)<br>Prescaler mode register 00 (PRM00)<br>Port mode register 3 (PM3)<br>Port register 3 (P3) |

Figures 6-1 shows a block diagram of these counters.

Figure 6-1. Block Diagram of 16-Bit Timer/Event Counter 00



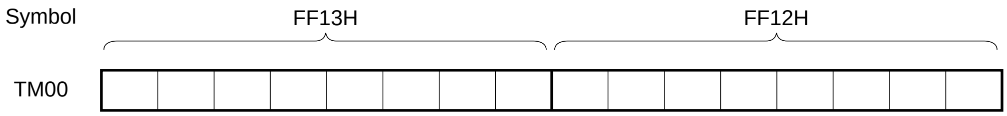
**(1) 16-bit timer counter 00 (TM00)**

TM00 is a 16-bit read-only register that counts count pulses.

The counter is incremented in synchronization with the rising edge of the count clock. If the count value is read during operation, input of the count clock is temporarily stopped, and the count value at that point is read.

**Figure 6-2. Format of 16-Bit Timer Counter 00 (TM00)**

Address: FF12H, FF13H After reset: 0000H R



The count value is reset to 0000H in the following cases.

<1> At reset signal generation
<2> If TMC003 and TMC002 are cleared
<3> If the valid edge of TI000 is input in the clear & start mode entered by inputting the valid edge of TI000
<4> If TM00 and CR000 match in the clear & start mode entered on a match between TM00 and CR000
<5> If OSPT00 is set to 1 in the one-shot pulse output mode

**Cautions 1. Even if TM00 is read, the value is not captured by CR010.**
**2. During TM00 is read, the count clock is stopped.**

**(2) 16-bit timer capture/compare register 000 (CR000)**

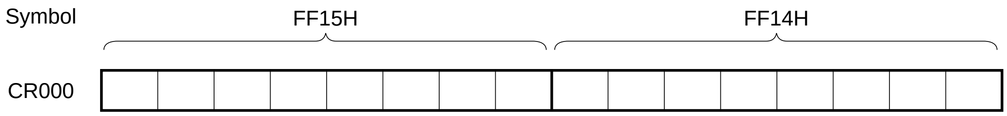
CR000 is a 16-bit register which has the functions of both a capture register and a compare register. Whether it is used as a capture register or as a compare register is set by bit 0 (CRC000) of capture/compare control register 00 (CRC00).

CR000 is set by 16-bit memory manipulation instruction.

Reset signal generation clears this register to 0000H.

**Figure 6-3. Format of 16-Bit Timer Capture/Compare Register 000 (CR000)**

Address: FF14H, FF15H After reset: 0000H R/W



- **When CR000 is used as a compare register**

  The value set in CR000 is constantly compared with the 16-bit timer/counter 00 (TM00) count value, and an interrupt request (INTTM000) is generated if they match. It can also be used as the register that holds the interval time then TM00 is set to interval timer operation.

- **When CR000 is used as a capture register**

  It is possible to select the valid edge of the TI000 pin or the TI010 pin as the capture trigger. Setting of the TI000 or TI010 valid edge is performed by means of prescaler mode register 00 (PRM00) (refer to **Table 6-2**).

### Table 6-2. CR000 Capture Trigger and Valid Edges of TI000 and TI010 Pins

**(1) TI000 pin valid edge selected as capture trigger (CRC001 = 1, CRC000 = 1)**

| CR000 Capture Trigger | TI000 Pin Valid Edge | | |
|---|---|---|---|
| | | ES010 | ES000 |
| Falling edge | Rising edge | 0 | 1 |
| Rising edge | Falling edge | 0 | 0 |
| No capture operation | Both rising and falling edges | 1 | 1 |

**(2) TI010 pin valid edge selected as capture trigger (CRC001 = 0, CRC000 = 1)**

| CR000 Capture Trigger | TI010 Pin Valid Edge | | |
|---|---|---|---|
| | | ES110 | ES100 |
| 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.** Setting ES010, ES000 = 1, 0 and ES110, ES100 = 1, 0 is prohibited.

**2.** ES010, ES000: Bits 5 and 4 of prescaler mode register 00 (PRM00)
ES110, ES100: Bits 7 and 6 of prescaler mode register 00 (PRM00)
CRC001, CRC000: Bits 1 and 0 of capture/compare control register 00 (CRC00)

**Cautions 1. Set CR000 to other than 0000H in the clear & start mode entered on match between TM00 and CR000. This means a 1-pulse count operation cannot be performed when this register is used as an external event counter. However, in the free-running mode and in the clear & start mode using the valid edge of TI000 pin, if CR000 is set to 0000H, an interrupt request (INTTM000) is generated when CR000 changes from 0000H to 0001H following overflow (FFFFH).**

**2. If the new value of CR000 is less than the value of 16-bit timer counter 0 (TM00), TM00 continues counting, overflows, and then starts counting from 0 again. If the new value of CR000 is less than the old value, therefore, the timer must be reset to be restarted after the value of CR000 is changed.**

**3. The value of CR000 after 16-bit timer/event counter 00 has stopped is not guaranteed.**

**4. The capture operation may not be performed for CR000 set in compare mode even if a capture trigger is input.**

**5. When P31 is used as the input pin for the valid edge of TI010, it cannot be used as a timer output (TO00). Moreover, when P31 is used as TO00, it cannot be used as the input pin for the valid edge of TI010.**

**6. If the register read period and the input of the capture trigger conflict when CR000 is used as a capture register, the capture trigger input takes precedence and the read data is undefined. Also, if the count stop of the timer and the input of the capture trigger conflict, the capture trigger is undefined.**

**7. Changing the CR000 setting may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation.**

### (3) 16-bit timer capture/compare register 010 (CR010)

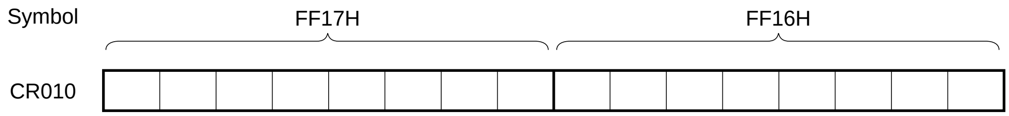
CR010 is a 16-bit register which has the functions of both a capture register and a compare register. Whether it is used as a capture register or a compare register is set by bit 2 (CRC002) of capture/compare control register 00 (CRC00).

CR010 is set by 16-bit memory manipulation instruction.

Reset signal generation clears CR010 to 0000H.

**Figure 6-4. Format of 16-Bit Timer Capture/Compare Register 010 (CR010)**

Address: FF16H, FF17H After reset: 0000H R/W



- **When CR010 is used as a compare register**
  The value set in CR010 is constantly compared with the 16-bit timer counter 00 (TM00) count value, and an interrupt request (INTTM010) is generated if they match.

- **When CR010 is used as a capture register**
  It is possible to select the valid edge of the TI000 pin as the capture trigger. The TI000 valid edge is set by means of prescaler mode register 00 (PRM00) (refer to **Table 6-3**).

**Table 6-3. CR010 Capture Trigger and Valid Edge of TI000 Pin (CRC002 = 1)**

| CR010 Capture Trigger | TI000 Pin Valid Edge | | |
|---|---|---|---|
| | | ES010 | ES000 |
| 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.** Setting ES010, ES000 = 1, 0 is prohibited.

**2.** ES010, ES000: Bits 5 and 4 of prescaler mode register 00 (PRM00)
CRC002: Bit 2 of capture/compare control register 00 (CRC00)

**Cautions 1. In the free-running mode and in the clear & start mode using the valid edge of the TI000 pin, if CR010 is set to 0000H, an interrupt request (INTTM010) is generated when CR010 changes from 0000H to 0001H following overflow (FFFFH).**

**2. If the new value of CR010 is less than the value of 16-bit timer counter 0 (TM00), TM00 continues counting, overflows, and then starts counting from 0 again. If the new value of CR010 is less than the old value, therefore, the timer must be reset to be restarted after the value of CR010 is changed.**

**3. The value of CR010 after 16-bit timer/event counter 00 has stopped is not guaranteed.**

**4. The capture operation may not be performed for CR010 set in compare mode even if a capture trigger is input.**

**5. If the register read period and the input of the capture trigger conflict when CR010 is used as a capture register, the capture trigger input takes precedence and the read data is undefined. Also, if the timer count stop and the input of the capture trigger conflict, the capture data is undefined.**

**6. Changing the CR010 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation.**

## 6.3 Registers to Control 16-Bit Timer/Event Counter 00

The following six types of registers are used to control 16-bit timer/event counter 00.

- 16-bit timer mode control register 00 (TMC00)
- Capture/compare control register 00 (CRC00)
- 16-bit timer output control register 00 (TOC00)
- Prescaler mode register 00 (PRM00)
- Port mode register 3 (PM3)
- Port register 3 (P3)

### (1) 16-bit timer mode control register 00 (TMC00)

This register sets the 16-bit timer operating mode, the 16-bit timer counter 00 (TM00) clear mode, and output timing, and detects an overflow.

TMC00 is set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets the value of TMC00 to 00H.

**Caution 16-bit timer counter 00 (TM00) starts operation at the moment TMC002 and TMC003 (operation stop mode) are set to a value other than 0, 0, respectively. Set TMC002 and TMC003 to 0, 0 to stop the operation.**

**Figure 6-5. Format of 16-Bit Timer Mode Control Register 00 (TMC00)**

Address: FF60H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | TMC003 | TMC002 | TMC001 | OVF00 |

| TMC003 | TMC002 | TMC001 | Operating mode and clear mode selection | TO00 inversion timing selection | Interrupt request generation |
|---|---|---|---|---|---|
| 0 | 0 | 0 | Operation stop<br>(TM00 cleared to 0) | No change | Not generated |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | Free-running mode | Match between TM00 and CR000 or match between TM00 and CR010 | <When operating as compare register><br>Generated on match between TM00 and CR000, or match between TM00 and CR010<br><When operating as capture register><br>Generated on TI000 pin and TI010 pin valid edge |
| 0 | 1 | 1 | | Match between TM00 and CR000, match between TM00 and CR010 or TI000 pin valid edge | |
| 1 | 0 | 0 | Clear & start occurs on valid edge of TI000 pin | – | |
| 1 | 0 | 1 | | | |
| 1 | 1 | 0 | Clear & start occurs on match between TM00 and CR000 | Match between TM00 and CR000 or match between TM00 and CR010 | |
| 1 | 1 | 1 | | Match between TM00 and CR000, match between TM00 and CR010 or TI000 pin valid edge | |

| OVF00 | Overflow detection of 16-bit timer counter 00 (TM00) |
|---|---|
| 0 | Overflow not detected |
| 1 | Overflow detected |

**Cautions**
1. **The timer operation must be stopped before writing to bits other than the OVF00 flag.**
2. **Regardless of the CPU's operation mode, when the timer stops, the signals input to pins TI000/TI010 are not acknowledged.**
3. **Except when TI000 pin valid edge is selected as the count clock, stop the timer operation before setting STOP mode or system clock stop mode; otherwise the timer may malfunction when the system clock starts.**
4. **Set the valid edge of the TI000 pin with bits 4 and 5 of prescaler mode register 00 (PRM00) after stopping the timer operation.**
5. **If the clear & start mode entered on a match between TM00 and CR000, clear & start mode at the valid edge of the TI000 pin, or free-running mode is selected, when the set value of CR000 is FFFFH and the TM00 value changes from FFFFH to 0000H, the OVF00 flag is set to 1.**
6. **Even if the OVF00 flag is cleared before the next count clock is counted (before TM00 becomes 0001H) after the occurrence of a TM00 overflow, the OVF00 flag is re-set newly and clear is disabled.**
7. **The capture operation is performed at the fall of the count clock. An interrupt request input (INTTM0n0), however, occurs at the rise of the next count clock.**

**Remark** TM00: 16-bit timer counter 00
CR000: 16-bit timer capture/compare register 000
CR010: 16-bit timer capture/compare register 010

### (2) Capture/compare control register 00 (CRC00)

This register controls the operation of the 16-bit capture/compare registers (CR000, CR010).
CRC00 is set by a 1-bit or 8-bit memory manipulation instruction.
Generation of reset signal sets the value of CRC00 to 00H.

**Figure 6-6. Format of Capture/Compare Control Register 00 (CRC00)**

Address: FF62H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | CRC002 | CRC001 | CRC000 |

| CRC002 | CR010 operating mode selection |
|---|---|
| 0 | Operate as compare register |
| 1 | Operate as capture register |

| CRC001 | CR000 capture trigger selection |
|---|---|
| 0 | Capture on valid edge of TI010 pin |
| 1 | Capture on valid edge of TI000 pin by reverse phase[Note] |

| CRC000 | CR000 operating mode selection |
|---|---|
| 0 | Operate as compare register |
| 1 | Operate as capture register |

**Note** If both the rising and falling edges have been selected as the valid edges of the TI000 pin, capture is not performed.

**Cautions 1. The timer operation must be stopped before setting CRC00.**

**2. When the clear & start mode entered on a match between TM00 and CR000 is selected by 16-bit timer mode control register 00 (TMC00), CR000 should not be specified as a capture register.**

**3. To ensure the reliability of the capture operation, the capture trigger requires a pulse longer than two cycles of the count clock selected by prescaler mode register 00 (PRM00) (refer to Figure 6-17).**

### (3) 16-bit timer output control register 00 (TOC00)

This register controls the operation of the 16-bit timer/event counter output controller. It sets timer output F/F set/reset, output inversion enable/disable, 16-bit timer/event counter 00 timer output enable/disable, one-shot pulse output operation enable/disable, and output trigger of one-shot pulse by software.

TOC00 is set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets the value of TOC00 to 00H.

**Figure 6-7. Format of 16-Bit Timer Output Control Register 00 (TOC00)**

Address: FF63H After reset: 00H R/W

| Symbol | 7 | <6> | <5> | 4 | <3> | <2> | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TOC00 | 0 | OSPT00 | OSPE00 | TOC004 | LVS00 | LVR00 | TOC001 | TOE00 |

| OSPT00 | One-shot pulse output trigger control via software |
|---|---|
| 0 | No one-shot pulse output trigger |
| 1 | One-shot pulse output trigger |

| OSPE00 | One-shot pulse output operation control |
|---|---|
| 0 | Successive pulse output mode |
| 1 | One-shot pulse output mode[Note] |

| TOC004 | Timer output F/F control using match of CR010 and TM00 |
|---|---|
| 0 | Disables inversion operation |
| 1 | Enables inversion operation |

| LVS00 | LVR00 | Timer output F/F status setting |
|---|---|---|
| 0 | 0 | No change |
| 0 | 1 | Timer output F/F reset (0) |
| 1 | 0 | Timer output F/F set (1) |
| 1 | 1 | Setting prohibited |

| TOC001 | Timer output F/F control using match of CR000 and TM00 |
|---|---|
| 0 | Disables inversion operation |
| 1 | Enables inversion operation |

| TOE00 | Timer output control |
|---|---|
| 0 | Disables output (output fixed to level 0) |
| 1 | Enables output |

**Note** The one-shot pulse output mode operates correctly only in the free-running mode and the mode in which clear & start occurs at the TI000 pin valid edge. In the mode in which clear & start occurs on a match between TM00 and CR000, one-shot pulse output is not possible because an overflow does not occur.

**Cautions**

1. **Timer operation must be stopped before setting other than OSPT00.**
2. **If LVS00 and LVR00 are read, 0 is read.**
3. **OSPT00 is automatically cleared after data is set, so 0 is read.**
4. **Do not set OSPT00 to 1 other than in one-shot pulse output mode.**
5. **A write interval of two cycles or more of the count clock selected by prescaler mode register 00 (PRM00) is required to write to OSPT00 successively.**
6. **When the TOE00 is 0, set the TOE00, LVS00, and LVR00 at the same time with the 8-bit memory manipulation instruction. When the TOE00 is 1, the LVS00 and LVR00 can be set with the 1-bit memory manipulation instruction.**

### (4) Prescaler mode register 00 (PRM00)

This register is used to set the 16-bit timer counter 00 (TM00) count clock and the TI000, TI010 pin input valid edges.

PRM00 is set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets the value of PRM00 to 00H.

**Figure 6-8. Format of Prescaler Mode Register 00 (PRM00)**

Address: FF61H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | ES110 | ES100 | ES010 | ES000 | 0 | 0 | PRM001 | PRM000 |

| ES110 | ES100 | TI010 pin valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both falling and rising edges |

| ES010 | ES000 | TI000 pin valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both falling and rising edges |

| PRM001 | PRM000 | Count clock selection |
|---|---|---|
| 0 | 0 | $f_{XP}$ (10 MHz) |
| 0 | 1 | $f_{XP}/2^2$ (2.5 MHz) |
| 1 | 0 | $f_{XP}/2^8$ (39.06 kHz) |
| 1 | 1 | TI000 pin valid edge[Note] |

**Remarks 1.** $f_{XP}$: Oscillation frequency of clock supplied to peripheral hardware

**2.** ( ): $f_{XP}$ = 10 MHz

**Note** The external clock requires a pulse longer than two cycles of the internal count clock ($f_{XP}$).

**Cautions 1. Always set data to PRM00 after stopping the timer operation.**

**2. If the valid edge of the TI000 pin is to be set as the count clock, do not set the clear/start mode and the capture trigger at the valid edge of the TI000 pin.**

**3. In the following cases, note with caution that the valid edge of the TI0n0 pin is detected.**

**<1> Immediately after a system reset, if a high level is input to the TI0n0 pin, the operation of the 16-bit timer counter 00 (TM00) is enabled**

**→ If the rising edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.**

**<2> If the TM00 operation is stopped while the TI0n0 pin is high level, TM00 operation is then enabled after a low level is input to the TI0n0 pin**

**→ If the falling edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a falling edge is detected immediately after the TM00 operation is enabled.**

**<3> If the TM00 operation is stopped while the TI0n0 pin is low level, TM00 operation is then enabled after a high level is input to the TI0n0 pin**

**→ If the rising edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.**

**4. The sampling clock used to eliminate noise differs when a TI000 valid edge is used as the count clock and when it is used as a capture trigger. In the former case, the count clock is $f_{XP}$, and in the latter case the count clock is selected by prescaler mode register 00 (PRM00). The capture operation is not performed until the valid edge is sampled and the valid level is detected twice, thus eliminating noise with a short pulse width.**

**5. When using P31 as the input pin (TI010) of the valid edge, it cannot be used as a timer output (TO00). When using P31 as the timer output pin (TO00), it cannot be used as the input pin (TI010) of the valid edge.**

**Remark** n = 0, 1

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

This register sets port 3 input/output in 1-bit units.

When using the P31/TO00/TI010/INTP2 pin for timer output, set PM31 and the output latch of P31 to 0.

When using the P30/TI000/INTP0 and P31/TO00/TI010/INTP2 pins as a timer input, set PM30 and PM31 to 1.

At this time, the output latches of P30 and P31 can be either 0 or 1.

PM3 is set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets the value of PM3 to FFH.

**Figure 6-9. Format of Port Mode Register 3 (PM3)**

Address: FF23H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM3 | 1 | 1 | 1 | 1 | PM33 | PM32 | PM31 | PM30 |

| PM3n | P3n pin I/O mode selection (n = 0 to 3) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

## 6.4 Operation of 16-Bit Timer/Event Counter 00

### 6.4.1 Interval timer operation

Setting 16-bit timer mode control register 00 (TMC00) and capture/compare control register 00 (CRC00) as shown in Figure 6-10 allows operation as an interval timer.

Setting

The basic operation setting procedure is as follows.

<1> Set the CRC00 register (see **Figure 6-10** for the set value).
<2> Set any value to the CR000 register.
<3> Set the count clock by using the PRM00 register.
<4> Set the TMC00 register to start the operation (see **Figure 6-10** for the set value).

**Caution Changing the CR000 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation.**

**Remark** For how to enable the INTTM000 interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.

Interrupt requests are generated repeatedly using the count value set in 16-bit timer capture/compare register 000 (CR000) beforehand as the interval.

When the count value of 16-bit timer counter 00 (TM00) matches the value set to CR000, counting continues with the TM00 value cleared to 0 and the interrupt request signal (INTTM000) is generated.

The count clock of the 16-bit timer/event counter can be selected using bits 0 and 1 (PRM000, PRM001) of prescaler mode register 00 (PRM00).

### Figure 6-10. Control Register Settings for Interval Timer Operation

#### (a) 16-bit timer mode control register 00 (TMC00)

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 1 | 0/1 | 0 |

TMC003, TMC002, TMC001: Clears and starts on match between TM00 and CR000.

#### (b) Capture/compare control register 00 (CRC00)

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 |

CRC000: CR000 used as compare register

#### (c) Prescaler mode register 00 (PRM00)

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0/1 | 0/1 |

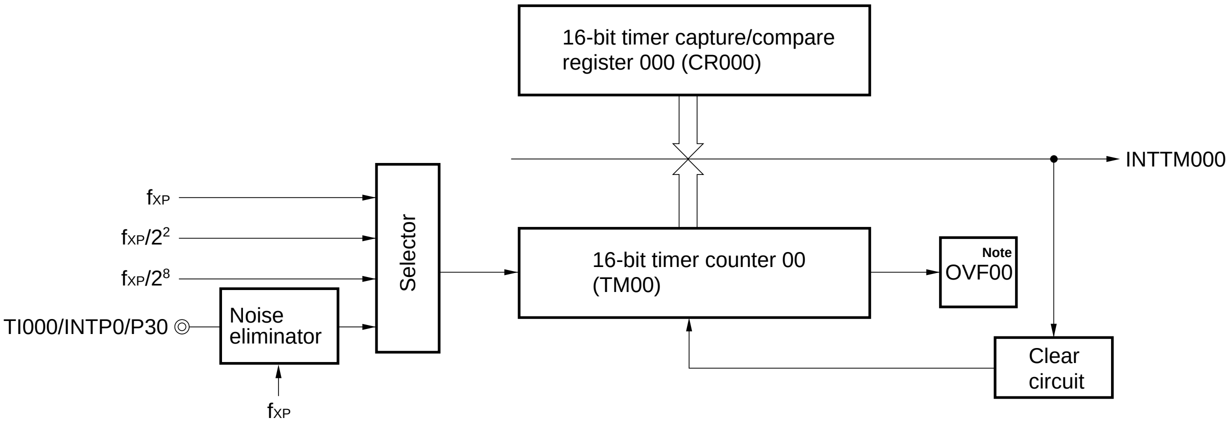
PRM001, PRM000: Selects count clock.

ES010, ES000: Setting invalid (setting “10” is prohibited.)

ES110, ES100: Setting invalid (setting “10” is prohibited.)

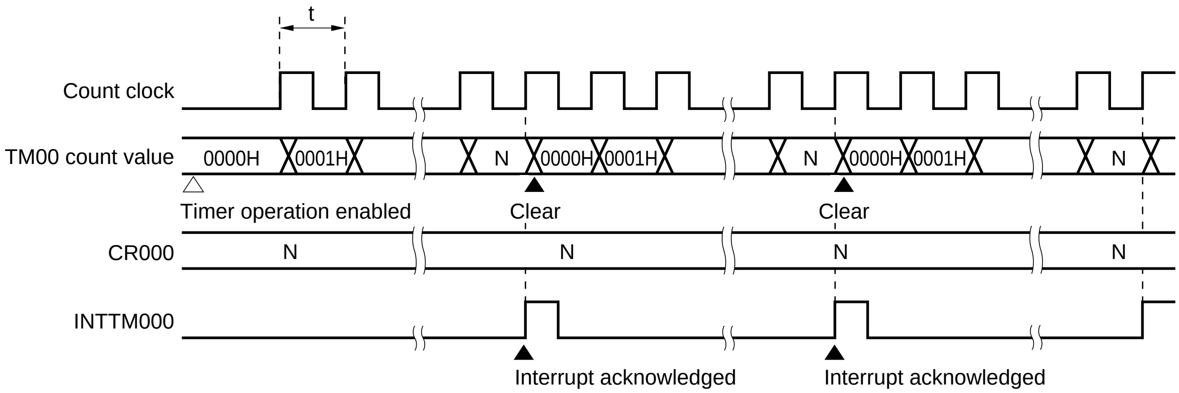
**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with the interval timer. See the description of the respective control registers for details.

**Figure 6-11. Interval Timer Configuration Diagram**



**Note** OVF00 is set to 1 only when 16-bit timer capture/compare register 000 (CR000) is set to FFFFH.

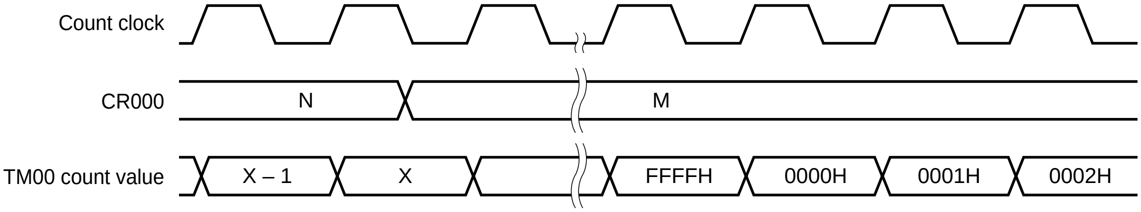
**Figure 6-12. Timing of Interval Timer Operation**



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

When the compare register is changed during timer count operation, if the value after 16-bit timer capture/compare register 000 (CR000) is changed is smaller than that of 16-bit timer counter 00 (TM00), TM00 continues counting, overflows and then restarts counting from 0. Thus, if the value (M) after the CR000 change is smaller than that (N) before the change, it is necessary to restart the timer after changing CR000.

**Figure 6-13. Timing After Change of Compare Register During Timer Count Operation (N → M: N > M)**



**Remark** N > X > M

### 6.4.2 External event counter operation

Setting

The basic operation setting procedure is as follows.

<1> Set the CRC00 register (see **Figure 6-14** for the set value).
<2> Set the count clock by using the PRM00 register.
<3> Set any value to the CR000 register (0000H cannot be set).
<4> Set the TMC00 register to start the operation (see **Figure 6-14** for the set value).

**Remarks 1.** For the setting of the TI000 pin, see **6.3 (5) Port mode register 3 (PM3)**.
**2.** For how to enable the INTTM000 interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.

The external event counter counts the number of external clock pulses to be input to the TI000 pin with using 16-bit timer counter 00 (TM00).

TM00 is incremented each time the valid edge specified by prescaler mode register 00 (PRM00) is input.

When the TM00 count value matches the 16-bit timer capture/compare register 000 (CR000) value, TM00 is cleared to 0 and the interrupt request signal (INTTM000) is generated.

Input a value other than 0000H to CR000. (A count operation with a pulse cannot be carried out.)

The rising edge, the falling edge, or both edges can be selected using bits 4 and 5 (ES000 and ES010) of prescaler mode register 00 (PRM00).

Because an operation is carried out only when the valid edge of the TI000 pin is detected twice after sampling with the internal clock ($f_{XP}$), noise with a short pulse width can be removed.

### Figure 6-14. Control Register Settings in External Event Counter Mode (with Rising Edge Specified)

#### (a) 16-bit timer mode control register 00 (TMC00)

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 1 | 0/1 | 0 |

Clears and starts on match between TM00 and CR000.

#### (b) Capture/compare control register 00 (CRC00)

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 |

CR000 used as compare register

#### (c) Prescaler mode register 00 (PRM00)

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0 | 1 | 0 | 0 | 1 | 1 |

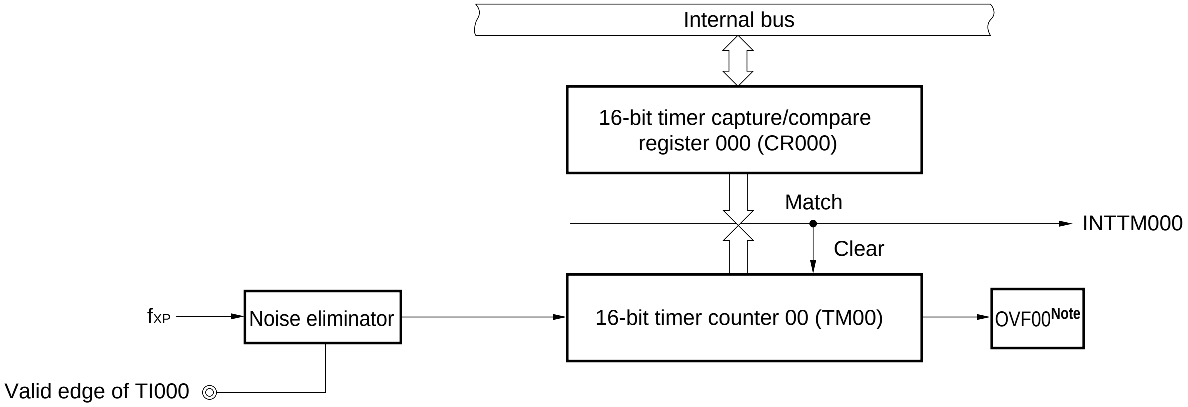
Selects external clock.

Specifies rising edge for pulse width detection.

Setting invalid (setting “10” is prohibited.)

**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with the external event counter. See the description of the respective control registers for details.

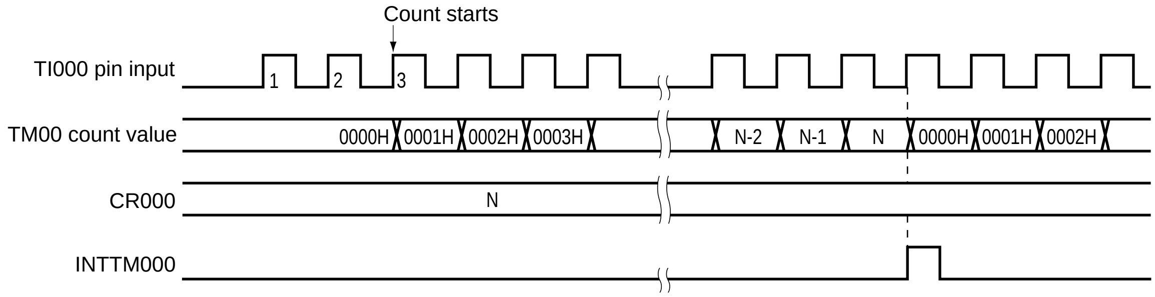
**Figure 6-15. External Event Counter Configuration Diagram**


Internal bus
16-bit timer capture/compare register 000 (CR000)
Match
INTTM000
Clear
fXP
Noise eliminator
16-bit timer counter 00 (TM00)
OVF00Note
Valid edge of TI000

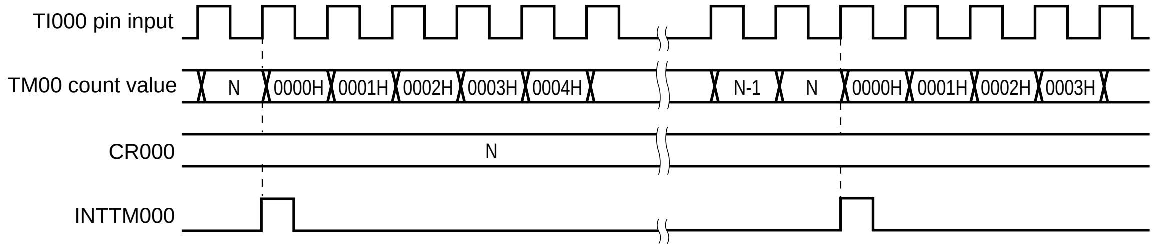
**Note** OVF00 is 1 only when 16-bit timer capture/compare register 000 (CR000) is set to FFFFH.

**Figure 6-16. External Event Counter Operation Timing (with Rising Edge Specified)**

**(1) INTTM000 generation timing immediately after operation starts**
**Counting is started after a valid edge is detected twice.**



**(2) INTTM000 generation timing after INTTM000 has been generated twice**



**Caution When reading the external event counter count value, TM00 should be read.**

### 6.4.3 Pulse width measurement operations

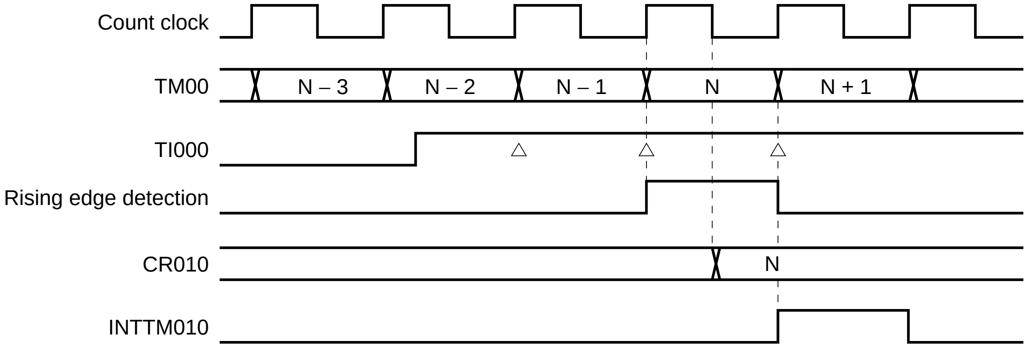
It is possible to measure the pulse width of the signals input to the TI000 pin and TI010 pin using 16-bit timer counter 00 (TM00).

There are two measurement methods: measuring with TM00 used in free-running mode, and measuring by restarting the timer in synchronization with the edge of the signal input to the TI000 pin.

When an interrupt occurs, read the valid value of the capture register, check the overflow flag, and then calculate the necessary pulse width. Clear the overflow flag after checking it.

The capture operation is not performed until the signal pulse width is sampled in the count clock cycle selected by prescaler mode register 00 (PRM00) and the valid level of the TI000 or TI010 pin is detected twice, thus eliminating noise with a short pulse width.

**Figure 6-17. CR010 Capture Operation with Rising Edge Specified**



Setting

The basic operation setting procedure is as follows.

<1> Set the CRC00 register (see **Figures 6-18**, **6-21**, **6-23**, and **6-25** for the set value).
<2> Set the count clock by using the PRM00 register.
<3> Set the TMC00 register to start the operation (see **Figures 6-18**, **6-21**, **6-23**, and **6-25** for the set value).

**Caution To use two capture registers, set the TI000 and TI010 pins.**

**Remarks 1.** For the setting of the TI000 (or TI010) pin, see **6.3 (5) Port mode register 3 (PM3)**.

**2.** For how to enable the INTTM000 (or INTTM010) interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.

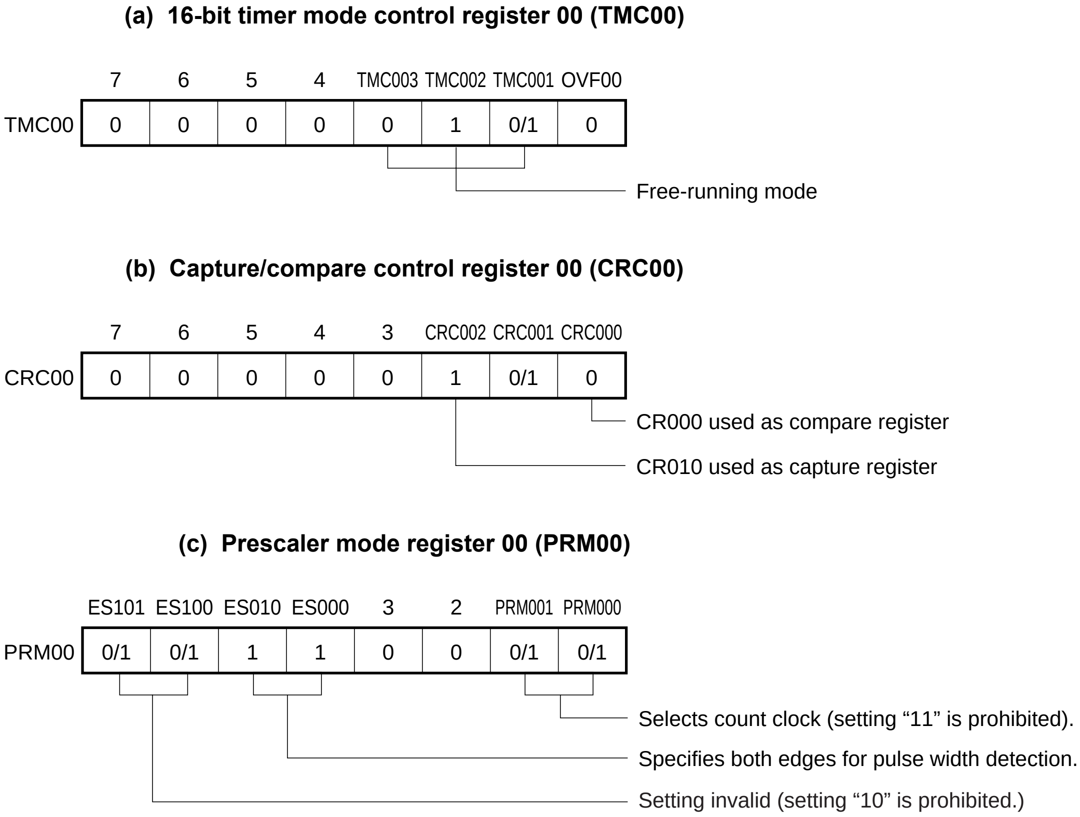
**(1) Pulse width measurement with free-running counter and one capture register**

When 16-bit timer counter 00 (TM00) is operated in free-running mode, and the edge specified by prescaler mode register 00 (PRM00) is input to the TI000 pin, the value of TM00 is taken into 16-bit timer capture/compare register 010 (CR010) and an external interrupt request signal (INTTM010) is set.

Specify both the rising and falling edges by using bits 4 and 5 (ES000 and ES010) of PRM00.

Sampling is performed using the count clock selected by PRM00, and a capture operation is only performed when a valid level of the TI000 pin is detected twice, thus eliminating noise with a short pulse width.

**Figure 6-18. Control Register Settings for Pulse Width Measurement with Free-Running Counter and One Capture Register (When TI000 and CR010 Are Used)**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0/1 | 0 |

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0/1 | 0 |

CR000 used as compare register
CR010 used as capture register

**(c) Prescaler mode register 00 (PRM00)**

| | ES101 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 1 | 1 | 0 | 0 | 0/1 | 0/1 |

Selects count clock (setting “11” is prohibited).
Specifies both edges for pulse width detection.
Setting invalid (setting “10” is prohibited.)

**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with pulse width measurement. See the description of the respective control registers for details.

**Figure 6-19. Configuration Diagram for Pulse Width Measurement by Free-Running Counter**

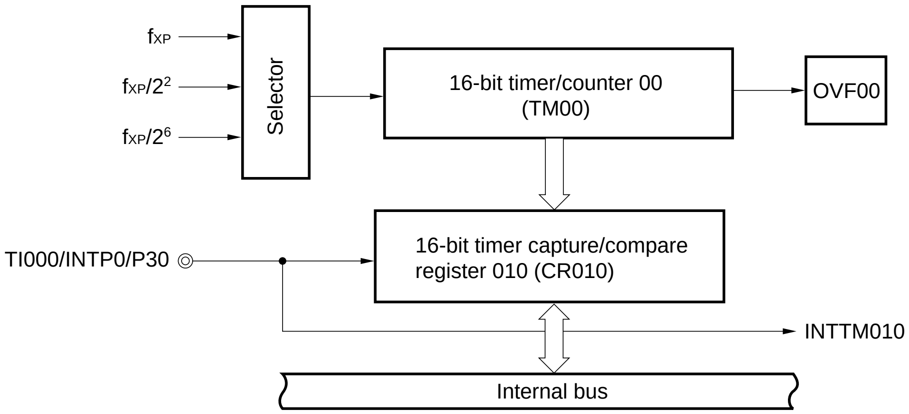
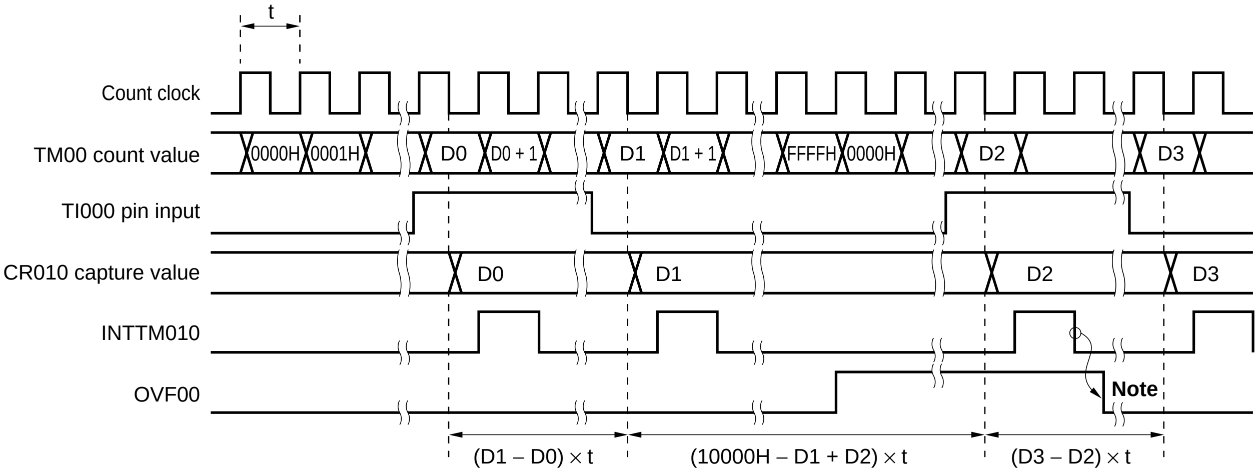

fXP
fXP/2²
fXP/2⁶
Selector
16-bit timer/counter 00 (TM00)
OVF00
16-bit timer capture/compare register 010 (CR010)
TI000/INTP0/P30
INTTM010
Internal bus

**Figure 6-20. Timing of Pulse Width Measurement Operation by Free-Running Counter and One Capture Register (with Both Edges Specified)**



**Note** OVF00 must be cleared by software.

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

When 16-bit timer counter 00 (TM00) is operated in free-running mode, it is possible to simultaneously measure the pulse widths of the two signals input to the TI000 pin and the TI010 pin.

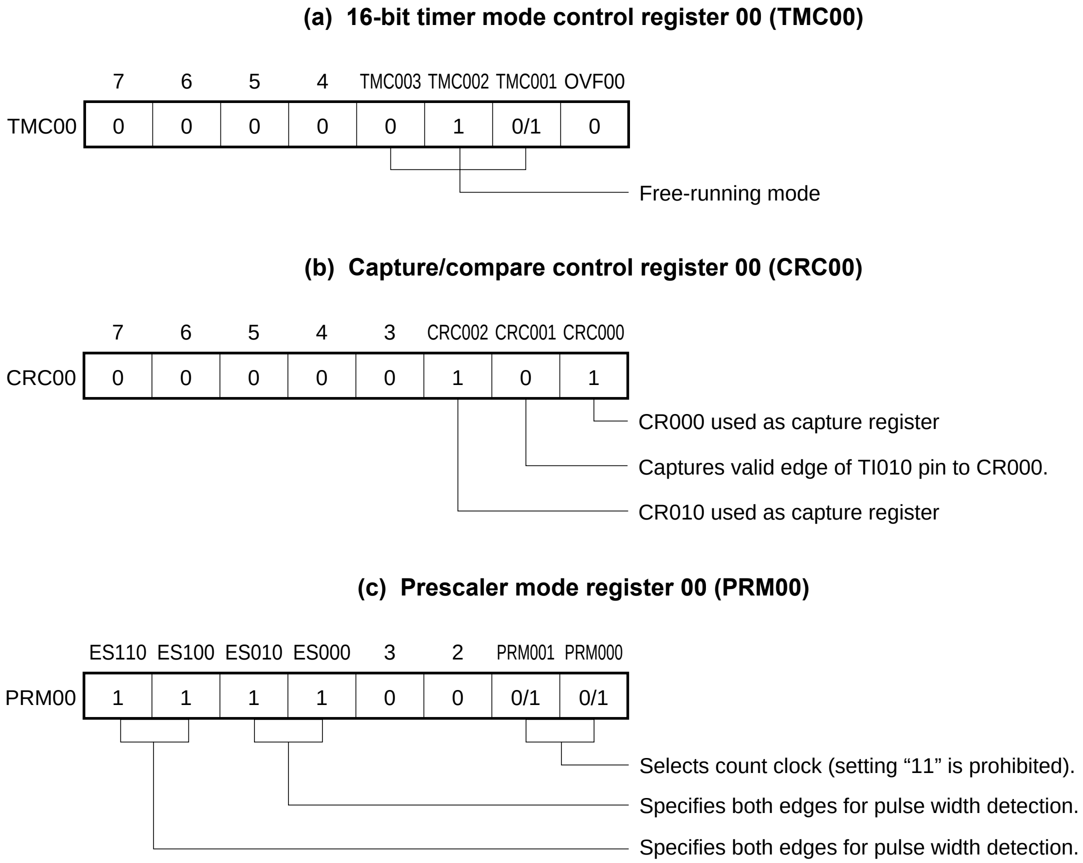
When the edge specified by bits 4 and 5 (ES000 and ES010) of prescaler mode register 00 (PRM00) is input to the TI000 pin, the value of TM00 is taken into 16-bit timer capture/compare register 010 (CR010) and an interrupt request signal (INTTM010) is set.

Also, when the edge specified by bits 6 and 7 (ES100 and ES110) of PRM00 is input to the TI010 pin, the value of TM00 is taken into 16-bit timer capture/compare register 000 (CR000) and an interrupt request signal (INTTM000) is set.

Specify both the rising and falling edges as the edges of the TI000 and TI010 pins, by using bits 4 and 5 (ES000 and ES010) and bits 6 and 7 (ES100 and ES110) of PRM00.

Sampling is performed using the count clock cycle selected by prescaler mode register 00 (PRM00), and a capture operation is only performed when a valid level of the TI000 or TI010 pin is detected twice, thus eliminating noise with a short pulse width.

**Figure 6-21. Control Register Settings for Measurement of Two Pulse Widths with Free-Running Counter**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0/1 | 0 |

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

CR000 used as capture register
Captures valid edge of TI010 pin to CR000.
CR010 used as capture register

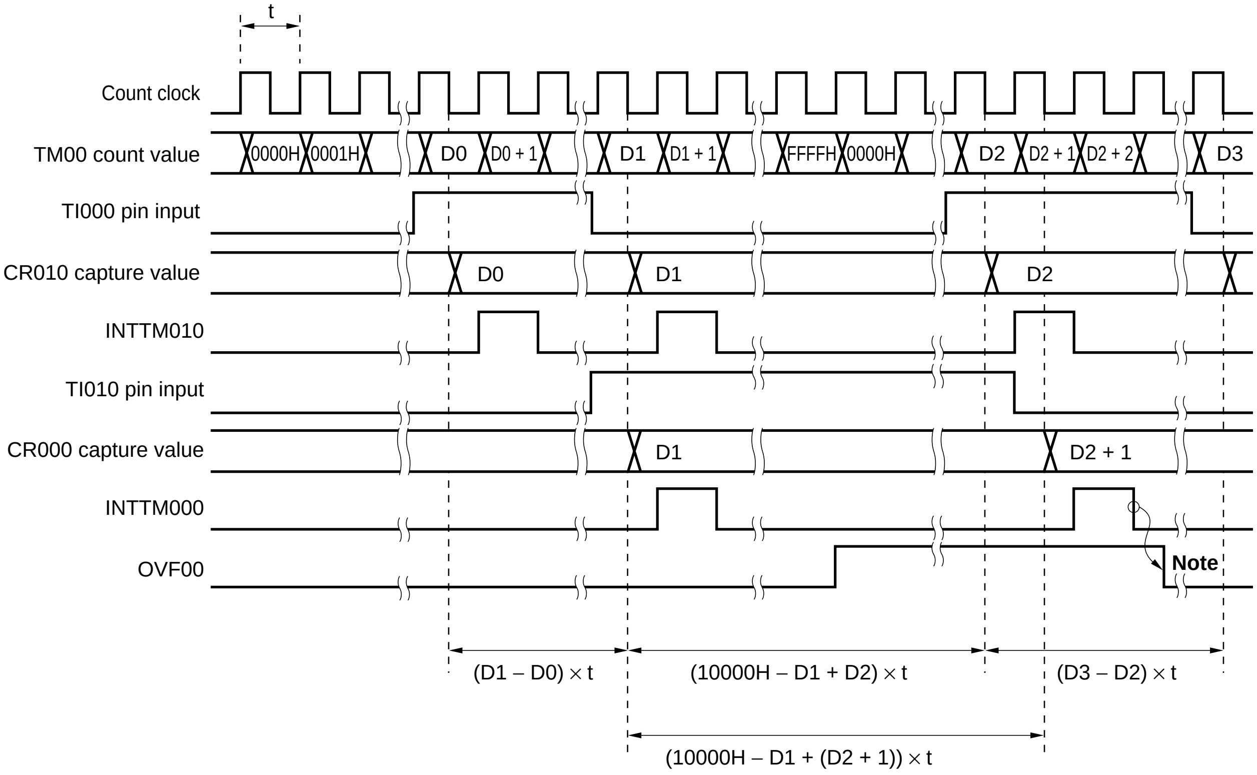
**(c) Prescaler mode register 00 (PRM00)**

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 1 | 1 | 1 | 1 | 0 | 0 | 0/1 | 0/1 |

Selects count clock (setting “11” is prohibited).
Specifies both edges for pulse width detection.
Specifies both edges for pulse width detection.

**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with pulse width measurement. See the description of the respective control registers for details.

**Figure 6-22. Timing of Pulse Width Measurement Operation with Free-Running Counter (with Both Edges Specified)**



**Note** OVF00 must be cleared by software.

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

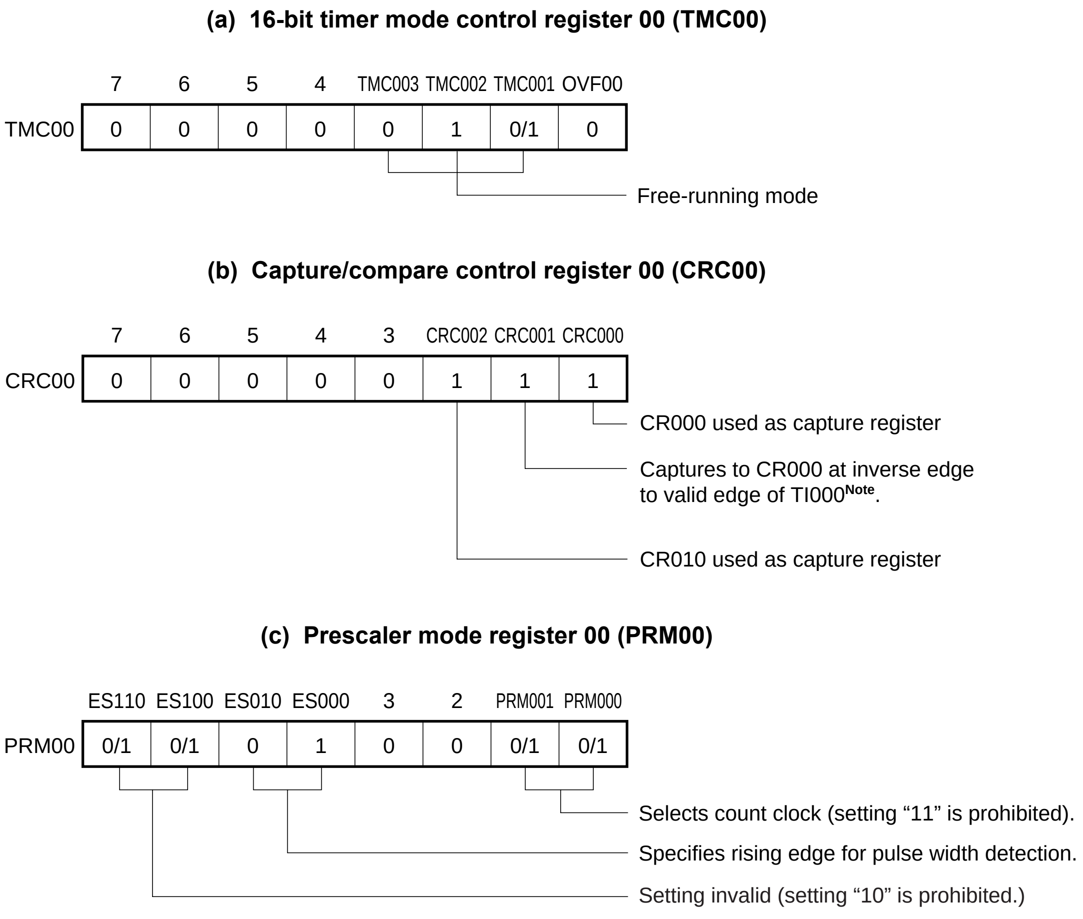
When 16-bit timer counter 00 (TM00) is operated in free-running mode, it is possible to measure the pulse width of the signal input to the TI000 pin.

When the rising or falling edge specified by bits 4 and 5 (ES000 and ES010) of prescaler mode register 00 (PRM00) is input to the TI000 pin, the value of TM00 is taken into 16-bit timer capture/compare register 010 (CR010) and an interrupt request signal (INTTM010) is set.

Also, when the inverse edge to that of the capture operation is input into CR010, the value of TM00 is taken into 16-bit timer capture/compare register 000 (CR000).

Sampling is performed using the count clock cycle selected by prescaler mode register 00 (PRM00), and a capture operation is only performed when a valid level of the TI000 pin is detected twice, thus eliminating noise with a short pulse width.

**Figure 6-23. Control Register Settings for Pulse Width Measurement with Free-Running Counter and Two Capture Registers (with Rising Edge Specified)**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0/1 | 0 |

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

CR000 used as capture register
Captures to CR000 at inverse edge to valid edge of TI000[Note].
CR010 used as capture register

**(c) Prescaler mode register 00 (PRM00)**

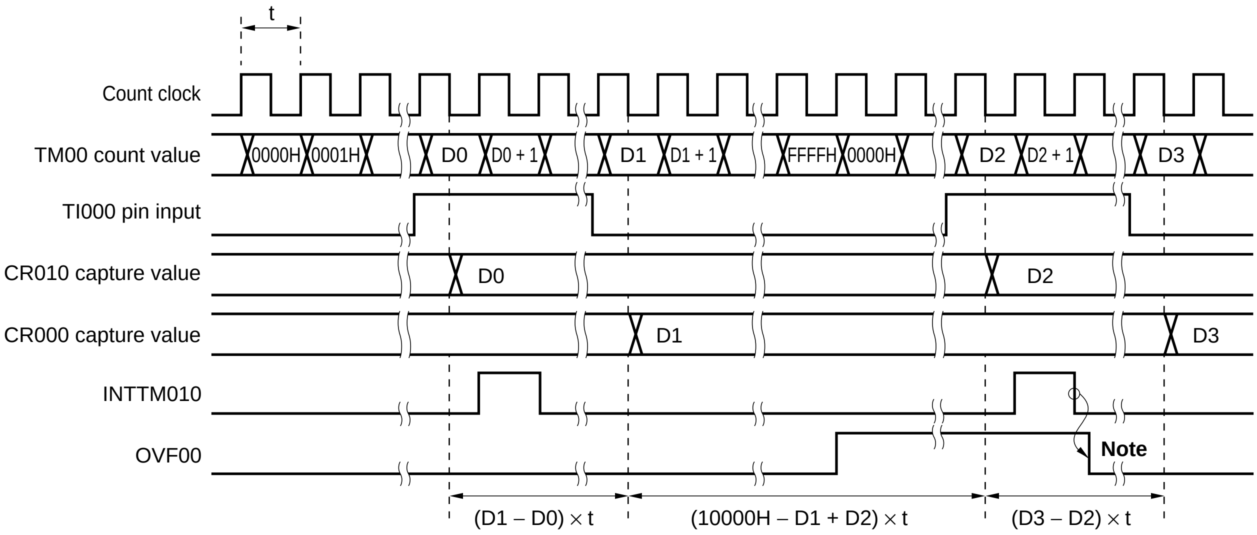
| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0 | 1 | 0 | 0 | 0/1 | 0/1 |

Selects count clock (setting “11” is prohibited).
Specifies rising edge for pulse width detection.
Setting invalid (setting “10” is prohibited.)

**Note** If the valid edge of TI000 pin is specified to be both the rising and falling edges, 16-bit timer capture/compare register 000 (CR000) cannot perform the capture operation. When the CRC001 bit value is 1, the TM00 count value is not captured in the CR000 register when a valid edge of the TI010 pin is detected, but the input from the TI010 pin can be used as an external interrupt source because INTTM000 is generated at that timing.

**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with pulse width measurement. See the description of the respective control registers for details.

**Figure 6-24. Timing of Pulse Width Measurement Operation by Free-Running Counter and Two Capture Registers (with Rising Edge Specified)**



**Note** OVF00 must be cleared by software.

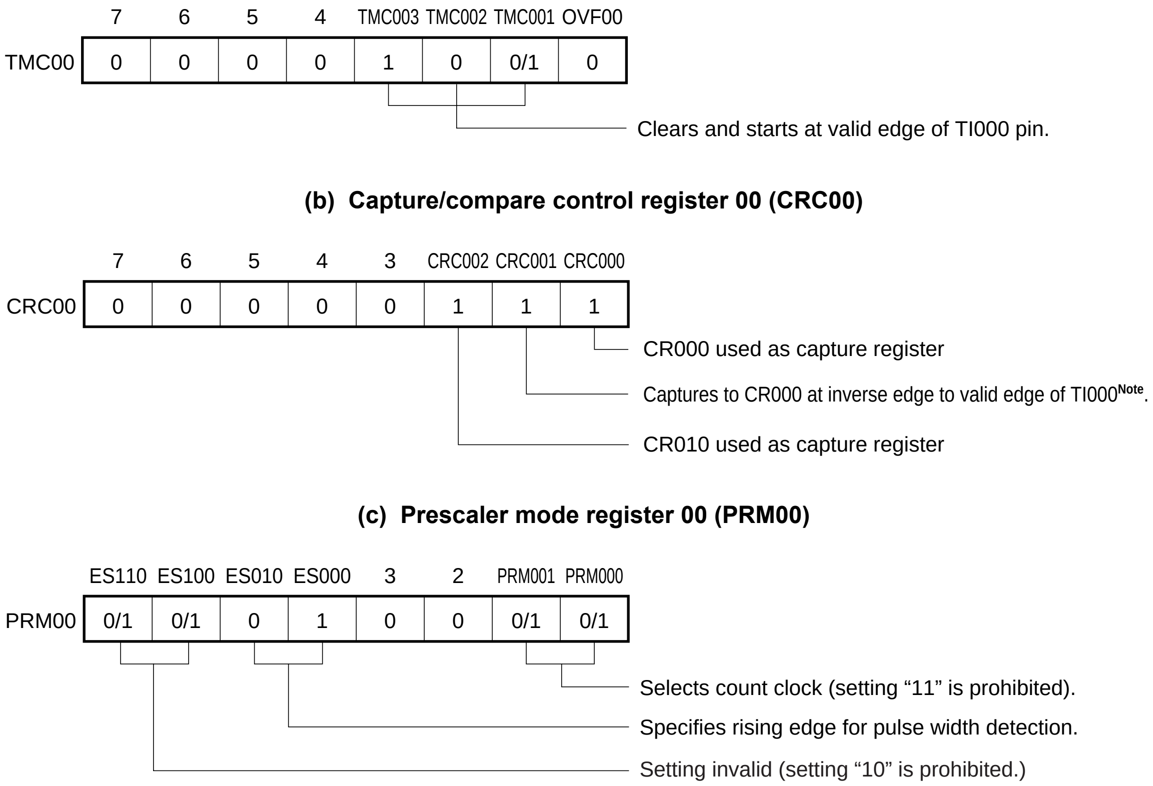
**(4) Pulse width measurement by means of restart**

When input of a valid edge to the TI000 pin is detected, the count value of 16-bit timer/counter 00 (TM00) is taken into 16-bit timer capture/compare register 010 (CR010), and then the pulse width of the signal input to the TI000 pin is measured by clearing TM00 and restarting the count.

The edge specification can be selected from two types, rising or falling edges, by bits 4 and 5 (ES000 and ES010) of prescaler mode register 00 (PRM00)

Sampling is performed at the interval selected by prescaler mode register 00 (PRM00) and a capture operation is only performed when a valid level of the TI000 pin is detected twice, thus eliminating noise with a short pulse width.

**Figure 6-25. Control Register Settings for Pulse Width Measurement by Means of Restart (with Rising Edge Specified)**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 0 | 0/1 | 0 |

TMC003, TMC002, TMC001: Clears and starts at valid edge of TI000 pin.

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

CRC000: CR000 used as capture register

CRC001: Captures to CR000 at inverse edge to valid edge of TI000[Note].

CRC002: CR010 used as capture register

**(c) Prescaler mode register 00 (PRM00)**

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0 | 1 | 0 | 0 | 0/1 | 0/1 |

PRM001, PRM000: Selects count clock (setting “11” is prohibited).

ES010, ES000: Specifies rising edge for pulse width detection.

ES110, ES100: Setting invalid (setting “10” is prohibited.)

**Note** If the valid edge of TI000 pin is specified to be both the rising and falling edges, 16-bit timer

### 6.4.4 Square-wave output operation

Setting

The basic operation setting procedure is as follows.

<1> Set the count clock by using the PRM00 register.
<2> Set the CRC00 register (see **Figure 6-27** for the set value).
<3> Set the TOC00 register (see **Figure 6-27** for the set value).
<4> Set any value to the CR000 register (0000H cannot be set).
<5> Set the TMC00 register to start the operation (see **Figure 6-27** for the set value).

**Caution Changing the CR000 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation.**

**Remarks 1.** For the setting of the TO00 pin, see **6.3 (5) Port mode register 3 (PM3)**.
**2.** For how to enable the INTTM000 interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.

A square wave with any selected frequency can be output at intervals determined by the count value preset to 16-bit timer capture/compare register 000 (CR000).

The TO00 pin output status is reversed at intervals determined by the count value preset to CR000 + 1 by setting bit 0 (TOE00) and bit 1 (TOC001) of 16-bit timer output control register 00 (TOC00) to 1. This enables a square wave with any selected frequency to be output.

**Figure 6-27. Control Register Settings in Square-Wave Output Mode (1/2)**

**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

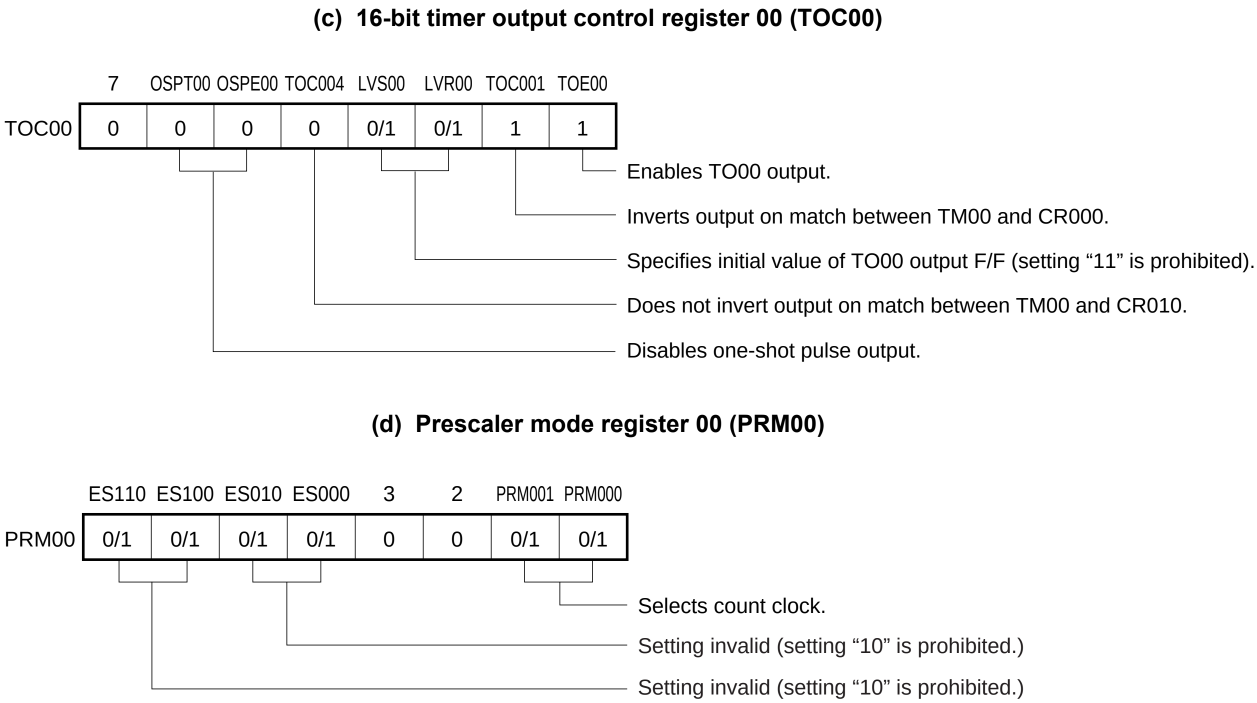
Clears and starts on match between TM00 and CR000.

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 |

CR000 used as compare register

### Figure 6-27. Control Register Settings in Square-Wave Output Mode (2/2)


(c) 16-bit timer output control register 00 (TOC00)

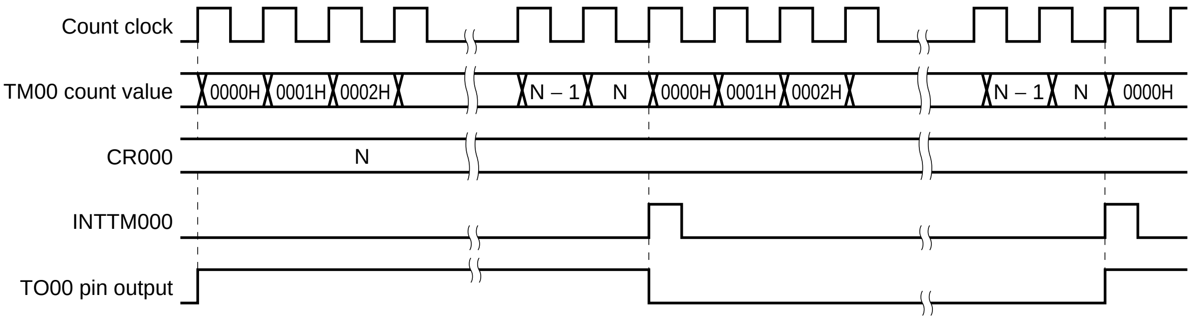
7
OSPT00
OSPE00
TOC004
LVS00
LVR00
TOC001
TOE00
TOC00
0
0
0
0
0/1
0/1
1
1
Enables TO00 output.
Inverts output on match between TM00 and CR000.
Specifies initial value of TO00 output F/F (setting “11” is prohibited).
Does not invert output on match between TM00 and CR010.
Disables one-shot pulse output.

(d) Prescaler mode register 00 (PRM00)

ES110
ES100
ES010
ES000
3
2
PRM001
PRM000
PRM00
0/1
0/1
0/1
0/1
0
0
0/1
0/1
Selects count clock.
Setting invalid (setting “10” is prohibited.)
Setting invalid (setting “10” is prohibited.)

**Remark** 0/1: Setting 0 or 1 allows another function to be used simultaneously with square-wave output. See the description of the respective control registers for details.

### Figure 6-28. Square-Wave Output Operation Timing



### 6.4.5 PPG output operations

Setting 16-bit timer mode control register 00 (TMC00) and capture/compare control register 00 (CRC00) as shown in Figure 6-29 allows operation as PPG (Programmable Pulse Generator) output.

Setting

The basic operation setting procedure is as follows.

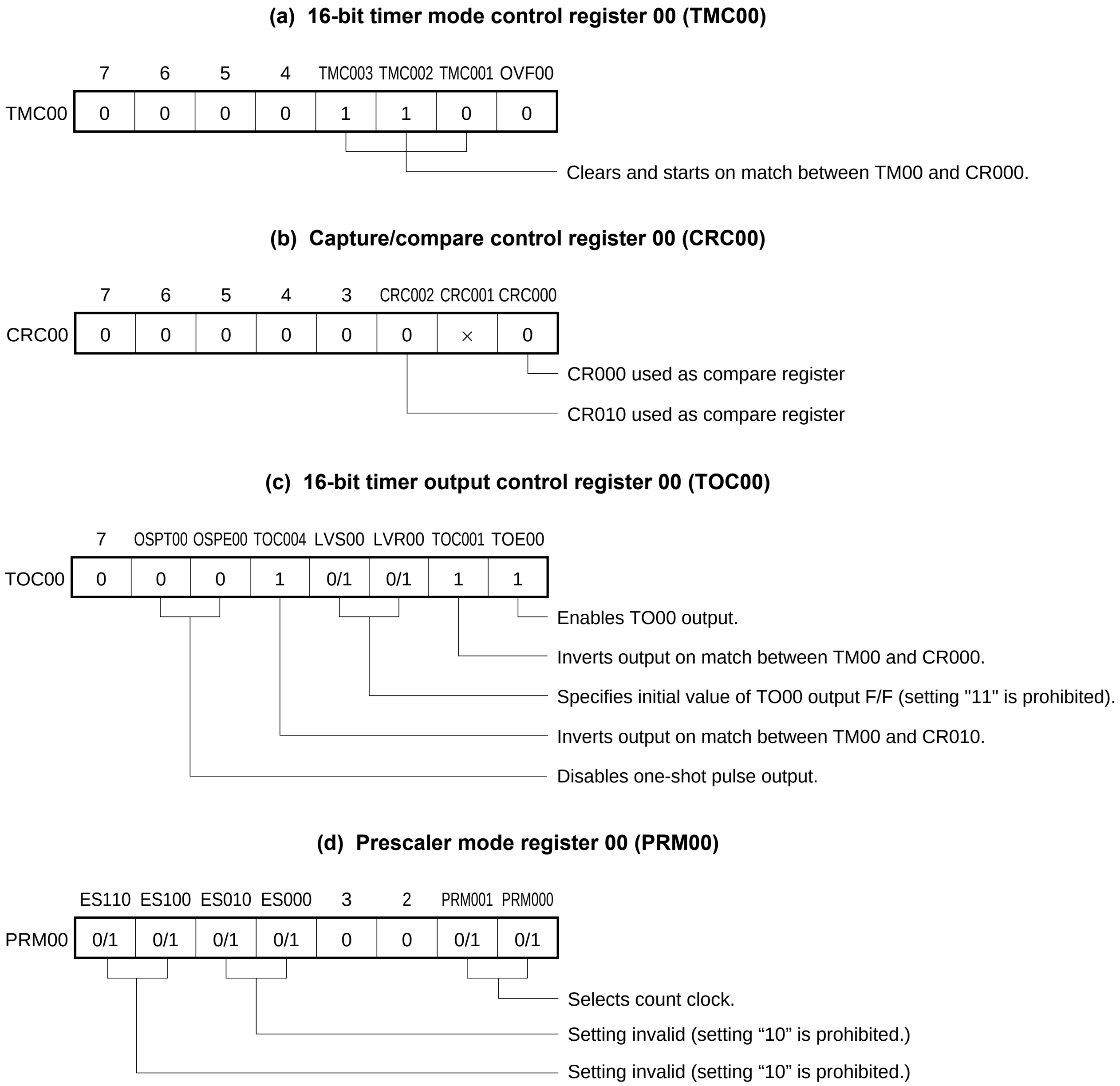
<1> Set the CRC00 register (see **Figure 6-29** for the set value).
<2> Set any value to the CR000 register as the cycle.
<3> Set any value to the CR010 register as the duty factor.
<4> Set the TOC00 register (see **Figure 6-29** for the set value).
<5> Set the count clock by using the PRM00 register.
<6> Set the TMC00 register to start the operation (see **Figure 6-29** for the set value).

**Caution Changing the CRC0n0 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation.**

**Remarks 1.** For the setting of the TO00 pin, see **6.3 (5) Port mode register 3 (PM3)**.
**2.** For how to enable the INTTM000 interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.
**3.** n = 0 or 1

In the PPG output operation, rectangular waves are output from the TO00 pin with the pulse width and the cycle that correspond to the count values preset in 16-bit timer capture/compare register 010 (CR010) and in 16-bit timer capture/compare register 000 (CR000), respectively.

**Figure 6-29. Control Register Settings for PPG Output Operation**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Clears and starts on match between TM00 and CR000.

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0 | × | 0 |

CR000 used as compare register

CR010 used as compare register

**(c) 16-bit timer output control register 00 (TOC00)**

| | 7 | OSPT00 | OSPE00 | TOC004 | LVS00 | LVR00 | TOC001 | TOE00 |
|---|---|---|---|---|---|---|---|---|
| TOC00 | 0 | 0 | 0 | 1 | 0/1 | 0/1 | 1 | 1 |

Enables TO00 output.

Inverts output on match between TM00 and CR000.

Specifies initial value of TO00 output F/F (setting "11" is prohibited).

Inverts output on match between TM00 and CR010.

Disables one-shot pulse output.

**(d) Prescaler mode register 00 (PRM00)**

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0/1 | 0/1 |

Selects count clock.

Setting invalid (setting “10” is prohibited.)

Setting invalid (setting “10” is prohibited.)

**Cautions 1. Values in the following range should be set in CR000 and CR010.**
**0000H < CR010 < CR000 ≤ FFFFH**

**2. The cycle of the pulse generated through PPG output (CR000 setting value + 1) has a duty of (CR010 setting value + 1)/(CR000 setting value + 1).**

**Remark** ×: Don’t care

**Figure 6-30. Configuration Diagram of PPG Output**

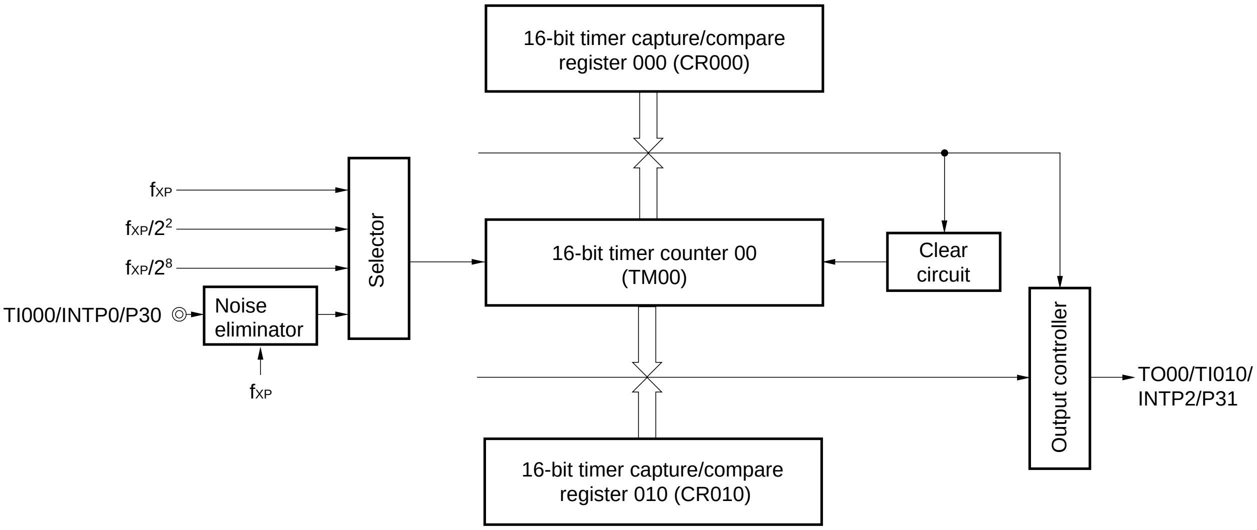
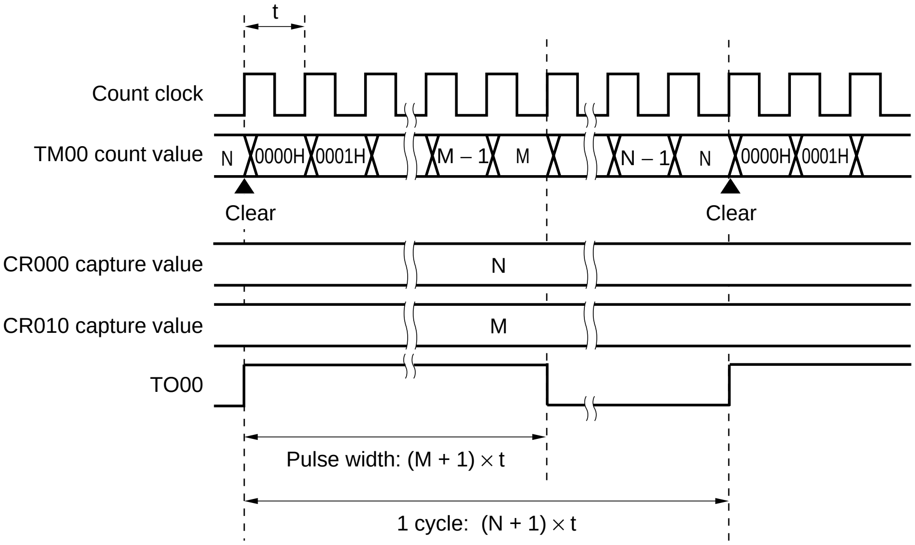

16-bit timer capture/compare register 000 (CR000)
fXP
fXP/2^2
fXP/2^8
TI000/INTP0/P30
Noise eliminator
fXP
Selector
16-bit timer counter 00 (TM00)
Clear circuit
Output controller
TO00/TI010/INTP2/P31
16-bit timer capture/compare register 010 (CR010)

**Figure 6-31. PPG Output Operation Timing**



**Remark** 0000H < M < N ≤ FFFFH

### 6.4.6 One-shot pulse output operation

16-bit timer/event counter 00 can output a one-shot pulse in synchronization with a software trigger or an external trigger (TI000 pin input).

Setting

The basic operation setting procedure is as follows.

<1> Set the count clock by using the PRM00 register.
<2> Set the CRC00 register (see **Figures 6-32** and **6-34** for the set value).
<3> Set the TOC00 register (see **Figures 6-32** and **6-34** for the set value).
<4> Set any value to the CR000 and CR010 registers (0000H cannot be set).
<5> Set the TMC00 register to start the operation (see **Figures 6-32** and **6-34** for the set value).

**Remarks 1.** For the setting of the TO00 pin, see **6.3 (5) Port mode register 3 (PM3)**.

**2.** For how to enable the INTTM000 (if necessary, INTTM010) interrupt, see **CHAPTER 13 INTERRUPT FUNCTIONS**.

**(1) One-shot pulse output with software trigger**

A one-shot pulse can be output from the TO00 pin by setting 16-bit timer mode control register 00 (TMC00), capture/compare control register 00 (CRC00), and 16-bit timer output control register 00 (TOC00) as shown in Figure 6-32, and by setting bit 6 (OSPT00) of the TOC00 register to 1 by software.

By setting the OSPT00 bit to 1, 16-bit timer/event counter 00 is cleared and started, and its output becomes active at the count value (N) set in advance to 16-bit timer capture/compare register 010 (CR010). After that, the output becomes inactive at the count value (M) set in advance to 16-bit timer capture/compare register 000 (CR000)[Note].

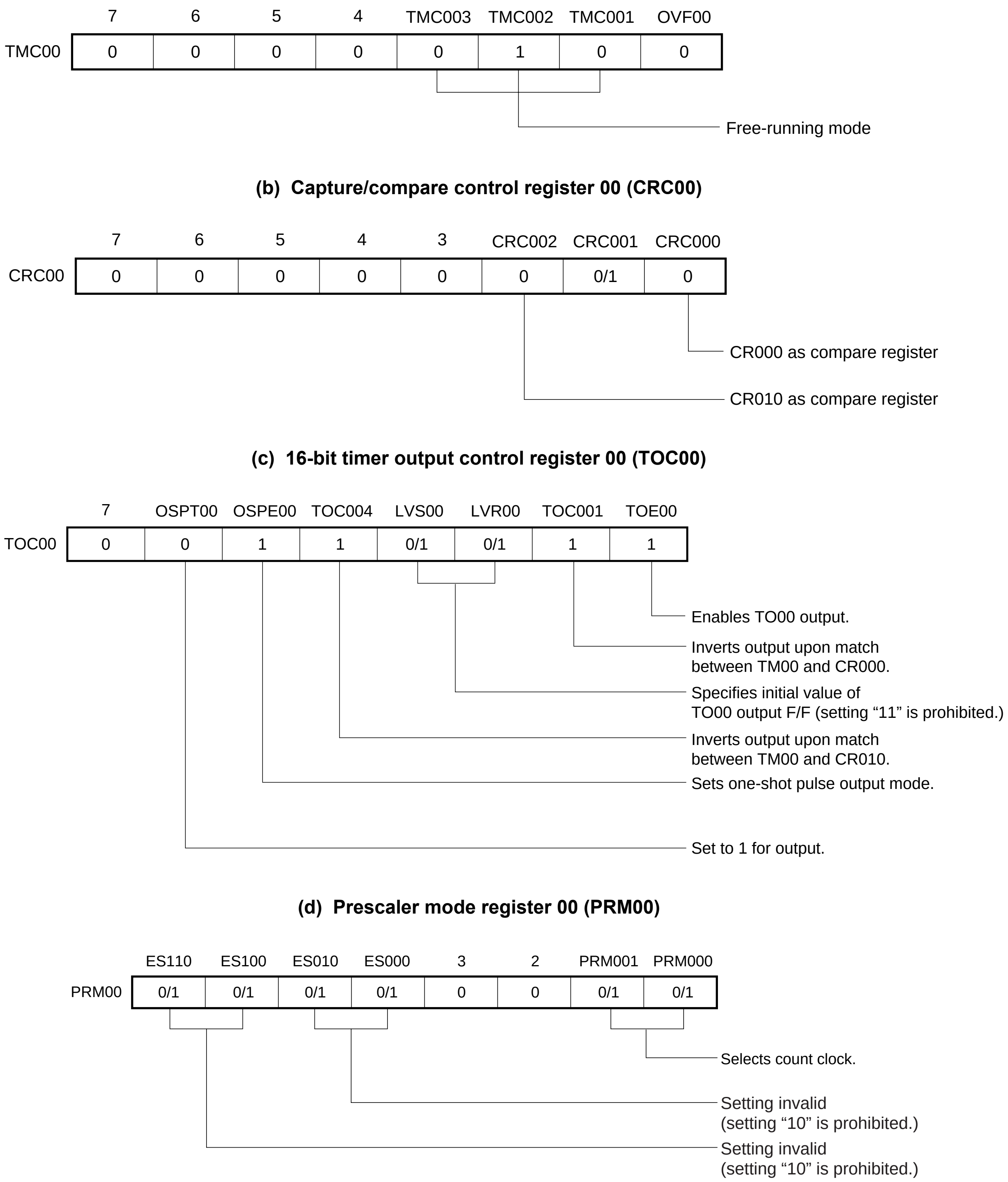
Even after the one-shot pulse has been output, the TM00 register continues its operation. To stop the TM00 register, the TMC003 and TMC002 bits of the TMC00 register must be cleared to 00.

**Note** The case where N < M is described here. When N > M, the output becomes active with the CR000 register and inactive with the CR010 register. Do not set N to M.

**Cautions 1. Do not set the OSPT00 bit to 1 again 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. When using the one-shot pulse output of 16-bit timer/event counter 00 with a software trigger, do not change the level of the TI000 pin or its alternate-function port pin.**
**Because the external trigger is valid even in this case, the timer is cleared and started even at the level of the TI000 pin or its alternate-function port pin, resulting in the output of a pulse at an undesired timing.**

### Figure 6-32. Control Register Settings for One-Shot Pulse Output with Software Trigger



(a) 16-bit timer mode control register 00 (TMC00)

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

TMC003 to TMC001: Free-running mode

(b) Capture/compare control register 00 (CRC00)

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0 |

CRC000: CR000 as compare register

CRC002: CR010 as compare register

(c) 16-bit timer output control register 00 (TOC00)

| | 7 | OSPT00 | OSPE00 | TOC004 | LVS00 | LVR00 | TOC001 | TOE00 |
|---|---|---|---|---|---|---|---|---|
| TOC00 | 0 | 0 | 1 | 1 | 0/1 | 0/1 | 1 | 1 |

TOE00: Enables TO00 output.

TOC001: Inverts output upon match between TM00 and CR000.

LVS00, LVR00: Specifies initial value of TO00 output F/F (setting “11” is prohibited.)

TOC004: Inverts output upon match between TM00 and CR010.

OSPE00: Sets one-shot pulse output mode.

OSPT00: Set to 1 for output.

(d) Prescaler mode register 00 (PRM00)

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0/1 | 0/1 |

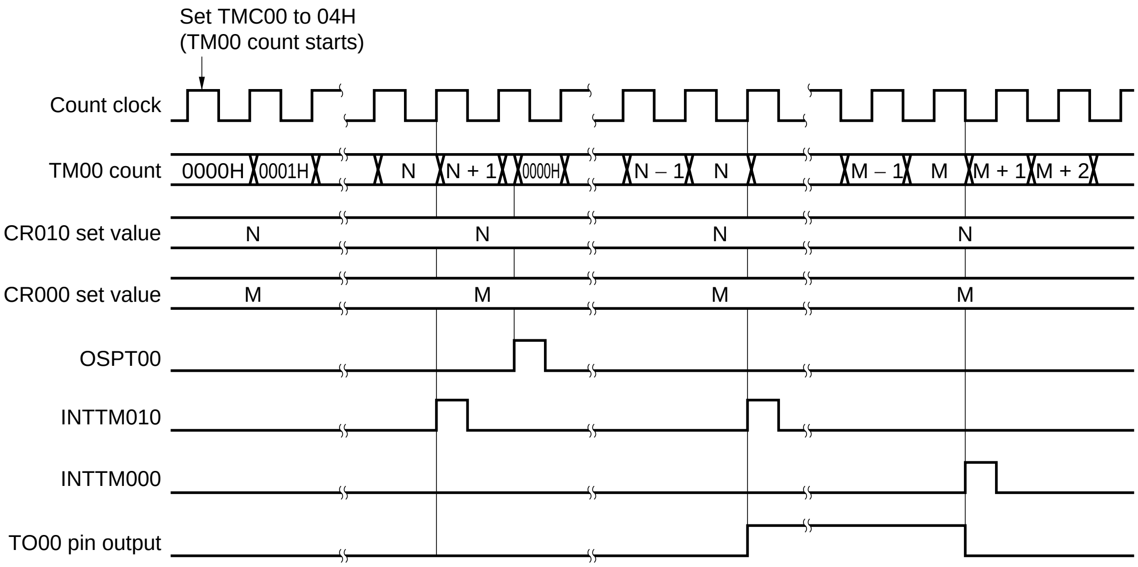
PRM001, PRM000: Selects count clock.

ES010, ES000: Setting invalid (setting “10” is prohibited.)

ES110, ES100: Setting invalid (setting “10” is prohibited.)

**Caution Do not set 0000H to the CR000 and CR010 registers.**

**Figure 6-33. Timing of One-Shot Pulse Output Operation with Software Trigger**



**Caution 16-bit timer counter 00 starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC003 and TMC002 bits.**

**Remark** N < M

**(2) One-shot pulse output with external trigger**

A one-shot pulse can be output from the TO00 pin by setting 16-bit timer mode control register 00 (TMC00), capture/compare control register 00 (CRC00), and 16-bit timer output control register 00 (TOC00) as shown in Figure 6-34, and by using the valid edge of the TI000 pin as an external trigger.

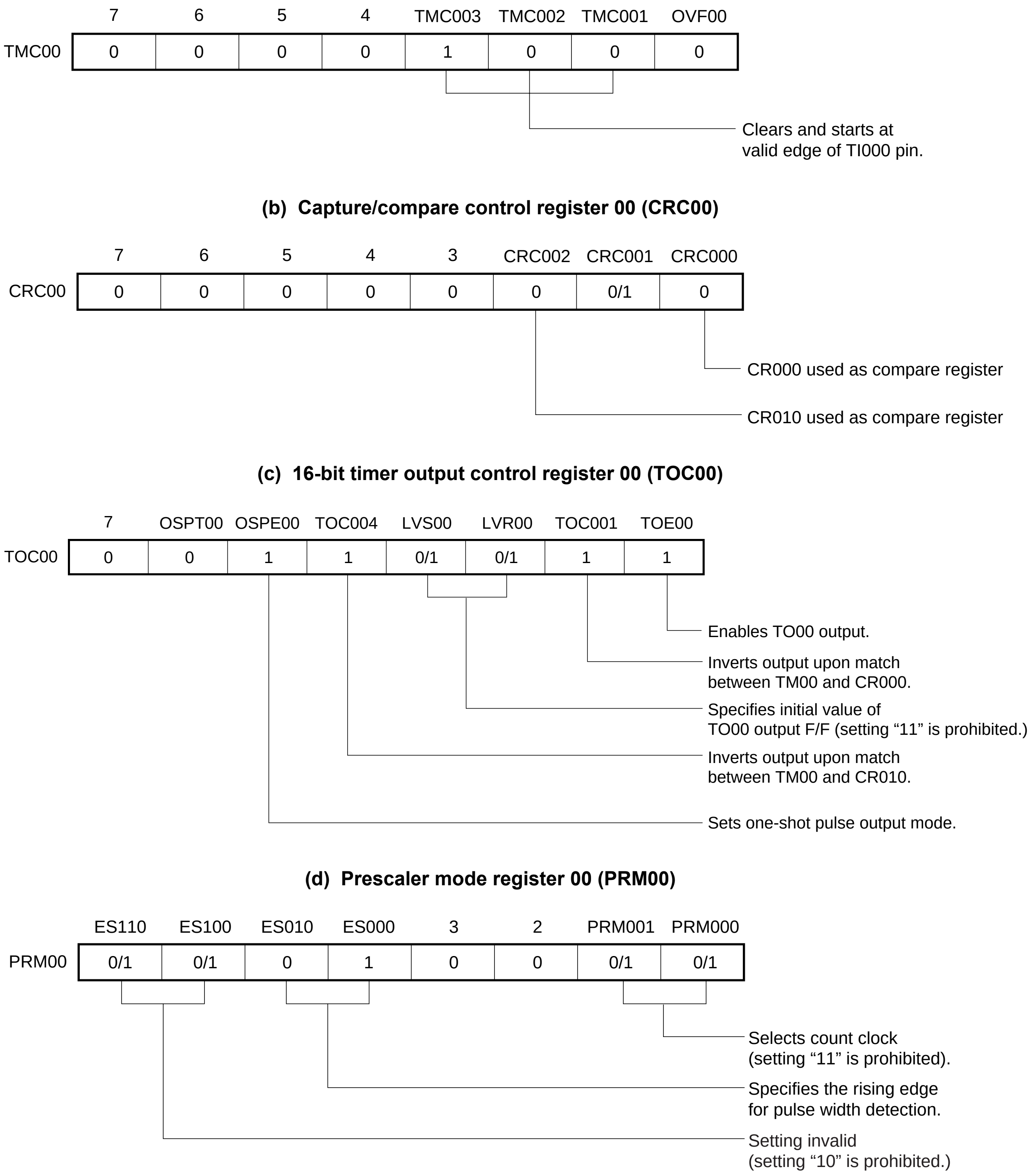
The valid edge of the TI000 pin is specified by bits 4 and 5 (ES000, ES010) of prescaler mode register 00 (PRM00). The rising, falling, or both the rising and falling edges can be specified.

When the valid edge of the TI000 pin is detected, the 16-bit timer/event counter is cleared and started, and the output becomes active at the count value set in advance to 16-bit timer capture/compare register 010 (CR010). After that, the output becomes inactive at the count value set in advance to 16-bit timer capture/compare register 000 (CR000)[Note].

**Note** The case where N < M is described here. When N > M, the output becomes active with the CR000 register and inactive with the CR010 register. Do not set N to M.

**Caution Do not input the external trigger again 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.**

**Figure 6-34. Control Register Settings for One-Shot Pulse Output with External Trigger (with Rising Edge Specified)**



**(a) 16-bit timer mode control register 00 (TMC00)**

| | 7 | 6 | 5 | 4 | TMC003 | TMC002 | TMC001 | OVF00 |
|---|---|---|---|---|---|---|---|---|
| TMC00 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Clears and starts at
valid edge of TI000 pin.

**(b) Capture/compare control register 00 (CRC00)**

| | 7 | 6 | 5 | 4 | 3 | CRC002 | CRC001 | CRC000 |
|---|---|---|---|---|---|---|---|---|
| CRC00 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0 |

CR000 used as compare register
CR010 used as compare register

**(c) 16-bit timer output control register 00 (TOC00)**

| | 7 | OSPT00 | OSPE00 | TOC004 | LVS00 | LVR00 | TOC001 | TOE00 |
|---|---|---|---|---|---|---|---|---|
| TOC00 | 0 | 0 | 1 | 1 | 0/1 | 0/1 | 1 | 1 |

Enables TO00 output.
Inverts output upon match
between TM00 and CR000.
Specifies initial value of
TO00 output F/F (setting “11” is prohibited.)
Inverts output upon match
between TM00 and CR010.
Sets one-shot pulse output mode.

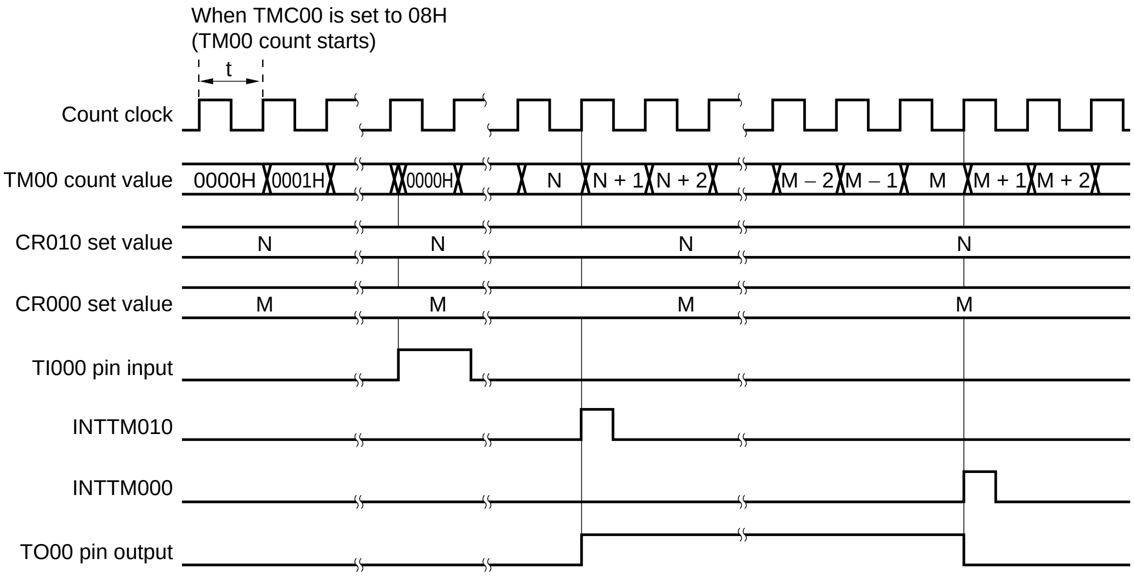
**(d) Prescaler mode register 00 (PRM00)**

| | ES110 | ES100 | ES010 | ES000 | 3 | 2 | PRM001 | PRM000 |
|---|---|---|---|---|---|---|---|---|
| PRM00 | 0/1 | 0/1 | 0 | 1 | 0 | 0 | 0/1 | 0/1 |

Selects count clock
(setting “11” is prohibited).
Specifies the rising edge
for pulse width detection.
Setting invalid
(setting “10” is prohibited.)

**Caution Do not set the CR000 and CR010 registers to 0000H.**

**Figure 6-35. Timing of One-Shot Pulse Output Operation with External Trigger (with Rising Edge Specified)**



**Caution 16-bit timer counter 00 starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC002 and TMC003 bits.**

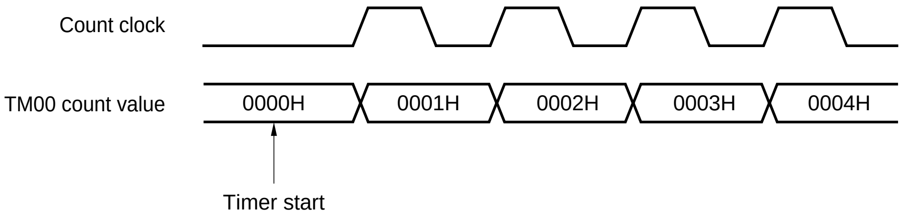
**Remark** N < M

## 6.5 Cautions Related to 16-Bit Timer/Event Counter 00

**(1) Timer start errors**

An error of up to one clock may occur in the time required for a match signal to be generated after timer start. This is because 16-bit timer counter 00 (TM00) is started asynchronously to the count clock.

**Figure 6-36. Start Timing of 16-Bit Timer Counter 00 (TM00)**



**(2) 16-bit timer counter 00 (TM00) operation**

<1> 16-bit timer counter 00 (TM00) starts operation at the moment TMC002 and TMC003 (operation stop mode) are set to a value other than 0, 0, respectively. Set TMC002 and TMC003 to 0, 0 to stop the operation.

<2> Even if TM00 is read, the value is not captured by 16-bit timer capture/compare register 010 (CR010).

<3> During TM00 is read, the count clock is stopped.

<4> Regardless of the CPU’s operation mode, when the timer stops, the signals input to pins TI000/TI010 are not acknowledged.

**(3) Setting of 16-bit timer capture/compare registers 000, 010 (CR000, CR010)**

<1> Set 16-bit timer capture/compare register 000 (CR000) to other than 0000H in the clear & start mode entered on match between TM00 and CR000. This means a 1-pulse count operation cannot be performed when this register is used as an external event counter.

<2> When the clear & start mode entered on a match between TM00 and CR000 is selected, CR000 should not be specified as a capture register.

<3> In the free-running mode and in the clear & start mode using the valid edge of the TI000 pin, if CR0n0 is set to 0000H, an interrupt request (INTTM0n0) is generated when CR0n0 changes from 0000H to 0001H following overflow (FFFFH).

<4> If the new value of CR0n0 is less than the value of TM00, TM00 continues counting, overflows, and then starts counting from 0 again. If the new value of CR0n0 is less than the old value, therefore, the timer must be reset to be restarted after the value of CR0n0 is changed.

**(4) Capture register data retention**

The values of 16-bit timer capture/compare register 0n0 (CR0n0) after 16-bit timer/event counter 00 has stopped are not guaranteed.

**Remark** n = 0, 1

**(5) Setting of 16-bit timer mode control register 00 (TMC00)**

The timer operation must be stopped before writing to bits other than the OVF flag.

**(6) Setting of capture/compare control register 00 (CRC00)**

The timer operation must be stopped before setting CRC00.

**(7) Setting of 16-bit timer output control register 00 (TOC00)**

<1> Timer operation must be stopped before setting other than OSPT00.

<2> If LVS00 and LVR00 are read, 0 is read.

<3> OSPT00 is automatically cleared after data is set, so 0 is read.

<4> Do not set OSPT00 to 1 other than in one-shot pulse output mode.

<5> A write interval of two cycles or more of the count clock selected by prescaler mode register 00 (PRM00) is required to write to OSPT00 successively.

**(8) Setting of prescaler mode register 00 (PRM00)**

Always set data to PRM00 after stopping the timer operation.

**(9) Valid edge setting**

Set the valid edge of the TI000 pin with bits 4 and 5 (ES000 and ES010) of prescaler mode register 00 (PRM00) after stopping the timer operation.

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

One-shot pulse output normally operates only in the free-running mode or in the clear & start mode at the valid edge of the TI000 pin. Because an overflow does not occur in the clear & start mode on a match between TM00 and CR000, one-shot pulse output is not possible.

**(11) One-shot pulse output by software**

<1> Do not set the OSPT00 bit to 1 again 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> When using the one-shot pulse output of 16-bit timer/event counter 00 with a software trigger, do not change the level of the TI000 pin or its alternate function port pin. Because the external trigger is valid even in this case, the timer is cleared and started even at the level of the TI000 pin or its alternate function port pin, resulting in the output of a pulse at an undesired timing.

<3> Do not set the 16-bit timer capture/compare registers 000 and 010 (CR000 and CR010) to 0000H.

**(12) One-shot pulse output with external trigger**

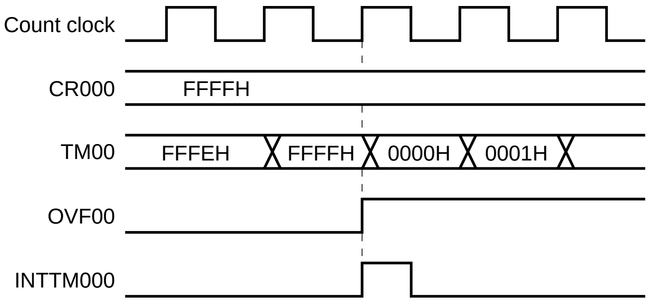
<1> Do not input the external trigger again while the one-shot pulse is being output.To output the one-shot pulse again, wait until the current one-shot pulse output is completed.

<2> Do not set the 16-bit timer capture/compare registers 000 and 010 (CR000 and CR010) to 0000H.

**(13) Operation of OVF00 flag**

<1> The OVF00 flag is also set to 1 in the following case.
Either of the clear & start mode entered on a match between TM00 and CR000, clear & start at the valid edge of the TI000 pin, or free-running mode is selected.
↓
CR000 is set to FFFFH.
↓
When TM00 is counted up from FFFFH to 0000H.

**Figure 6-37. Operation Timing of OVF00 Flag**

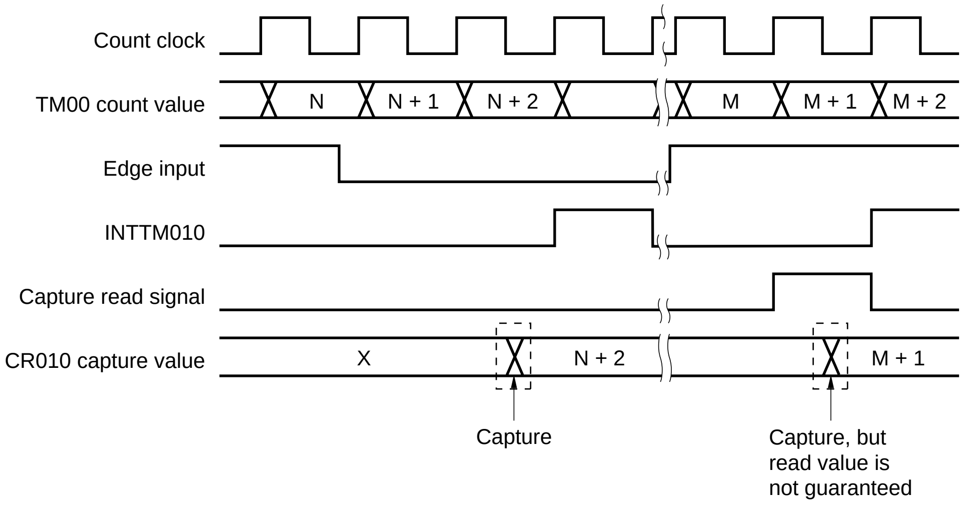


<2> Even if the OVF00 flag is cleared before the next count clock is counted (before TM00 becomes 0001H) after the occurrence of a TM00 overflow, the OVF00 flag is re-set newly and clear is disabled.

**(14) Conflicting operations**

If the register read period and the input of the capture trigger conflict when CR000/CR010 is used as a capture register, the capture trigger input takes precedence and the read data is undefined. Also, if the count stop of the timer and the input of the capture trigger conflict, the captured data is undefined.

**Figure 6-38. Capture Register Data Retention Timing**



**(15) Capture operation**

<1> If the TI000 pin is specified as the valid edge of the count clock, a capture operation by the capture register specified as the trigger for the TI000 pin is not possible.

<2> If both the rising and falling edges are selected as the valid edges of the TI000 pin, capture is not performed.

<3> When the CRC001 bit value is 1, the TM00 count value is not captured in the CR000 register when a valid edge of the TI010 pin is detected, but the input from the TI010 pin can be used as an external interrupt source because INTTM000 is generated at that timing.

<4> To ensure the reliability of the capture operation, the capture trigger requires a pulse longer than two cycles of the count clock selected by prescaler mode register 00 (PRM00).

<5> The capture operation is performed at the fall of the count clock. A interrupt request input (INTTM0n0), however, occurs at the rise of the next count clock.

<6> To use two capture registers, set the TI000 and TI010 pins.

**(16) Compare operation**

The capture operation may not be performed for CR0n0 set in compare mode even if a capture trigger is input.

**Remark** n = 0, 1

### (17) Changing compare register during timer operation

<1> With the 16-bit timer capture/compare register 0n0 (CR0n0) used as a compare register, when changing CR0n0 around the timing of a match between 16-bit timer counter 00 (TM00) and 16-bit timer capture/compare register 0n0 (CR0n0) during timer counting, the change timing may conflict with the timing of the match, so the operation is not guaranteed in such cases. To change CR0n0 during timer counting, follow the procedure below using an INTTM000 interrupt.

<Changing cycle (CR000)>

1. Disable the timer output inversion operation at the match between TM00 and CR000 (TOC001 = 0).
2. Disable the INTTM000 interrupt (TMMK000 = 1).
3. Rewrite CR000.
4. Wait for 1 cycle of the TM00 count clock.
5. Enable the timer output inversion operation at the match between TM00 and CR000 (TOC001 = 1).
6. Clear the interrupt request flag of INTTM000 (TMIF000 = 0).
7. Enable the INTTM000 interrupt (TMMK000 = 0).

<Changing duty (CR010)>

1. Disable the timer output inversion operation at the match between TM00 and CR010 (TOC004 = 0).
2. Disable the INTTM000 interrupt (TMMK000 = 1).
3. Rewrite CR010.
4. Wait for 1 cycle of the TM00 count clock.
5. Enable the timer output inversion operation at the match between TM00 and CR010 (TOC004 = 1).
6. Clear the interrupt request flag of INTTM000 (TMIF000 = 0).
7. Enable the INTTM000 interrupt (TMMK000 = 0).

While interrupts and timer output inversion are disabled (1 to 4 above), timer counting is continued. If the value to be set in CR0n0 is small, the value of TM00 may exceed CR0n0. Therefore, set the value, considering the time lapse of the timer clock and CPU clock after an INTTM000 interrupt has been generated.

**Remark** n = 0, 1

<2> If CR010 is changed during timer counting without performing processing <1> above, the value in CR010 may be rewritten twice or more, causing an inversion of the output level of the TO00 pin at each rewrite.

### (18) Edge detection

<1> In the following cases, note with caution that the valid edge of the TI0n0 pin is detected.

(a) Immediately after a system reset, if a high level is input to the TI0n0 pin, the operation of the 16-bit timer counter 00 (TM00) is enabled

→ If the rising edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.

(b) If the TM00 operation is stopped while the TI0n0 pin is high level, TM00 operation is then enabled after a low level is input to the TI0n0 pin

→ If the falling edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a falling edge is detected immediately after the TM00 operation is enabled.

(c) When the TM00 operation is stopped while the TI0n0 pin is low level, TM00 operation is then enabled after a high level is input to the TI0n0 pin

→ If the rising edge or both rising and falling edges are specified as the valid edge, of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.

**Remark** n = 0, 1

<2> The sampling clock used to remove noise differs when a TI000 valid edge is used as the count clock and when it is used as a capture trigger. In the former case, the count clock is $f_{XP}$, and in the latter case the count clock is selected by prescaler mode register 00 (PRM00). The capture operation is not performed until the valid edge is sampled and the valid level is detected twice, thus eliminating, noise with a short pulse width.

### (19) External event counter

When reading the external event counter count value, TM00 should be read.

### (20) PPG output

<1> Values in the following range should be set in CR000 and CR010:
0000H < CR010 < CR000 ≤ FFFFH (setting CR000 to 0000H is prohibited)

<2> The cycle of the pulse generated through PPG output (CR000 setting value + 1) has a duty of (CR010 setting value + 1)/(CR000 setting value + 1).

### (21) STOP mode or system clock stop mode setting

Except when TI000 pin valid edge is selected as the count clock, stop the timer operation before setting STOP mode or system clock stop mode; otherwise the timer may malfunction when the system clock starts.

### (22) P31/TI010/TO00 pin

When using P31 as the input pin (TI010) of the valid edge, it cannot be used as a timer output pin (TO00). When using P31 as the timer output pin (TO00), it cannot be used as the input pin (TI010) of the valid edge.

# CHAPTER 7 8-BIT TIMER 80

## 7.1 Function of 8-Bit Timer 80

8-bit timer 80 has an 8-bit interval timer function and generates an interrupt at intervals specified in advance.

Table 7-1. Interval Time of 8-Bit Timer 80

| | Minimum Interval Time | Maximum Interval Time | Resolution |
|---|---|---|---|
| $f_{XP}$ = 8.0 MHz | $2^6/f_{XP}$ (8 μs) | $2^{14}/f_{XP}$ (2.05 ms) | $2^6/f_{XP}$ (8 μs) |
| | $2^8/f_{XP}$ (32 μs) | $2^{16}/f_{XP}$ (8.19 ms) | $2^8/f_{XP}$ (32 μs) |
| | $2^{10}/f_{XP}$ (128 μs) | $2^{18}/f_{XP}$ (32.7 ms) | $2^{10}/f_{XP}$ (128 μs) |
| | $2^{16}/f_{XP}$ (8.19 ms) | $2^{24}/f_{XP}$ (2.01 s) | $2^{16}/f_{XP}$ (8.19 ms) |
| $f_{XP}$ = 10.0 MHz | $2^6/f_{XP}$ (6.4 μs) | $2^{14}/f_{XP}$ (1.64 ms) | $2^6/f_{XP}$ (6.4 μs) |
| | $2^8/f_{XP}$ (25.6 μs) | $2^{16}/f_{XP}$ (6.55 ms) | $2^8/f_{XP}$ (25.6 μs) |
| | $2^{10}/f_{XP}$ (102 μs) | $2^{18}/f_{XP}$ (26.2 ms) | $2^{10}/f_{XP}$ (102 μs) |
| | $2^{16}/f_{XP}$ (6.55 ms) | $2^{24}/f_{XP}$ (1.68 s) | $2^{16}/f_{XP}$ (6.55 ms) |

**Remark** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

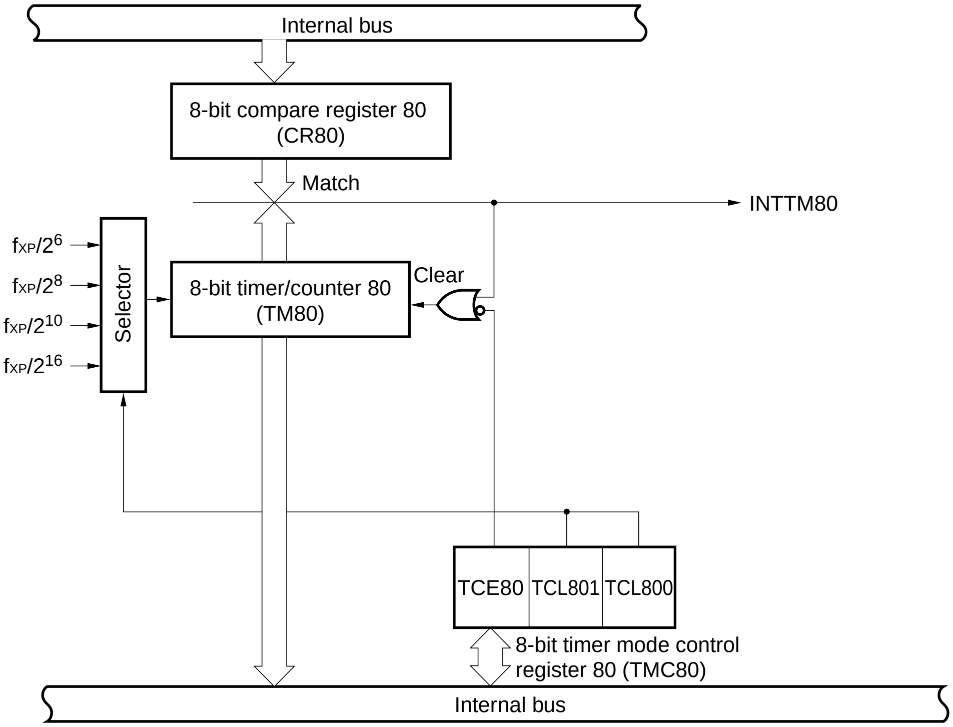
## 7.2 Configuration of 8-Bit Timer 80

8-bit timer 80 consists of the following hardware.

Table 7-2. Configuration of 8-Bit Timer 80

| Item | Configuration |
|---|---|
| Timer counter | 8-bit timer counter 80 (TM80) |
| Register | 8-bit compare register 80 (CR80) |
| Control register | 8-bit timer mode control register 80 (TMC80) |

Figure 7-1. Block Diagram of 8-Bit Timer 80



**Remark** fXP: Oscillation frequency of clock to peripheral hardware

### (1) 8-bit compare register 80 (CR80)

This 8-bit register always compares its set value with the count value of 8-bit timer/counter 80 (TM80). It generates an interrupt request signal (INTTM80) if the two values match.

CR80 is set by using an 8-bit memory manipulation instruction. A value of 00H to FFH can be set to this register.

Reset signal generation makes the contents of this register undefined.

**Figure 7-2. Format of 8-Bit Compare Register 80 (CR80)**

Address: FFCDH After reset: Undefined W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CR80 |  |  |  |  |  |  |  |  |

**Caution When changing the value of CR80, be sure to stop the timer operation. If the value of CR80 is changed with the timer operation enabled, a match interrupt request signal may be generated immediately.**

### (2) 8-bit timer counter 80 (TM80)

This 8-bit register counts the count pulses.

The value of TM80 can be read by using an 8-bit memory manipulation instruction.

Reset signal generation clears TM80 to 00H.

**Figure 7-3. Format of 8-Bit Timer Counter 80 (TM80)**

Address: FFCEH After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TM80 |  |  |  |  |  |  |  |  |

## 7.3 Register Controlling 8-Bit Timer 80

8-bit timer 80 is controlled by 8-bit timer mode control register 80 (TMC80).

### (1) 8-bit timer mode control register 80 (TMC80)

This register is used to enable or stop the operation of 8-bit timer/counter 80 (TM80), and to set the count clock of TM80.

This register is set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears TMC80 to 00H.

**Figure 7-4. Format of 8-Bit Timer Mode Control Register 80 (TMC80)**

Address: FFCCH After reset: 00H R/W

| Symbol | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TMC80 | TCE80 | 0 | 0 | 0 | 0 | TCL801 | TCL800 | 0 |

| TCE80 | Control of operation of TM80 |
|---|---|
| 0 | Stop operation (clear TM80 to 00H). |
| 1 | Enable operation. |

| TCL801 | TCL800 | Selection of count clock of 8-bit timer 80 | | |
|---|---|---|---|---|
| | | | $f_{XP}$ = 8.0 MHz | $f_{XP}$ = 10.0 MHz |
| 0 | 0 | $f_{XP}/2^6$ | 125 kHz | 156.3 kHz |
| 0 | 1 | $f_{XP}/2^8$ | 31.25 kHz | 39.06 kHz |
| 1 | 0 | $f_{XP}/2^{10}$ | 7.81 kHz | 9.77 kHz |
| 1 | 1 | $f_{XP}/2^{16}$ | 0.12 kHz | 0.15 kHz |

**Cautions 1. Be sure to set TMC80 after stopping the timer operation.**

**2. Be sure to clear bits 0 and 6 to 0.**

**Remark** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

## 7.4 Operation of 8-Bit Timer 80

### 7.4.1 Operation as interval timer

When 8-bit timer 80 operates as an interval timer, it can repeatedly generate an interrupt at intervals specified by the count value set in advance to 8-bit compare register 80 (CR80).

To use 8-bit timer 80 as an interval timer, make the following setting.

<1> Disable the operation of 8-bit timer counter 80 (clear TCE80 (bit 7 of 8-bit timer mode control register 80 (TMC80)) to 0).

<2> Set the count clock of 8-bit timer 80 (refer to **Tables 7-3** and **7-4**).

<3> Set the count value to CR80.

<4> Enable the operation of TM80 (set TCE80 to 1).

When the count value of 8-bit timer counter 80 (TM80) matches the set value of CR80, the value of TM80 is cleared to 00H and counting is continued. At the same time, an interrupt request signal (INTTM80) is generated.

Tables 7-3 and 7-4 show the interval time, and Figure 7-5 shows the timing of the interval timer operation.

**Cautions 1. When changing the value of CR80, be sure to stop the timer operation. If the value of CR80 is changed with the timer operation enabled, a match interrupt request signal may be generated immediately.**

**2. If the count clock of TMC80 is set and the operation of TM80 is enabled at the same time by using an 8-bit memory manipulation instruction, the error of one cycle after the timer is started may be 1 clock or more. Therefore, be sure to follow the above sequence when using TM80 as an interval timer.**

**Table 7-3. Interval Time of 8-Bit Timer 80 ($f_{XP}$ = 8.0 MHz)**

| TCL801 | TCL800 | Minimum Interval Time | Maximum Interval Time | Resolution |
|---|---|---|---|---|
| 0 | 0 | $2^6/f_{XP}$ (8 μs) | $2^{14}/f_{XP}$ (2.05 ms) | $2^6/f_{XP}$ (8 μs) |
| 0 | 1 | $2^8/f_{XP}$ (32 μs) | $2^{16}/f_{XP}$ (8.19 ms) | $2^8/f_{XP}$ (32 μs) |
| 1 | 0 | $2^{10}/f_{XP}$ (128 μs) | $2^{18}/f_{XP}$ (32.7 ms) | $2^{10}/f_{XP}$ (128 μs) |
| 1 | 1 | $2^{16}/f_{XP}$ (8.19 ms) | $2^{24}/f_{XP}$ (2.01 s) | $2^{16}/f_{XP}$ (8.19 ms) |

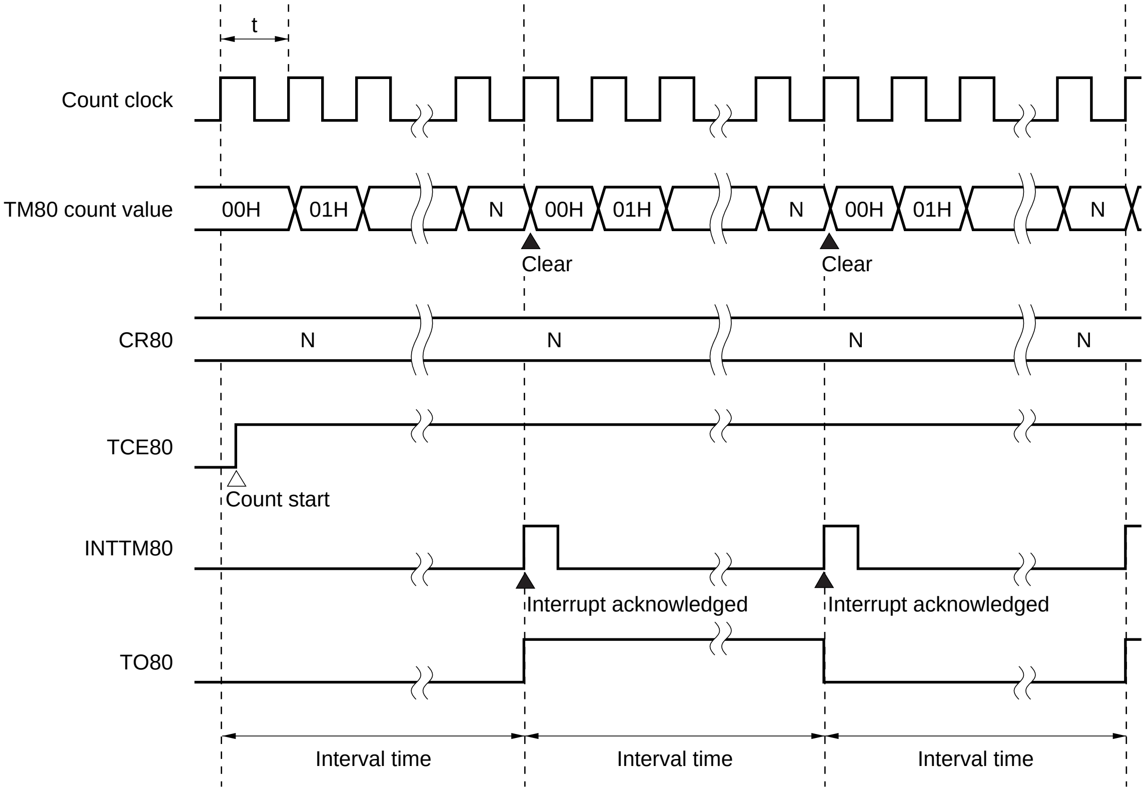
**Remark** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

**Table 7-4. Interval Time of 8-Bit Timer 80 ($f_{XP}$ = 10.0 MHz)**

| TCL801 | TCL800 | Minimum Interval Time | Maximum Interval Time | Resolution |
|---|---|---|---|---|
| 0 | 0 | $2^6/f_{XP}$ (6.4 μs) | $2^{14}/f_{XP}$ (1.64 ms) | $2^6/f_{XP}$ (6.4 μs) |
| 0 | 1 | $2^8/f_{XP}$ (25.6 μs) | $2^{16}/f_{XP}$ (6.55 ms) | $2^8/f_{XP}$ (25.6 μs) |
| 1 | 0 | $2^{10}/f_{XP}$ (102 μs) | $2^{18}/f_{XP}$ (26.2 ms) | $2^{10}/f_{XP}$ (102 μs) |
| 1 | 1 | $2^{16}/f_{XP}$ (6.55 ms) | $2^{24}/f_{XP}$ (1.68 s) | $2^{16}/f_{XP}$ (6.55 ms) |

**Remark** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

**Figure 7-5. Timing of Interval Timer Operation**



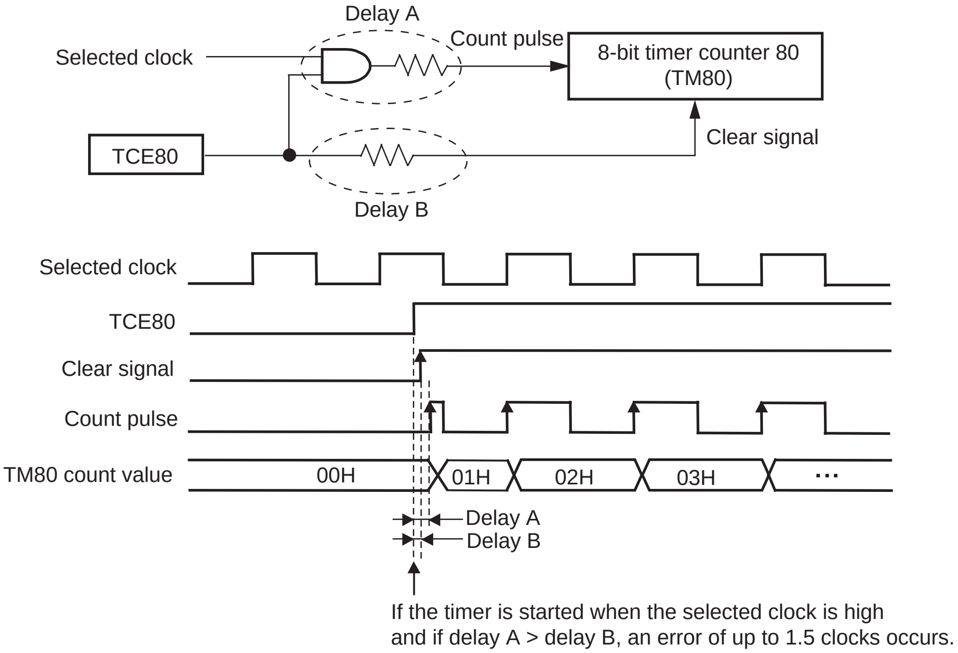
**Remark** Interval time = (N + 1) × t: N = 00H to FFH

## 7.5 Notes on 8-Bit Timer 80

### (1) Error when timer starts

The time from starting the timer to generation of the match signal includes an error of up to 1.5 clocks. This is because, if the timer is started while the count clock is high, the rising edge may be immediately detected and the counter may be incremented (refer to **Figure 7-6**).

**Figure 7-6. Case Where Error of 1.5 Clocks (Max.) Occurs**



### (2) Setting of 8-bit compare register 80

8-bit compare register 80 (CR80) can be set to 00H.

### (3) Note on setting STOP mode

Before executing the STOP instruction, be sure to stop the timer operation (TCE80 = 0).

# CHAPTER 8 8-BIT TIMER H1

## 8.1 Functions of 8-Bit Timer H1

8-bit timer H1 has the following functions.

- Interval timer
- PWM output mode
- Square-wave output

## 8.2 Configuration of 8-Bit Timer H1

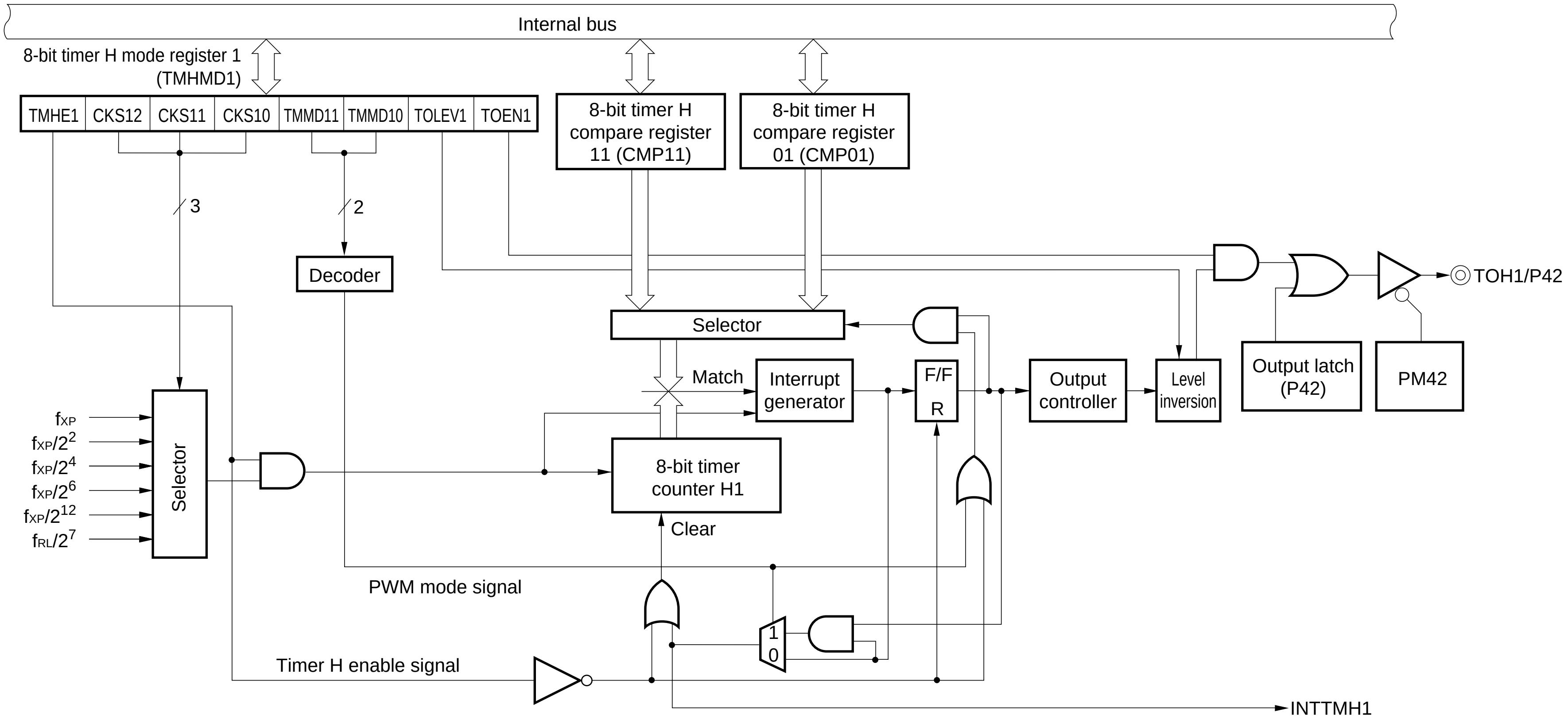
8-bit timer H1 consists of the following hardware.

**Table 8-1. Configuration of 8-Bit Timer H1**

| Item | Configuration |
|---|---|
| Timer register | 8-bit timer counter H1 |
| Registers | 8-bit timer H compare register 01 (CMP01)<br>8-bit timer H compare register 11 (CMP11) |
| Timer output | TOH1 |
| Control registers | 8-bit timer H mode register 1 (TMHMD1)<br>Port mode register 4 (PM4)<br>Port register 4 (P4) |

Figure 8-1 shows a block diagram.

**Figure 8-1. Block Diagram of 8-Bit Timer H1**



**(1) 8-bit timer H compare register 01 (CMP01)**

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

Reset signal generation clears this register to 00H.

**Figure 8-2. Format of 8-Bit Timer H Compare Register 01 (CMP01)**

Address: FF0EH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CMP01 | | | | | | | | |

**Caution CMP01 cannot be rewritten during timer count operation.**

**(2) 8-bit timer H compare register 11 (CMP11)**

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

Reset signal generation clears this register to 00H.

**Figure 8-3. Format of 8-Bit Timer H Compare Register 11 (CMP11)**

Address: FF0FH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CMP11 | | | | | | | | |

CMP11 can be rewritten during timer count operation.

If the CMP11 value is rewritten during timer operation, transferring is performed at the timing at which the count value and CMP11 value match. If the transfer timing and writing from CPU to CMP11 conflict, transfer is not performed.

**Caution In the PWM output mode, be sure to set CMP11 when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to CMP11).**

## 8.3 Registers Controlling 8-Bit Timer H1

The following three registers are used to control 8-Bit Timer H1.

- 8-bit timer H mode register 1 (TMHMD1)
- Port mode register 4 (PM4)
- Port register 4 (P4)

**(1) 8-bit timer H mode register 1 (TMHMD1)**

This register controls the mode of timer H.

This register can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Figure 8-4. Format of 8-Bit Timer H Mode Register 1 (TMHMD1)**

Address: FF70H After reset: 00H R/W

| Symbol | <7> | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| TMHMD1 | TMHE1 | CKS12 | CKS11 | CKS10 | TMMD11 | TMMD10 | TOLEV1 | TOEN1 |

| TMHE1 | Timer operation enable |
|---|---|
| 0 | Stop timer count operation (counter is cleared to 0) |
| 1 | Enable timer count operation (count operation started by inputting clock) |

| CKS12 | CKS11 | CKS10 | Count clock ($f_{CNT}$) selection |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XP}$ (10 MHz) |
| 0 | 0 | 1 | $f_{XP}/2^2$ (2.5 MHz) |
| 0 | 1 | 0 | $f_{XP}/2^4$ (625 kHz) |
| 0 | 1 | 1 | $f_{XP}/2^6$ (156.25 kHz) |
| 1 | 0 | 0 | $f_{XP}/2^{12}$ (2.44 kHz) |
| 1 | 0 | 1 | $f_{RL}/2^7$ (1.88 kHz (TYP.)) |
| Other than above | | | Setting prohibited |

| TMMD11 | TMMD10 | Timer operation mode |
|---|---|---|
| 0 | 0 | Interval timer mode |
| 1 | 0 | PWM output mode |
| Other than above | | Setting prohibited |

| TOLEV1 | Timer output level control (in default mode) |
|---|---|
| 0 | Low level |
| 1 | High level |

| TOEN1 | Timer output control |
|---|---|
| 0 | Disable output |
| 1 | Enable output |

**Cautions 1. When TMHE1 = 1, setting the other bits of the TMHMD1 register is prohibited.**

**2. In the PWM output mode, be sure to set 8-bit timer H compare register 11 (CMP11) when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).**

**Remarks 1.** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

**2.** $f_{RL}$: Low-speed Ring-OSC clock oscillation frequency

**3.** Figures in parentheses apply to operation at $f_{XP}$ = 10 MHz, $f_{RL}$ = 240 kHz (TYP.).

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

This register sets port 4 input/output in 1-bit units.

When using the P42/TOH1 pin for timer output, clear PM42 and the output latch of P42 to 0.

PM4 can be set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets this register to FFH.

**Figure 8-5. Format of Port Mode Register 4 (PM4)**

Address: FF24H After reset: FFH R/W

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

| PM4n | P4n pin I/O mode selection (n = 0 to 7) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

## 8.4 Operation of 8-Bit Timer H1

### 8.4.1 Operation as interval timer/square-wave output

When 8-bit timer counter H1 and compare register 01 (CMP01) match, an interrupt request signal (INTTMH1) is generated and 8-bit timer counter H1 is cleared to 00H.

Compare register 11 (CMP11) is not used in interval timer mode. Since a match of 8-bit timer counter H1 and the CMP11 register is not detected even if the CMP11 register is set, timer output is not affected.

By setting bit 0 (TOEN1) of timer H mode register 1 (TMHMD1) to 1, a square wave of any frequency (duty = 50%) is output from TOH1.

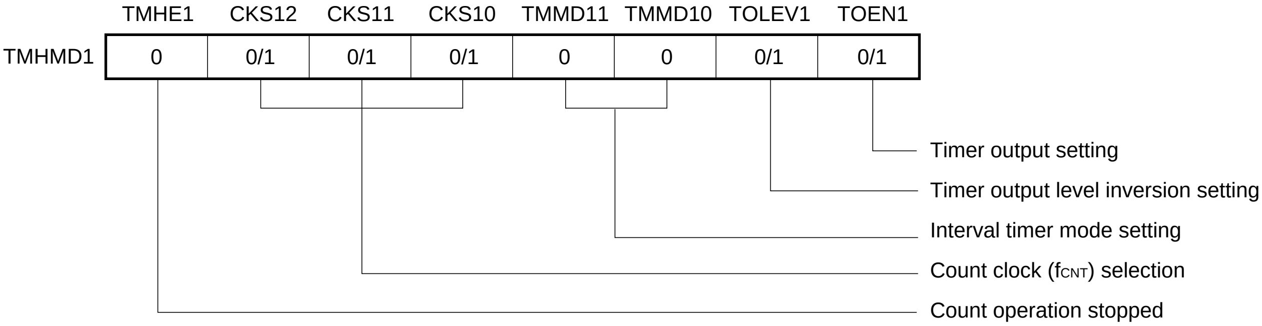
**(1) Usage**

Generates the INTTMH1 signal repeatedly at the same interval.

<1> Set each register.

**Figure 8-6. Register Setting During Interval Timer/Square-Wave Output Operation**

**(i) Setting timer H mode register 1 (TMHMD1)**



| | TMHE1 | CKS12 | CKS11 | CKS10 | TMMD11 | TMMD10 | TOLEV1 | TOEN1 |
|---|---|---|---|---|---|---|---|---|
| TMHMD1 | 0 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0/1 | 0/1 |

Timer output setting
Timer output level inversion setting
Interval timer mode setting
Count clock ($f_{CNT}$) selection
Count operation stopped

**(ii) CMP01 register setting**

- Compare value (N)

<2> Count operation starts when TMHE1 = 1.

<3> When the values of 8-bit timer counter H1 and the CMP01 register match, the INTTMH1 signal is generated and 8-bit timer counter H1 is cleared to 00H.

$$\boxed{\text{Interval time} = (N + 1)/f_{CNT}}$$

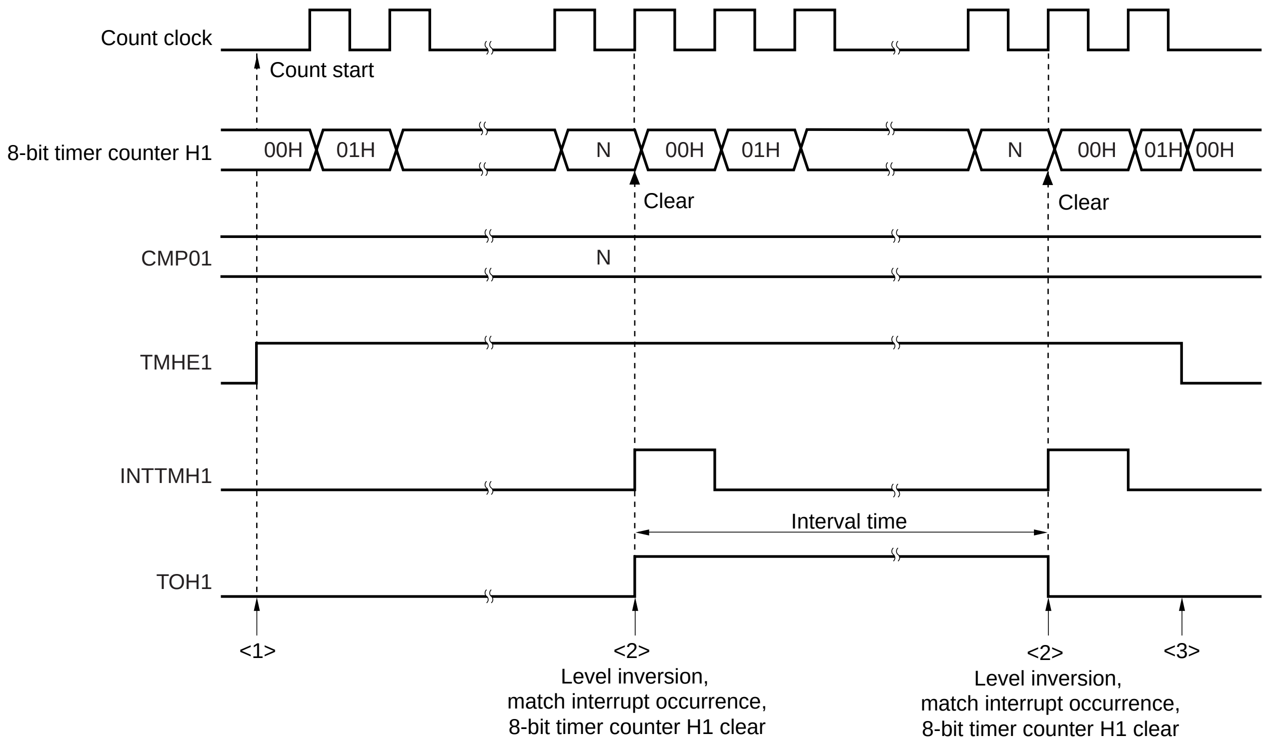
<4> Subsequently, the INTTMH1 signal is generated at the same interval. To stop the count operation, clear TMHE1 to 0.

### (2) Timing chart

The timing of the interval timer/square-wave output operation is shown below.

**Figure 8-7. Timing of Interval Timer/Square-Wave Output Operation (1/2)**

**(a) Basic operation**



<1> The count operation is enabled by setting the TMHE1 bit to 1. The count clock starts counting no more than 1 clock after the operation is enabled.

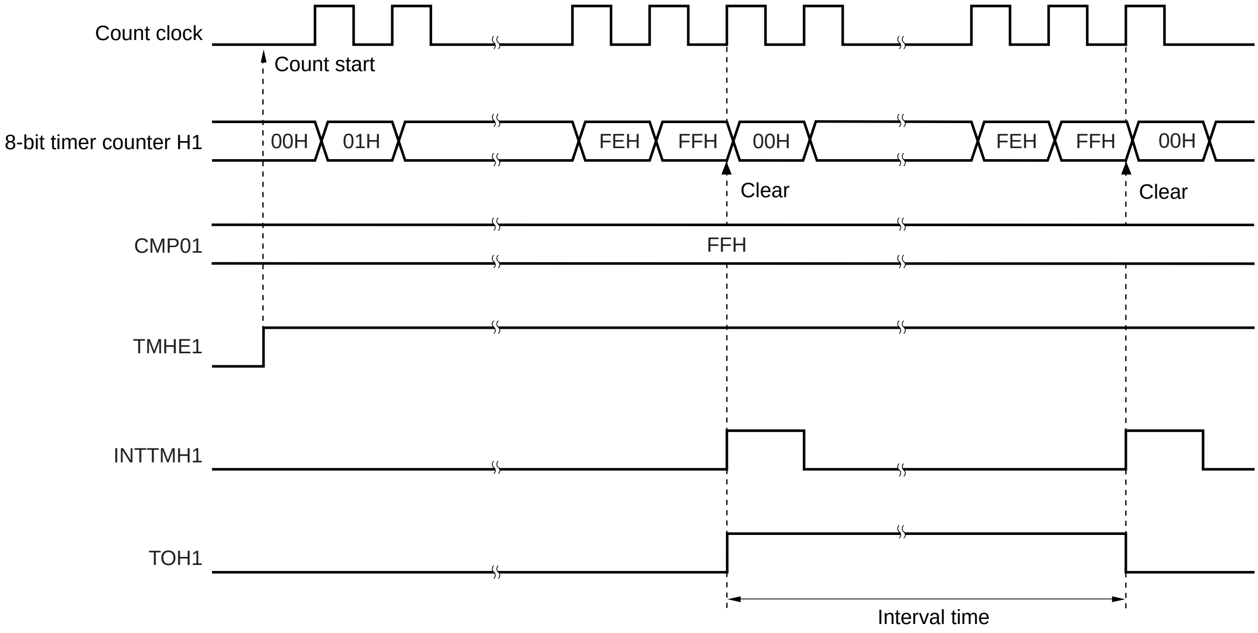
<2> When the values of 8-bit timer counter H1 and the CMP01 register match, the value of 8-bit timer counter H1 is cleared, the TOH1 output level is inverted, and the INTTMH1 signal is output.

<3> The INTTMH1 signal and TOH1 output become inactive by clearing the TMHE1 bit to 0 during timer H1 operation. If these are inactive from the first, the level is retained.

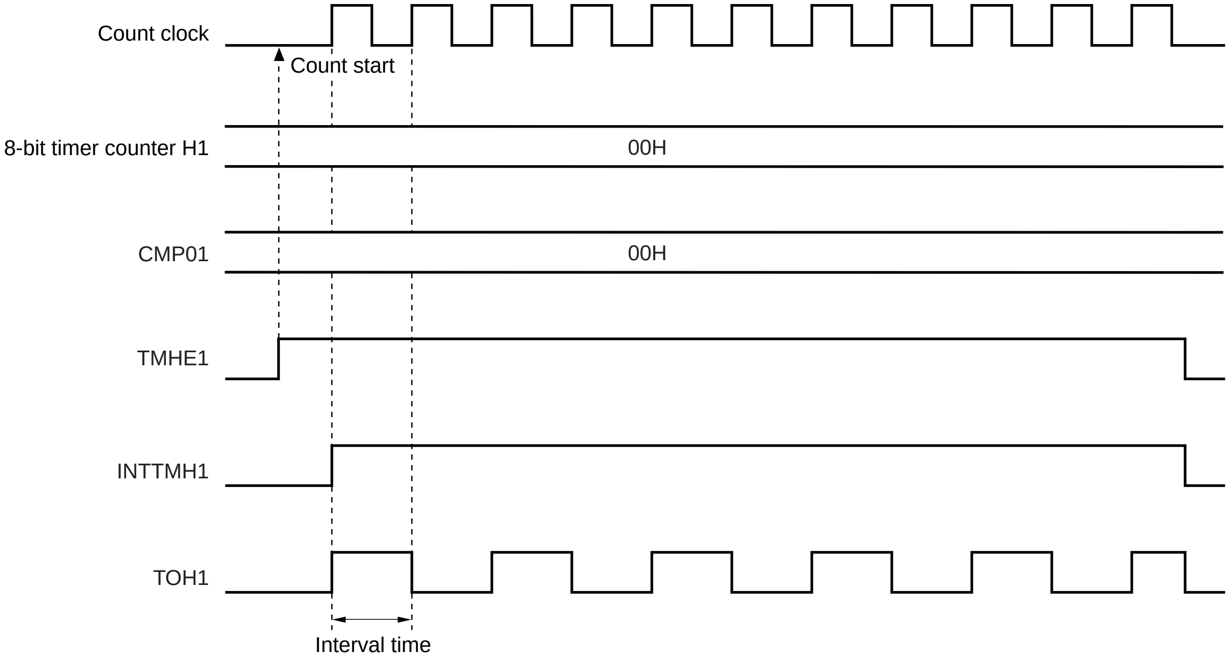
**Remark** N = 01H to FEH

Figure 8-7. Timing of Interval Timer/Square-Wave Output Operation (2/2)

(b) Operation when CMP01 = FFH



(c) Operation when CMP01 = 00H



### 8.4.2 Operation as PWM output mode

In PWM output mode, a pulse with an arbitrary duty and arbitrary cycle can be output.

8-bit timer compare register 01 (CMP01) controls the cycle of timer output (TOH1). Rewriting the CMP01 register during timer operation is prohibited.

8-bit timer compare register 11 (CMP11) controls the duty of timer output (TOH1). Rewriting the CMP11 register during timer operation is possible.

The operation in PWM output mode is as follows.

TOH1 output becomes active and 8-bit timer counter H1 is cleared to 0 when 8-bit timer counter H1 and the CMP01 register match after the timer count is started. TOH1 output becomes inactive when 8-bit timer counter H1 and the CMP11 register match.

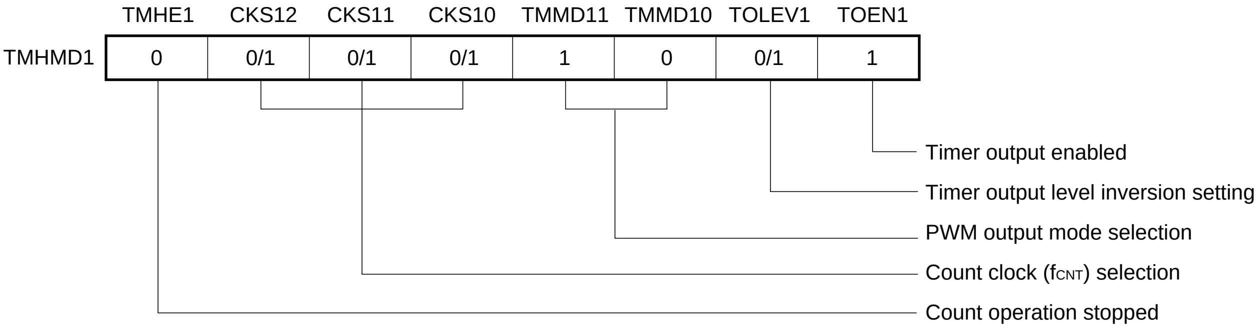
#### (1) Usage

In PWM output mode, a pulse for which an arbitrary duty and arbitrary cycle can be set is output.

<1> Set each register.

**Figure 8-8. Register Setting in PWM Output Mode**

**(i) Setting timer H mode register 1 (TMHMD1)**



| | TMHE1 | CKS12 | CKS11 | CKS10 | TMMD11 | TMMD10 | TOLEV1 | TOEN1 |
|---|---|---|---|---|---|---|---|---|
| TMHMD1 | 0 | 0/1 | 0/1 | 0/1 | 1 | 0 | 0/1 | 1 |

Timer output enabled
Timer output level inversion setting
PWM output mode selection
Count clock ($f_{CNT}$) selection
Count operation stopped

**(ii) Setting CMP01 register**

- Compare value (N): Cycle setting

**(iii) Setting CMP11 register**

- Compare value (M): Duty setting

**Remark** 00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH

<2> The count operation starts when TMHE1 = 1.

<3> The CMP01 register is the compare register that is to be compared first after count operation is enabled. When the values of 8-bit timer counter H1 and the CMP01 register match, 8-bit timer counter H1 is cleared, an interrupt request signal (INTTMH1) is generated, and TOH1 output becomes active. At the same time, the compare register to be compared with 8-bit timer counter H1 is changed from the CMP01 register to the CMP11 register.

<4> When 8-bit timer counter H1 and the CMP11 register match, TOH1 output becomes inactive and the compare register to be compared with 8-bit timer counter H1 is changed from the CMP11 register to the CMP01 register. At this time, 8-bit timer counter H1 is not cleared and the INTTMH1 signal is not generated.

<5> By performing procedures <3> and <4> repeatedly, a pulse with an arbitrary duty can be obtained.

<6> To stop the count operation, set TMHE1 = 0.

If the setting value of the CMP01 register is N, the setting value of the CMP11 register is M, and the count clock frequency is $f_{CNT}$, the PWM pulse output cycle and duty are as follows.

PWM pulse output cycle = $(N+1)/f_{CNT}$
Duty = Active width : Total width of PWM = (M + 1) : (N + 1)

**Cautions 1. In PWM output mode, three operation clocks (signal selected using the CKS12 to CKS10 bits of the TMHMD1 register) are required to transfer the CMP11 register value after rewriting the register.**

**2. Be sure to set the CMP11 register when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).**

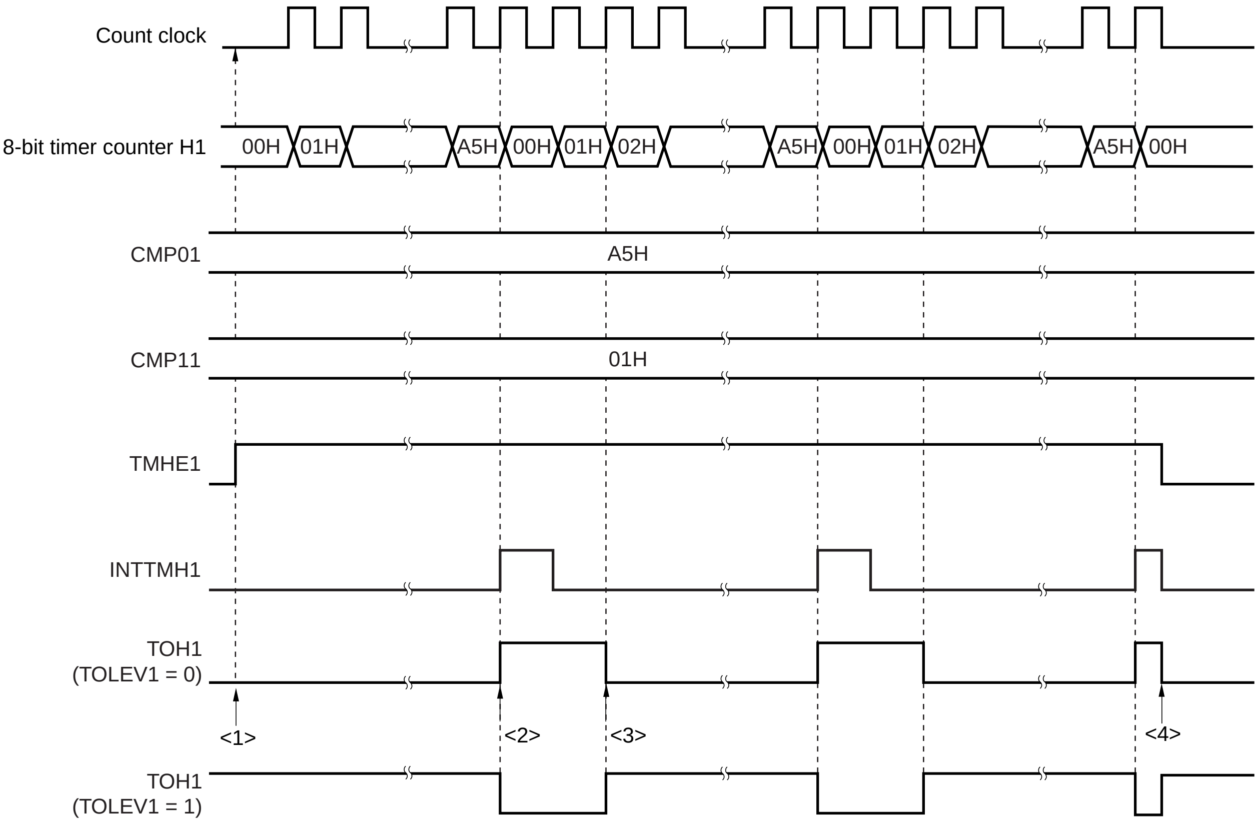
### (2) Timing chart

The operation timing in PWM output mode is shown below.

**Caution Make sure that the CMP11 register setting value (M) and CMP01 register setting value (N) are within the following range.**
**00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH**

**Figure 8-9. Operation Timing in PWM Output Mode (1/4)**

**(a) Basic operation**



<1> The count operation is enabled by setting the TMHE1 bit to 1. Start 8-bit timer counter H1 by masking one count clock to count up. At this time, TOH1 output remains inactive (when TOLEV1 = 0).

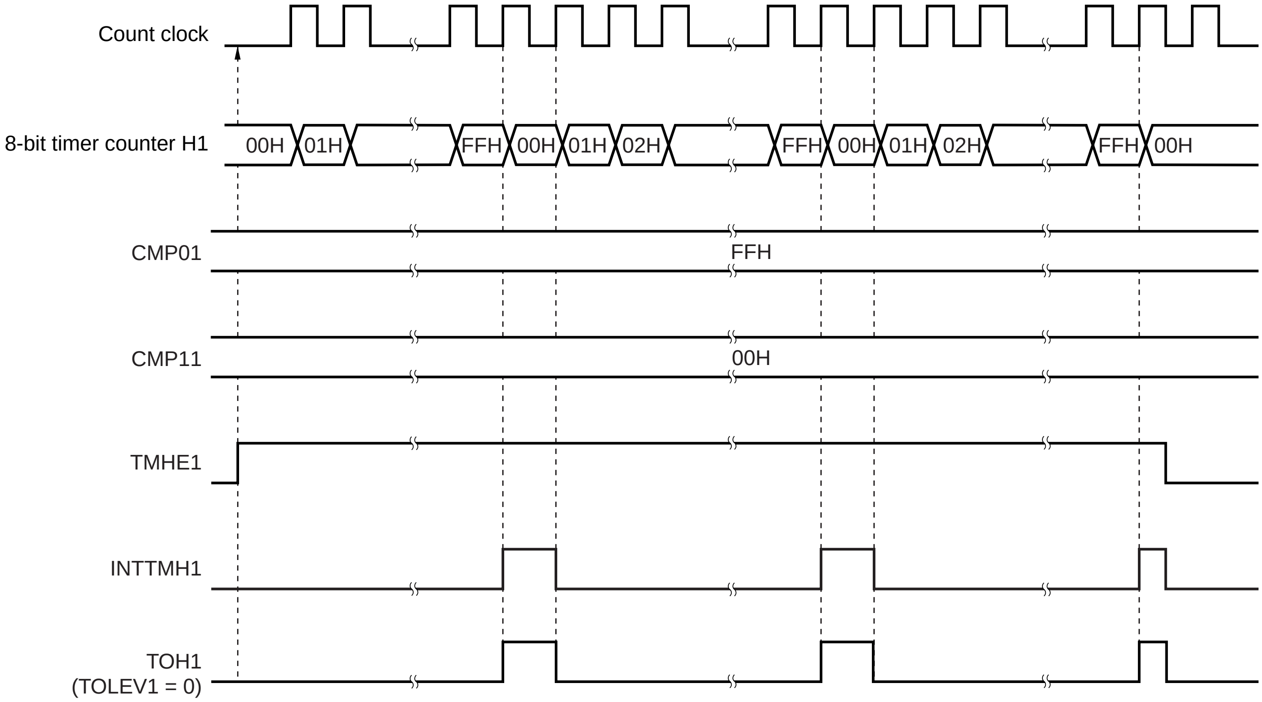
<2> When the values of 8-bit timer counter H1 and the CMP01 register match, the TOH1 output level is inverted, the value of 8-bit timer counter H1 is cleared, and the INTTMH1 signal is output.

<3> When the values of 8-bit timer counter H1 and the CMP11 register match, the level of the TOH1 output is returned. At this time, the 8-bit timer counter value is not cleared and the INTTMH1 signal is not output.

<4> Clearing the TMHE1 bit to 0 during timer H1 operation makes the INTTMH1 signal and TOH1 output inactive.

### Figure 8-9. Operation Timing in PWM Output Mode (2/4)

### (b) Operation when CMP01 = FFH, CMP11 = 00H



### (c) Operation when CMP01 = FFH, CMP11 = FEH

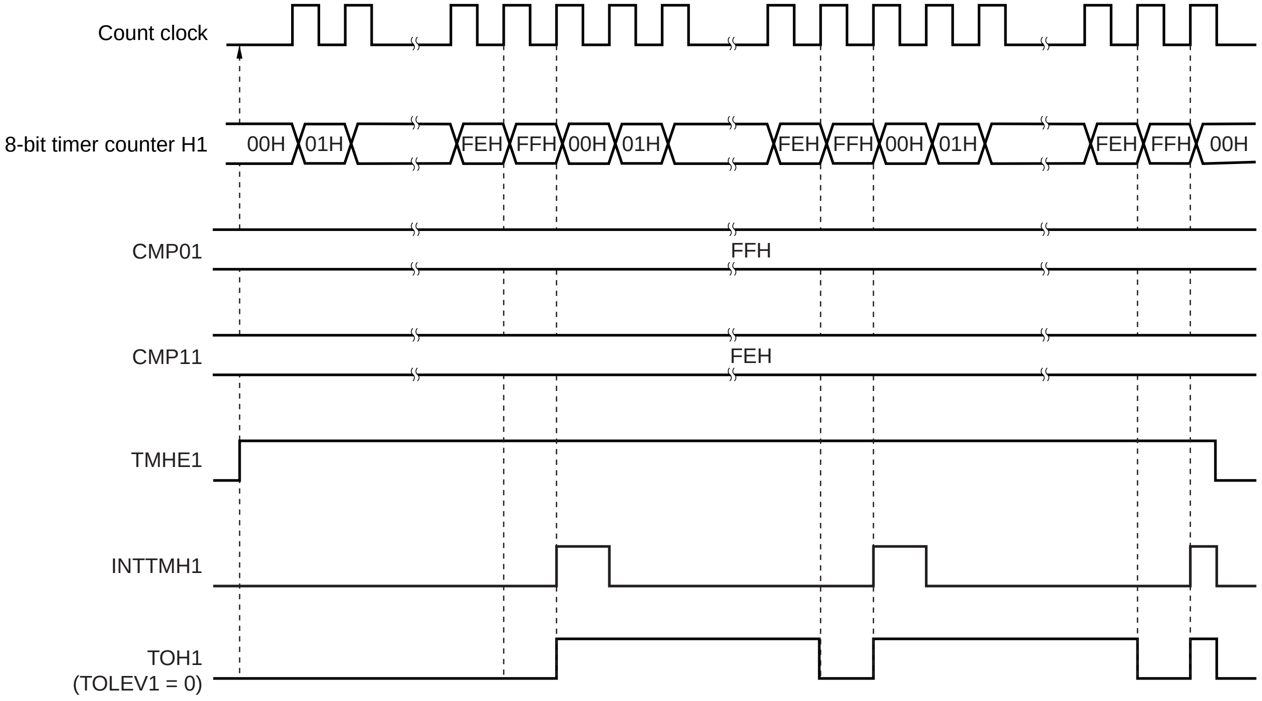


**Figure 8-9. Operation Timing in PWM Output Mode (3/4)**

**(d) Operation when CMP01 = 01H, CMP11 = 00H**

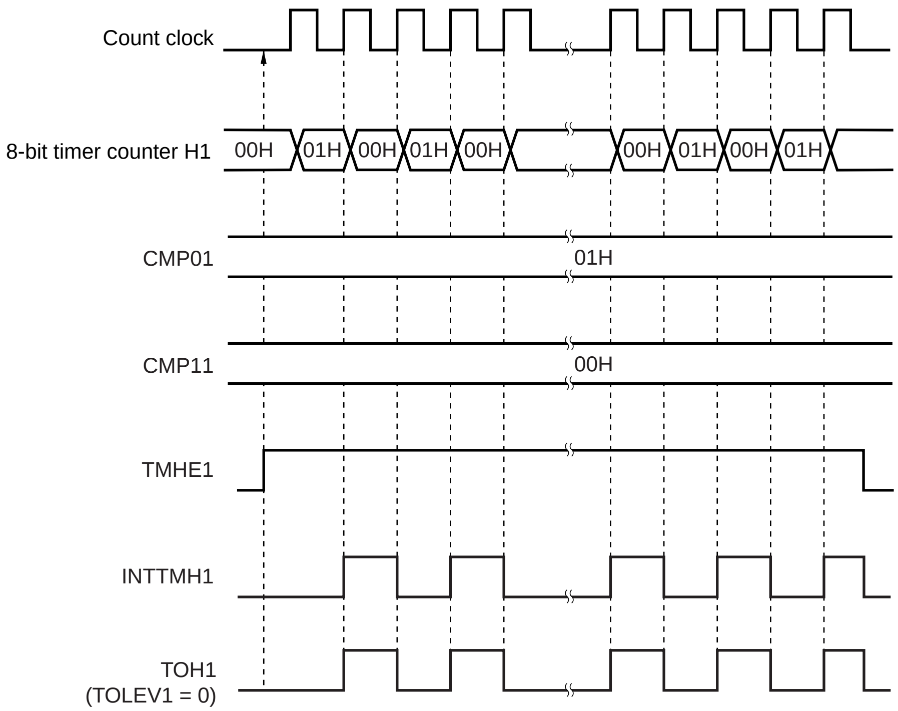
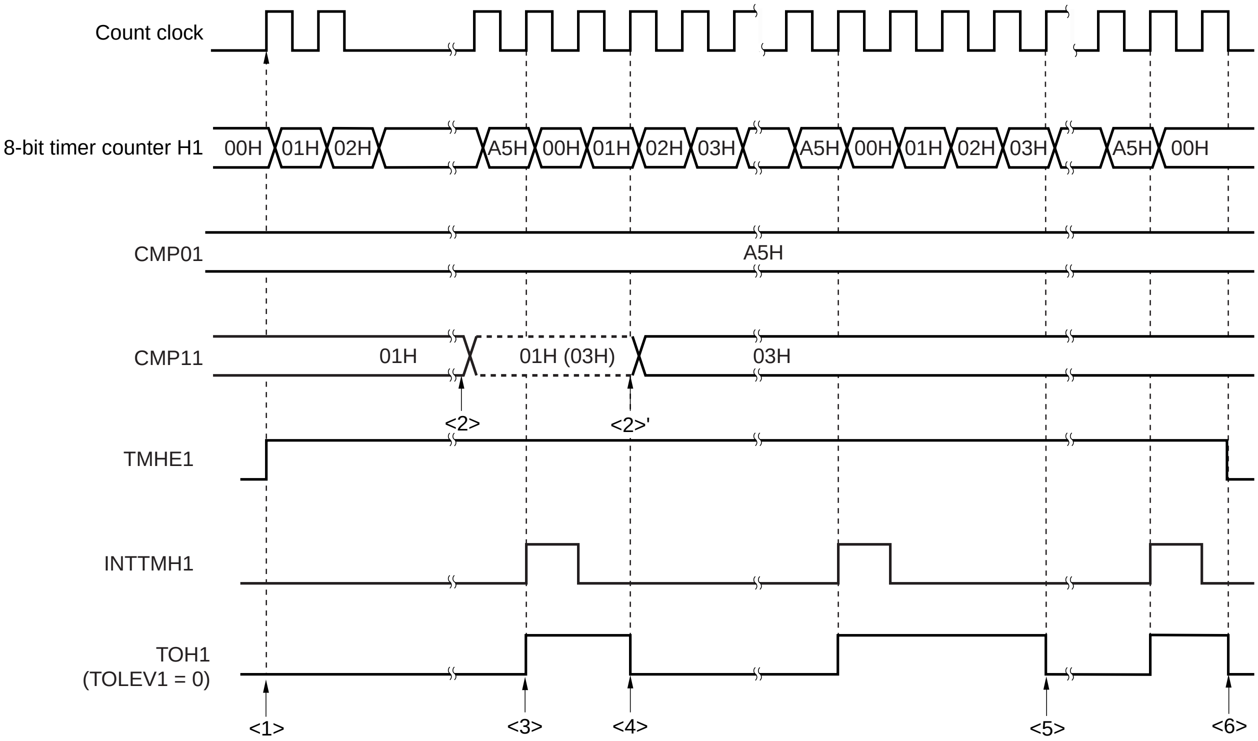


**Figure 8-9. Operation Timing in PWM Output Mode (4/4)**

**(e) Operation by changing CMP11 (CMP11 = 01H → 03H, CMP01 = A5H)**



<1> The count operation is enabled by setting TMHE1 = 1. Start 8-bit timer counter H1 by masking one count clock to count up. At this time, the TOH1 output remains inactive (when TOLEV1 = 0).

<2> The CMP11 register value can be changed during timer counter operation. This operation is asynchronous to the count clock.

<3> When the values of 8-bit timer counter H1 and the CMP01 register match, the value of 8-bit timer counter H1 is cleared, the TOH1 output becomes active, and the INTTMH1 signal is output.

<4> If the CMP11 register value is changed, the value is latched and not transferred to the register. When the values of 8-bit timer counter H1 and the CMP11 register before the change match, the value is transferred to the CMP11 register and the CMP11 register value is changed (<2>’).

However, three count clocks or more are required from when the CMP11 register value is changed to when the value is transferred to the register. If a match signal is generated within three count clocks, the changed value cannot be transferred to the register.

<5> When the values of 8-bit timer counter H1 and the CMP11 register after the change match, the TOH1 output becomes inactive. 8-bit timer counter H1 is not cleared and the INTTMH1 signal is not generated.

<6> Clearing the TMHE1 bit to 0 during timer H1 operation makes the INTTMH1 signal and TOH1 output inactive.

# CHAPTER 9 WATCHDOG TIMER

## 9.1 Functions of Watchdog Timer

The watchdog timer is used to detect an inadvertent program loop. If a program loop is detected, an internal reset signal is generated.

When a reset occurs due to the watchdog timer, bit 4 (WDTRF) of the reset control flag register (RESF) is set to 1. For details of RESF, see **CHAPTER 15 RESET FUNCTION**.

**Table 9-1. Loop Detection Time of Watchdog Timer**

| Loop Detection Time | |
|---|---|
| During Low-Speed Ring-OSC Clock Operation | During System Clock Operation |
| $2^{11}/f_{RL}$ (4.27 ms) | $2^{13}/f_X$ (819.2 $\mu$s) |
| $2^{12}/f_{RL}$ (8.53 ms) | $2^{14}/f_X$ (1.64 ms) |
| $2^{13}/f_{RL}$ (17.07 ms) | $2^{15}/f_X$ (3.28 ms) |
| $2^{14}/f_{RL}$ (34.13 ms) | $2^{16}/f_X$ (6.55 ms) |
| $2^{15}/f_{RL}$ (68.27 ms) | $2^{17}/f_X$ (13.11 ms) |
| $2^{16}/f_{RL}$ (136.53 ms) | $2^{18}/f_X$ (26.21 ms) |
| $2^{17}/f_{RL}$ (273.07 ms) | $2^{19}/f_X$ (52.43 ms) |
| $2^{18}/f_{RL}$ (546.13 ms) | $2^{20}/f_X$ (104.86 ms) |

**Remarks 1.** $f_{RL}$: Low-speed Ring-OSC clock oscillation frequency

**2.** $f_X$: Oscillation frequency of system clock

**3.** Figures in parentheses apply to operation at $f_{RL}$ = 480 kHz (MAX.), $f_X$ = 10 MHz.

The operation mode of the watchdog timer (WDT) is switched according to the option byte setting of the on-chip low-speed Ring-OSC oscillator as shown in Table 9-2.

**Table 9-2. Option Byte Setting and Watchdog Timer Operation Mode**

| | Option Byte Setting | |
|---|---|---|
| | Low-Speed Ring-OSC Cannot Be Stopped | Low-Speed Ring-OSC Can Be Stopped by Software |
| Watchdog timer clock source | Fixed to $f_{RL}$[Note 1]. | • Selectable by software ($f_X$, $f_{RL}$ or stopped)<br>• When reset is released: $f_{RL}$ |
| Operation after reset | Operation starts with the maximum interval ($2^{18}/f_{RL}$). | Operation starts with the maximum interval ($2^{18}/f_{RL}$). |
| Operation mode selection | The interval can be changed only once. | The clock selection/interval can be changed only once. |
| Features | The watchdog timer cannot be stopped. | The watchdog timer can be stopped[Note 2]. |

**Notes 1.** As long as power is being supplied, low-speed Ring-OSC oscillation cannot be stopped (except in the reset period).

**2.** The conditions under which clock supply to the watchdog timer is stopped differ depending on the clock source of the watchdog timer.

<1> If the clock source is $f_X$, clock supply to the watchdog timer is stopped under the following conditions.

- When $f_X$ is stopped
- In HALT/STOP mode
- During oscillation stabilization time

<2> If the clock source is $f_{RL}$, clock supply to the watchdog timer is stopped under the following conditions.

- If the CPU clock is $f_X$ and if $f_{RL}$ is stopped by software before execution of the STOP instruction
- In HALT/STOP mode

**Remarks 1.** $f_{RL}$: Low-speed Ring-OSC clock oscillation frequency

**2.** $f_X$: Oscillation frequency of system clock

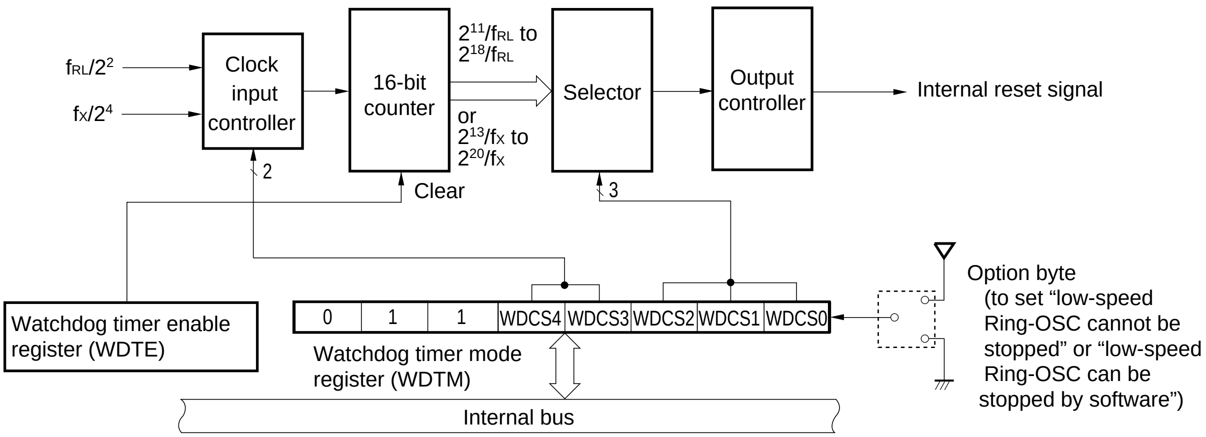
## 9.2 Configuration of Watchdog Timer

The watchdog timer consists of the following hardware.

**Table 9-3. Configuration of Watchdog Timer**

| Item | Configuration |
|---|---|
| Control registers | Watchdog timer mode register (WDTM)<br>Watchdog timer enable register (WDTE) |

**Figure 9-1. Block Diagram of Watchdog Timer**



$f_{RL}/2^2$
$f_X/2^4$
Clock input controller
2
16-bit counter
Clear
$2^{11}/f_{RL}$ to $2^{18}/f_{RL}$
or
$2^{13}/f_X$ to $2^{20}/f_X$
Selector
3
Output controller
Internal reset signal
Watchdog timer enable register (WDTE)
0
1
1
WDCS4
WDCS3
WDCS2
WDCS1
WDCS0
Watchdog timer mode register (WDTM)
Internal bus
Option byte (to set “low-speed Ring-OSC cannot be stopped” or “low-speed Ring-OSC can be stopped by software”)

## 9.3 Registers Controlling Watchdog Timer

The watchdog timer is controlled by the following two registers.

- Watchdog timer mode register (WDTM)
- Watchdog timer enable register (WDTE)

### (1) Watchdog timer mode register (WDTM)

This register sets the overflow time and operation clock of the watchdog timer.

This register can be set by an 8-bit memory manipulation instruction and can be read many times, but can be written only once after reset is released.

Generation of reset signal sets this register to 67H.

**Figure 9-2. Format of Watchdog Timer Mode Register (WDTM)**

Address: FF48H After reset: 67H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTM | 0 | 1 | 1 | WDCS4 | WDCS3 | WDCS2 | WDCS1 | WDCS0 |

| WDCS4[Note 1] | WDCS3[Note 1] | Operation clock selection |
|---|---|---|
| 0 | 0 | Low-speed Ring-OSC clock ($f_{RL}$) |
| 0 | 1 | System Clock ($f_X$) |
| 1 | × | Watchdog timer operation stopped |

| WDCS2[Note 2] | WDCS1[Note 2] | WDCS0[Note 2] | Overflow time setting | |
|---|---|---|---|---|
| | | | During low-speed Ring-OSC clock operation | During system clock operation |
| 0 | 0 | 0 | $2^{11}/f_{RL}$ (4.27 ms) | $2^{13}/f_X$ (819.2 $\mu$s) |
| 0 | 0 | 1 | $2^{12}/f_{RL}$ (8.53 ms) | $2^{14}/f_X$ (1.64 ms) |
| 0 | 1 | 0 | $2^{13}/f_{RL}$ (17.07 ms) | $2^{15}/f_X$ (3.28 ms) |
| 0 | 1 | 1 | $2^{14}/f_{RL}$ (34.13 ms) | $2^{16}/f_X$ (6.55 ms) |
| 1 | 0 | 0 | $2^{15}/f_{RL}$ (68.27 ms) | $2^{17}/f_X$ (13.11 ms) |
| 1 | 0 | 1 | $2^{16}/f_{RL}$ (136.53 ms) | $2^{18}/f_X$ (26.21 ms) |
| 1 | 1 | 0 | $2^{17}/f_{RL}$ (273.07 ms) | $2^{19}/f_X$ (52.43 ms) |
| 1 | 1 | 1 | $2^{18}/f_{RL}$ (546.13 ms) | $2^{20}/f_X$ (104.86 ms) |

**Notes 1.** If “low-speed Ring-OSC cannot be stopped” is specified by the option byte, this cannot be set. The low-speed Ring-OSC clock will be selected no matter what value is written.

**2.** Reset is released at the maximum cycle (WDCS2, 1, 0 = 1, 1, 1).

**Cautions 1. Set bits 7, 6, and 5 to 0, 1, and 1, respectively (when “low-speed Ring-OSC cannot be stopped” is selected by the option byte, other values are ignored).**

**2. After reset is released, WDTM can be written only once by an 8-bit memory manipulation instruction. If writing is attempted a second time, an internal reset signal is generated.**

**3. WDTM cannot be set by a 1-bit memory manipulation instruction.**

**4. When using the flash memory self programming by self writing, set the overflow time for the watchdog timer so that enough everflow time is secured (Example 1-byte writing: 200 μs MIN., 1-block deletion: 10 ms MIN.).**

**Remarks 1.** $f_{RL}$: Low-speed Ring-OSC clock oscillation frequency

**2.** $f_X$: System clock oscillation frequency

**3.** ×: Don’t care

**4.** Figures in parentheses apply to operation at $f_{RL}$ = 480 kHz (MAX.), $f_X$ = 10 MHz.

### (2) Watchdog timer enable register (WDTE)

Writing ACH to WDTE clears the watchdog timer counter and starts counting again.

This register can be set by an 8-bit memory manipulation instruction.

Generation of reset signal sets this register to 9AH.

**Figure 9-3. Format of Watchdog Timer Enable Register (WDTE)**

Address: FF49H After reset: 9AH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTE | | | | | | | | |

**Cautions 1. If a value other than ACH is written to WDTE, an internal reset signal is generated.**

**2. If a 1-bit memory manipulation instruction is executed for WDTE, an internal reset signal is generated.**

**3. The value read from WDTE is 9AH (this differs from the written value (ACH)).**

## 9.4 Operation of Watchdog Timer

### 9.4.1 Watchdog timer operation when “low-speed Ring-OSC cannot be stopped” is selected by option byte

The operation clock of watchdog timer is fixed to low-speed Ring-OSC.

After reset is released, operation is started at the maximum cycle (bits 2, 1, and 0 (WDCS2, WDCS1, WDCS0) of the watchdog timer mode register (WDTM) = 1, 1, 1). The watchdog timer operation cannot be stopped.

The following shows the watchdog timer operation after reset release.

1. The status after reset release is as follows.
   - Operation clock: Low-speed Ring-OSC clock
   - Cycle: $2^{18}/f_{RL}$ (546.13 ms: operation with $f_{RL}$ = 480 kHz (MAX.))
   - Counting starts
2. The following should be set in the watchdog timer mode register (WDTM) by an 8-bit memory manipulation instruction[Notes 1, 2].
   - Cycle: Set using bits 2 to 0 (WDCS2 to WDCS0)
3. After the above procedures are executed, writing ACH to WDTE clears the count to 0, enabling recounting.

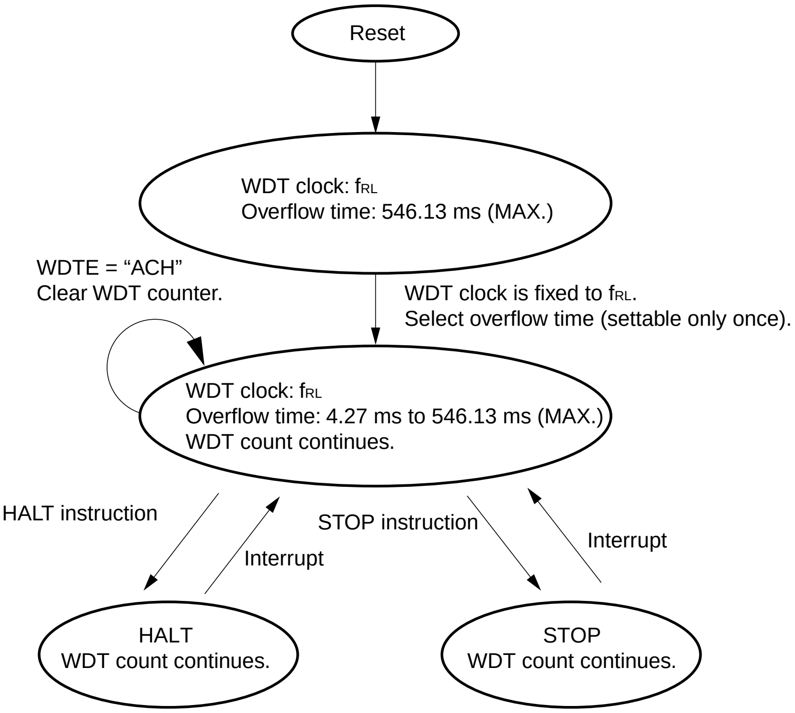
**Notes 1.** The operation clock (low-speed Ring-OSC clock) cannot be changed. If any value is written to bits 3 and 4 (WDCS3, WDCS4) of WDTM, it is ignored.

**2.** As soon as WDTM is written, the counter of the watchdog timer is cleared.

**Caution In this mode, operation of the watchdog timer cannot be stopped even during STOP instruction execution. For 8-bit timer H1 (TMH1), a division of the low-speed Ring-OSC clock can be selected as the count source, so clear the watchdog timer using the interrupt request of TMH1 before the watchdog timer overflows after STOP instruction execution. If this processing is not performed, an internal reset signal is generated when the watchdog timer overflows after STOP instruction execution.**

A status transition diagram is shown below

**Figure 9-4. Status Transition Diagram When “Low-Speed Ring-OSC Cannot Be Stopped” Is Selected by Option Byte**



### 9.4.2 Watchdog timer operation when "low-speed Ring-OSC can be stopped by software" is selected by option byte

The operation clock of the watchdog timer can be selected as either the low-speed Ring-OSC clock or the system clock.

After reset is released, operation is started at the maximum cycle of the low-speed Ring-OSC clock (bits 2, 1, and 0 (WDCS2, WDCS1, WDCS0) of the watchdog timer mode register (WDTM) = 1, 1, 1).

The following shows the watchdog timer operation after reset release.

1. The status after reset release is as follows.
   - Operation clock: Low-speed Ring-OSC clock
   - Cycle: $2^{18}/f_{RL}$ (546.13 ms: operation with $f_{RL}$ = 480 kHz (MAX.))
   - Counting starts
2. The following should be set in the watchdog timer mode register (WDTM) by an 8-bit memory manipulation instruction**Notes 1, 2, 3**.
   - Operation clock: Any of the following can be selected using bits 3 and 4 (WDCS3 and WDCS4).
     Low-speed Ring-OSC clock ($f_{RL}$)
     System clock ($f_X$)
     Watchdog timer operation stopped
   - Cycle: Set using bits 2 to 0 (WDCS2 to WDCS0)
3. After the above procedures are executed, writing ACH to WDTE clears the count to 0, enabling recounting.

**Notes 1.** As soon as WDTM is written, the counter of the watchdog timer is cleared.

**2.** Set bits 7, 6, and 5 to 0, 1, 1, respectively. Do not set the other values.

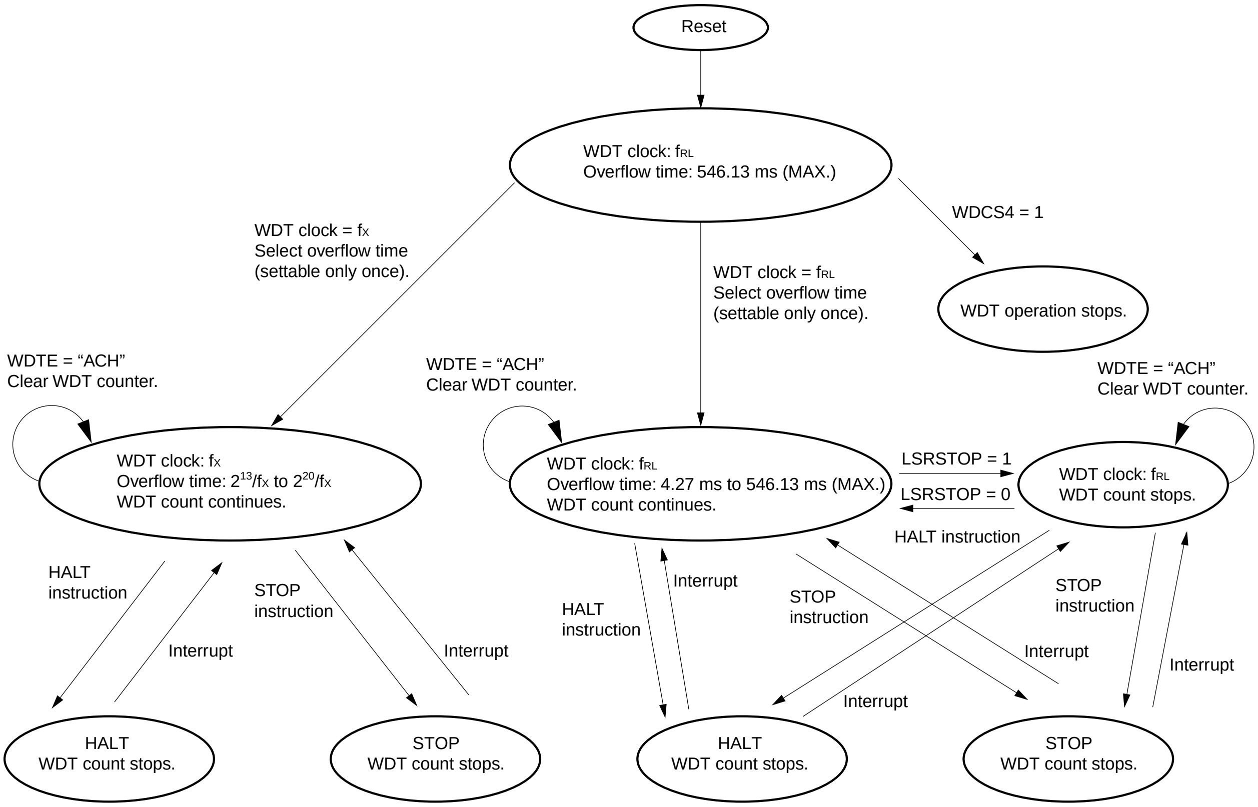
**3.** If the watchdog timer is stopped by setting WDCS4 and WDCS3 to 1 and ×, respectively, an internal reset signal is not generated even if the following processing is performed.
- WDTM is written a second time.
- A 1-bit memory manipulation instruction is executed to WDTE.
- A value other than ACH is written to WDTE.

**Caution In this mode, watchdog timer operation is stopped during HALT/STOP instruction execution. After HALT/STOP mode is released, counting is started again using the operation clock of the watchdog timer set before HALT/STOP instruction execution by WDTM. At this time, the counter is not cleared to 0 but holds its value.**

For the watchdog timer operation during STOP mode and HALT mode in each status, see **9.4.3 Watchdog timer operation in STOP mode (when "low-speed Ring-OSC can be stopped by software" is selected by option byte)** and **9.4.4 Watchdog timer operation in HALT mode (when "low-speed Ring-OSC can be stopped by software" is selected by option byte)**.

A status transition diagram is shown below.

**Figure 9-5. Status Transition Diagram When “Low-Speed Ring-OSC Can Be Stopped by Software” Is Selected by Option Byte**



### 9.4.3 Watchdog timer operation in STOP mode (when “low-speed Ring-OSC can be stopped by software” is selected by option byte)

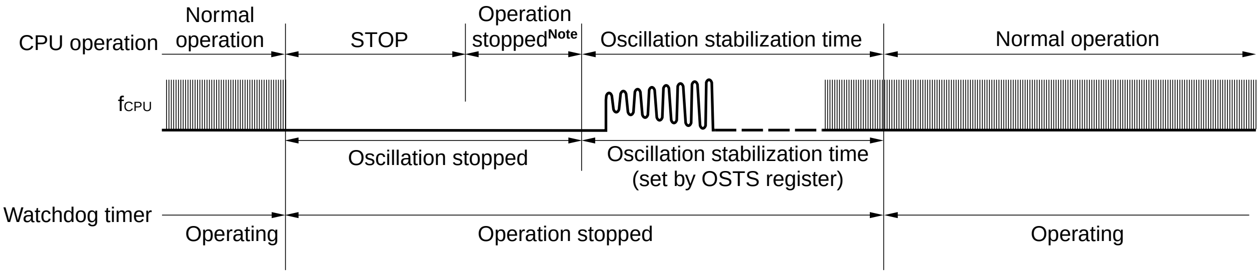
The watchdog timer stops counting during STOP instruction execution regardless of whether the system clock or low-speed Ring-OSC clock is being used.

**(1) When the watchdog timer operation clock is the clock to peripheral hardware (fX) when the STOP instruction is executed**

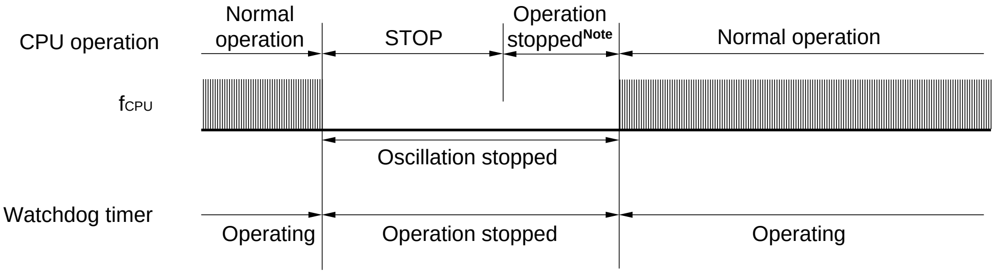
When STOP instruction is executed, operation of the watchdog timer is stopped. After STOP mode is released, operation stops for 34 μs (TYP.) (after waiting for the oscillation stabilization time set by the oscillation stabilization time select register (OSTS) after operation stops in the case of crystal/ceramic oscillation) and then counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

**Figure 9-6. Operation in STOP Mode (WDT Operation Clock: Clock to Peripheral Hardware)**

**<1> CPU clock: Crystal/ceramic oscillation clock**



**<2> CPU clock: High-speed Ring-OSC clock or external clock input**



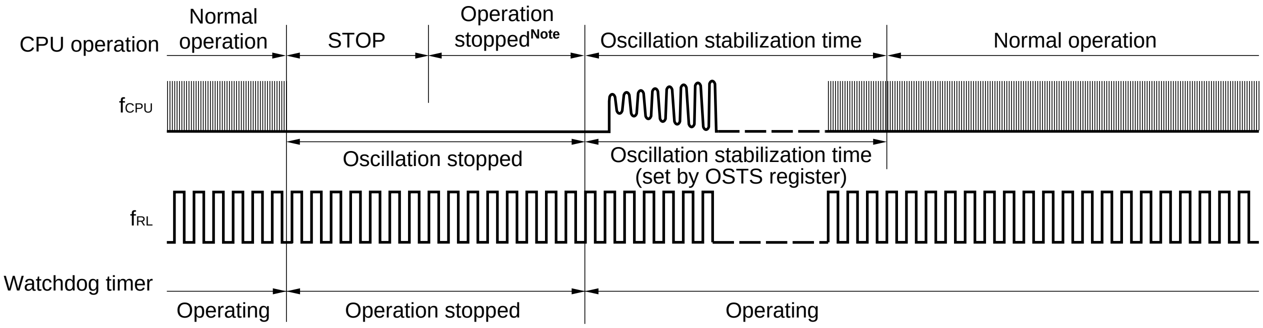
**Note** The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

**(2) When the watchdog timer operation clock is the low-speed Ring-OSC clock ($f_{RL}$) when the STOP instruction is executed**

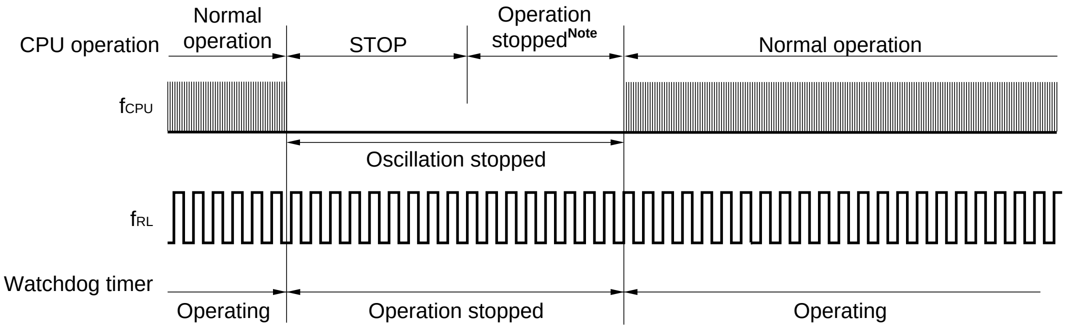
When the STOP instruction is executed, operation of the watchdog timer is stopped. After STOP mode is released, operation stops for 34 μs (TYP.) and then counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

**Figure 9-7. Operation in STOP Mode (WDT Operation Clock: Low-Speed Ring-OSC Clock)**

**<1> CPU clock: Crystal/ceramic oscillation clock**



**<2> CPU clock: High-speed Ring-OSC clock or external clock input**

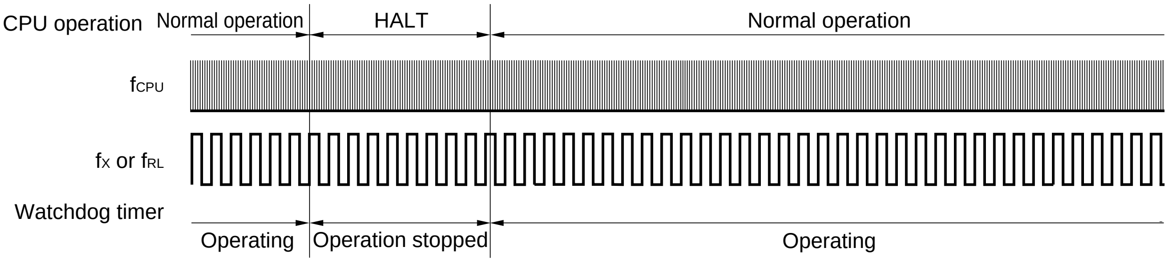


**Note** The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

### 9.4.4 Watchdog timer operation in HALT mode (when “low-speed Ring-OSC can be stopped by software” is selected by option byte)

The watchdog timer stops counting during HALT instruction execution regardless of whether the operation clock of the watchdog timer is the system clock ($f_X$) or low-speed Ring-OSC clock ($f_{RL}$). After HALT mode is released, counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

**Figure 9-8. Operation in HALT Mode**



## 10.1 Functions of A/D Converter

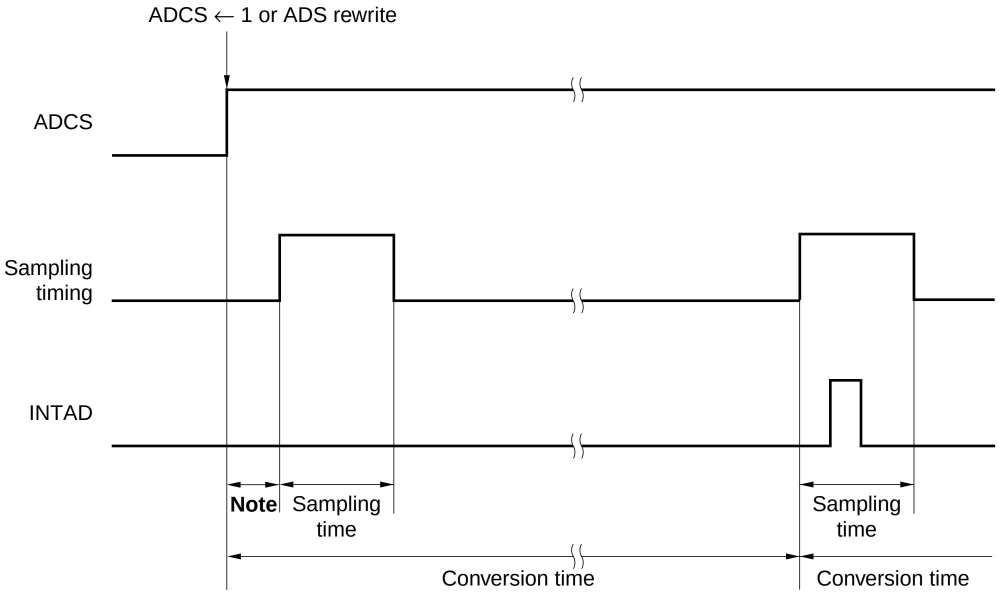
The A/D converter converts an analog input signal into a digital value, and consists of up to four channels (ANI0 to ANI3) with a resolution of 10 bits.

The A/D converter has the following function.

- **10-bit resolution A/D conversion**
  10-bit resolution A/D conversion is carried out repeatedly for one channel selected from analog inputs ANI0 to ANI3. Each time an A/D conversion operation ends, an interrupt request (INTAD) is generated.

Figure 10-1 shows the timing of sampling and A/D conversion, and Table 10-1 shows the sampling time and A/D conversion time.

**Figure 10-1. Timing of A/D Converter Sampling and A/D Conversion**



**Note** 2 or 3 clocks are required from the ADCS rising to sampling start.

**Table 10-1. Sampling Time and A/D Conversion Time**

| FR2 | FR1 | FR0 | Reference Voltage Range$^{\text{Note 1}}$ | Sampling Time$^{\text{Note 2}}$ | Conversion Time$^{\text{Note 3}}$ | $f_{XP}$ = 8 MHz | | $f_{XP}$ = 10 MHz | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Sampling Time$^{\text{Note 2}}$ | Conversion Time$^{\text{Note 3}}$ | Sampling Time$^{\text{Note 2}}$ | Conversion Time$^{\text{Note 3}}$ |
| 0 | 0 | 0 | $AV_{REF} \geq 4.5$ V | 12/$f_{XP}$ | 36/$f_{XP}$ | 1.5 µs | 4.5 µs | 1.2 µs | 3.6 µs |
| 0 | 0 | 1 | $AV_{REF} \geq 2.85$ V | 24/$f_{XP}$ | 48/$f_{XP}$ | 3.0 µs | 6.0 µs | Setting prohibited<br>(2.4 µs) | Setting prohibited<br>(4.8 µs) |
| 0 | 1 | 0 | $AV_{REF} \geq 2.7$ V | 48/$f_{XP}$ | 72/$f_{XP}$ | Setting prohibited<br>(6.0 µs) | Setting prohibited<br>(9.0 µs) | Setting prohibited<br>(4.8 µs) | Setting prohibited<br>(7.2 µs) |
| 0 | 1 | 1 | $AV_{REF} \geq 2.7$ V | 88/$f_{XP}$ | 112/$f_{XP}$ | 11.0 µs | 14.0 µs | Setting prohibited<br>(8.8 µs) | Setting prohibited<br>(11.2 µs) |
| 1 | 0 | 0 | $AV_{REF} \geq 4.5$ V | 24/$f_{XP}$ | 72/$f_{XP}$ | 3.0 µs | 9.0 µs | 2.4 µs | 7.2 µs |
| 1 | 0 | 1 | $AV_{REF} \geq 2.85$ V | 48/$f_{XP}$ | 96/$f_{XP}$ | 6.0 µs | 12.0 µs | 4.8 µs | 9.6 µs |
| 1 | 1 | 0 | $AV_{REF} \geq 2.7$ V | 96/$f_{XP}$ | 144/$f_{XP}$ | 12.0 µs | 18.0 µs | Setting prohibited<br>(9.6 µs) | Setting prohibited<br>(14.4 µs) |
| 1 | 1 | 1 | $AV_{REF} \geq 2.7$ V | 176/$f_{XP}$ | 224/$f_{XP}$ | 22.0 µs | 28.0 µs | 17.2 µs | 22.4 µs |

**Notes 1.** Be sure to set the FR2, FR1, and FR0 in accordance with the reference voltage range and satisfy **Notes 2** and **3** below.

Example When $AV_{REF} \geq 2.7$ V

- Set FR2, FR1, and FR0 = 0, 1, 1 or 1, 1, 1.
- The sampling time is 11.0 µs or more and the A/D conversion time is 14.0 µs or more and 100 µs or less.

**2.** Set the sampling time as follows.

- $AV_{REF} \geq 4.5$ V: 1.0 µs or more
- $AV_{REF} \geq 4.0$ V: 2.4 µs or more
- $AV_{REF} \geq 2.85$ V: 3.0 µs or more
- $AV_{REF} \geq 2.7$ V: 11.0 µs or more

**3.** Set the A/D conversion time as follows.

- $AV_{REF} \geq 4.5$ V: 3.0 µs or more and less than 100 µs
- $AV_{REF} \geq 4.0$ V: 4.8 µs or more and less than 100 µs
- $AV_{REF} \geq 2.85$ V: 6.0 µs or more and less than 100 µs
- $AV_{REF} \geq 2.7$ V: 14.0 µs or more and less than 100 µs

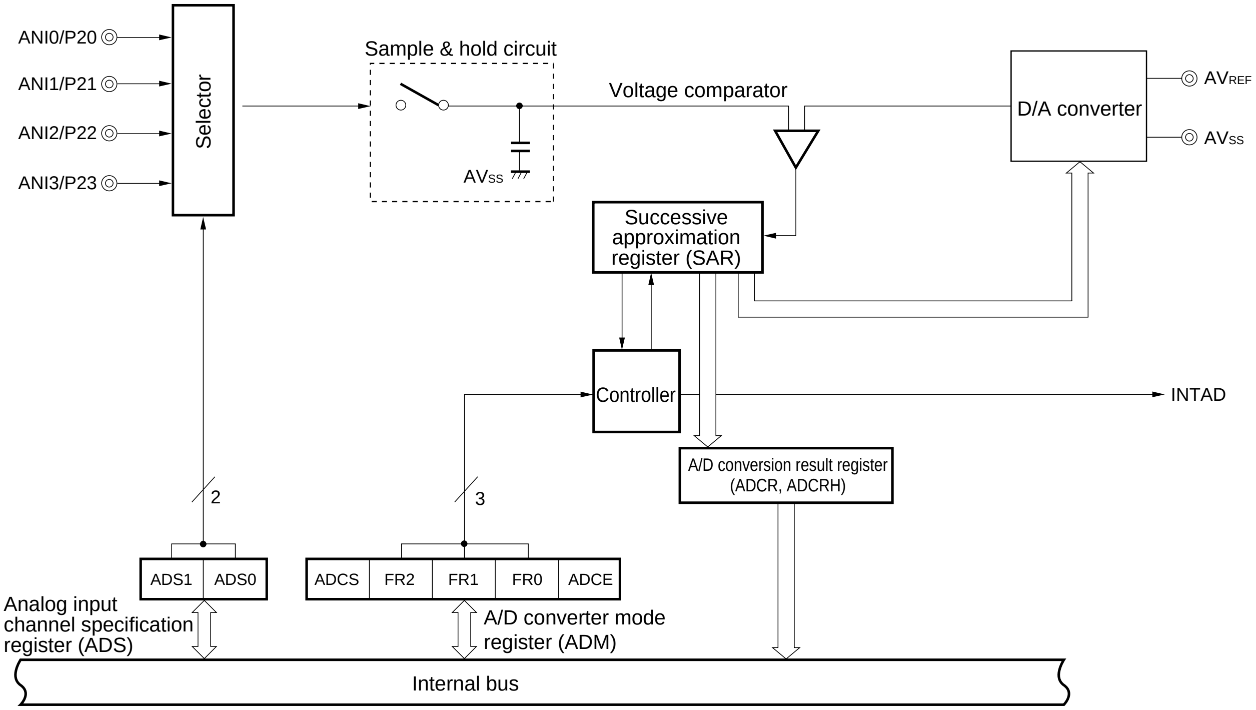
**Caution The above sampling time and conversion time do not include the clock frequency error. Select the conversion time taking the clock frequency error into consideration.**

**Remarks 1.** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

**2.** The conversion time refers to the total of the sampling time and the time from successively comparing with the sampling value until the conversion result is output.

Figure 10-2 shows the block diagram of A/D converter.

**Figure 10-2. Block Diagram of A/D Converter**


ANI0/P20
ANI1/P21
ANI2/P22
ANI3/P23
Selector
Sample & hold circuit
AVSS
Voltage comparator
D/A converter
AVREF
AVSS
Successive approximation register (SAR)
Controller
INTAD
A/D conversion result register (ADCR, ADCRH)
2
3
ADS1
ADS0
ADCS
FR2
FR1
FR0
ADCE
Analog input channel specification register (ADS)
A/D converter mode register (ADM)
Internal bus

## 10.2 Configuration of A/D Converter

The A/D converter consists of the following hardware.

**Table 10-2. Registers of A/D Converter Used on Software**

| Item | Configuration |
|---|---|
| Registers | 10-bit A/D conversion result register (ADCR)<br>8-bit A/D conversion result register (ADCRH)<br>A/D converter mode register (ADM)<br>Analog input channel specification register (ADS)<br>Port mode control register 2 (PMC2)<br>Port mode register 2 (PM2) |

**(1) ANI0 to ANI3 pins**

These are the analog input pins of the 4-channel A/D converter. They input analog signals to be converted into digital signals. Pins other than the one selected as the analog input pin by the analog input channel specification register (ADS) can be used as input port pins.

**(2) Sample & hold circuit**

The sample & hold circuit samples the input signal of the analog input pin selected by the selector when A/D conversion is started, and holds the sampled analog input voltage value during A/D conversion.

**(3) D/A converter**

The D/A converter is connected between $AV_{REF}$ and $AV_{SS}$, and generates a voltage to be compared with the analog input signal.

**(4) Voltage comparator**

The voltage comparator compares the sampled analog input voltage and the output voltage of the D/A converter.

**(5) Successive approximation register (SAR)**

This register compares the sampled analog voltage and the voltage of the D/A converter, and converts the result, starting from the most significant bit (MSB).

When the voltage value is converted into a digital value down to the least significant bit (LSB) (end of A/D conversion), the contents of the SAR register are transferred to the A/D conversion result register (ADCR).

**(6) 10-bit A/D conversion result register (ADCR)**

The result of A/D conversion is loaded from the successive approximation register to this register each time A/D conversion is completed, and the ADCR register holds the result of A/D conversion in its lower 10 bits (the higher 6 bits are fixed to 0).

**(7) 8-bit A/D conversion result register (ADCRH)**

The result of A/D conversion is loaded from the successive approximation register to this register each time A/D conversion is completed, and the ADCRH register holds the result of A/D conversion in its higher 8 bits.

**(8) Controller**

When A/D conversion has been completed, INTAD is generated.

**(9) AV$_{REF}$ pin**

This pin inputs an analog power/reference voltage to the A/D converter. When the A/D converter is not used, connect this pin to V$_{DD}$.

The signal input to ANI0 to ANI3 is converted into a digital signal, based on the voltage applied across AV$_{REF}$ and AV$_{SS}$.

**(10) AV$_{SS}$ pin**

This is the ground potential pin of the A/D converter. Always use this pin at the same potential as that of the V$_{SS}$ pin even when the A/D converter is not used.

**(11) A/D converter mode register (ADM)**

This register is used to set the conversion time of the analog input signal to be converted, and to start or stop the conversion operation.

**(12) Analog input channel specification register (ADS)**

This register is used to specify the port that inputs the analog voltage to be converted into a digital signal.

**(13) Port mode control register 2 (PMC2)**

This register is used when the P20/ANI0 to P23/ANI3 pins are used as the analog input pins of the A/D converter.

## 10.3 Registers Used by A/D Converter

The A/D converter uses the following six registers.

- A/D converter mode register (ADM)
- Analog input channel specification register (ADS)
- 10-bit A/D conversion result register (ADCR)
- 8-bit A/D conversion result register (ADCRH)
- Port mode control register 2 (PMC2)
- Port mode register 2 (PM2)

**(1) A/D converter mode register (ADM)**

This register sets the conversion time for analog input to be A/D converted, and starts/stops conversion.

ADM can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

Figure 10-3. Format of A/D Converter Mode Register (ADM)

Address: FF80H After reset: 00H R/W

| Symbol | <7> | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| ADM | ADCS | 0 | FR2 | FR1 | FR0 | 0 | 0 | ADCE |

| ADCS | A/D conversion operation control |
|---|---|
| 0 | Stops conversion operation |
| 1[Note 1] | Starts conversion operation |

| FR2 | FR1 | FR0 | Reference Voltage Range[Note 2] | Sampling Time[Note 3] | Conversion Time[Note 4] | $f_{XP}$ = 8 MHz | | $f_{XP}$ = 10 MHz | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Sampling Time[Note 3] | Conversion Time[Note 4] | Sampling Time[Note 3] | Conversion Time[Note 4] |
| 0 | 0 | 0 | $AV_{REF} \geq$ 4.5 V | $12/f_{XP}$ | $36/f_{XP}$ | 1.5 µs | 4.5 µs | 1.2 µs | 3.6 µs |
| 0 | 0 | 1 | $AV_{REF} \geq$ 2.85 V | $24/f_{XP}$ | $48/f_{XP}$ | 3.0 µs | 6.0 µs | Setting prohibited (2.4 µs) | Setting prohibited (4.8 µs) |
| 0 | 1 | 0 | $AV_{REF} \geq$ 2.7 V | $48/f_{XP}$ | $72/f_{XP}$ | Setting prohibited (6.0 µs) | Setting prohibited (9.0 µs) | Setting prohibited (4.8 µs) | Setting prohibited (7.2 µs) |
| 0 | 1 | 1 | $AV_{REF} \geq$ 2.7 V | $88/f_{XP}$ | $112/f_{XP}$ | 11.0 µs | 14.0 µs | Setting prohibited (8.8 µs) | Setting prohibited (11.2 µs) |
| 1 | 0 | 0 | $AV_{REF} \geq$ 4.5 V | $24/f_{XP}$ | $72/f_{XP}$ | 3.0 µs | 9.0 µs | 2.4 µs | 7.2 µs |
| 1 | 0 | 1 | $AV_{REF} \geq$ 2.85 V | $48/f_{XP}$ | $96/f_{XP}$ | 6.0 µs | 12.0 µs | 4.8 µs | 9.6 µs |
| 1 | 1 | 0 | $AV_{REF} \geq$ 2.7 V | $96/f_{XP}$ | $144/f_{XP}$ | 12.0 µs | 18.0 µs | Setting prohibited (9.6 µs) | Setting prohibited (14.4 µs) |
| 1 | 1 | 1 | $AV_{REF} \geq$ 2.7 V | $176/f_{XP}$ | $224/f_{XP}$ | 22.0 µs | 28.0 µs | 17.6 µs | 22.4 µs |

| ADCE | Comparator operation control[Note 5] |
|---|---|
| 0[Note 1] | Stops operation of comparator |
| 1 | Enables operation of comparator |

**Remarks 1.** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

**2.** The conversion time refers to the total of the sampling time and the time from successively comparing with the sampling value until the conversion result is output.

**Notes 1.** Even when the ADCE = 0 (comparator operation stopped), the A/D conversion operation starts if the ADCS is set to 1. However, the data of the first conversion is out of the guaranteed-value range, so ignore it.

**2.** Be sure to set the FR2, FR1, and FR0 in accordance with the reference voltage range and satisfy **Notes 3** and **4** below.

Example When $AV_{REF} \geq$ 2.7 V

- Set FR2, FR1, and FR0 = 0, 1, 1 or 1, 1, 1.
- The sampling time is 11.0 µs or more and the A/D conversion time is 14.0 µs or more and 100 µs or less.

**Notes 3.** Set the sampling time as follows.

- AVREF ≥ 4.5 V: 1.0 μs or more
- AVREF ≥ 4.0 V: 2.4 μs or more
- AVREF ≥ 2.85 V: 3.0 μs or more
- AVREF ≥ 2.7 V: 11.0 μs or more

**4.** Set the A/D conversion time as follows.

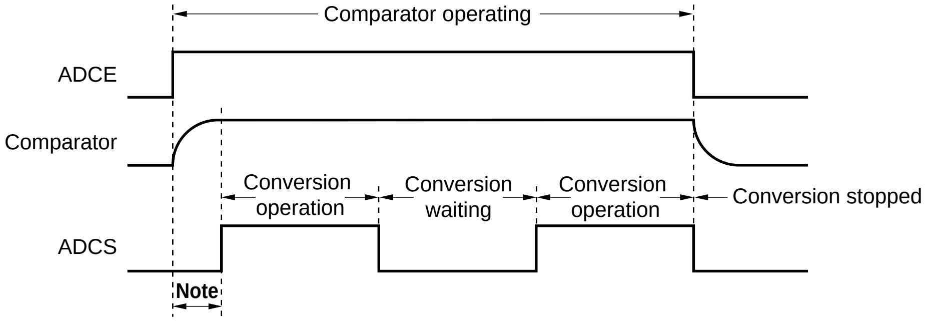
- AVREF ≥ 4.5 V: 3.0 μs or more and less than 100 μs
- AVREF ≥ 4.0 V: 4.8 μs or more and less than 100 μs
- AVREF ≥ 2.85 V: 6.0 μs or more and less than 100 μs
- AVREF ≥ 2.7 V: 14.0 μs or more and less than 100 μs

**5.** The operation of the comparator is controlled by ADCS and ADCE, and it takes 1 μs from operation start to operation stabilization. Therefore, when ADCS is set to 1 after 1 μs or more has elapsed from the time ADCE is set to 1, the conversion result at that time has priority over the first conversion result. If the ADCS is set to 1 without waiting for 1 μs or longer, ignore the data of the first conversion.

**Table 10-3. Settings of ADCS and ADCE**

| ADCS | ADCE | A/D Conversion Operation |
|---|---|---|
| 0 | 0 | Stop status (DC power consumption path does not exist) |
| 0 | 1 | Conversion waiting mode (only comparator consumes power) |
| 1 | × | Conversion mode |

**Figure 10-4. Timing Chart When Comparator Is Used**



**Note** The time from the rising of the ADCE bit to the rising of the ADCS bit must be 1 μs or longer to stabilize the internal circuit.

**Cautions 1. The above sampling time and conversion time do not include the clock frequency error. Select the conversion time taking the clock frequency error into consideration.**

**2. If a bit other than ADCS of ADM is manipulated while A/D conversion is stopped (ADCS = 0) and then A/D conversion is started, execute two NOP instructions or an instruction equivalent to two machine cycles, and set ADCS to 1.**

**3. A/D conversion must be stopped (ADCS = 0) before rewriting bits FR0 to FR2.**

**4. Be sure to clear bits 6, 2, and 1 to 0.**

### (2) Analog input channel specification register (ADS)

This register specifies the input port of the analog voltage to be A/D converted.

ADS can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Figure 10-5. Format of Analog Input Channel Specification Register (ADS)**

Address: FF81H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADS | 0 | 0 | 0 | 0 | 0 | 0 | ADS1 | ADS0 |

| ADS1 | ADS0 | Analog input channel specification |
|---|---|---|
| 0 | 0 | ANI0 |
| 0 | 1 | ANI1 |
| 1 | 0 | ANI2 |
| 1 | 1 | ANI3 |

**Caution Be sure to clear bits 2 to 7 of ADS to 0.**

### (3) 10-bit A/D conversion result register (ADCR)

This register is a 16-bit register that stores the A/D conversion result. The higher six bits are fixed to 0. Each time A/D conversion ends, the conversion result is loaded from the successive approximation register, and is stored in ADCR in order starting from bit 1 of FF19H. FF19H indicates the higher 2 bits of the conversion result, and FF18H indicates the lower 8 bits of the conversion result.

ADCR can be read by a 16-bit memory manipulation instruction.

Reset signal generation makes ADCR undefined.

**Figure 10-6. Format of 10-Bit A/D Conversion Result Register (ADCR)**

Address: FF18H, FF19H After reset: Undefined R

| Symbol | FF19H | | | | | | | | FF18H | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCR | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |

**Caution When writing to the A/D converter mode register (ADM) and analog input channel specification register (ADS), the contents of ADCR may become undefined. Read the conversion result following conversion completion before writing to ADM and ADS. Using timing other than the above may cause an incorrect conversion result to be read.**

**(4) 8-bit A/D conversion result register (ADCRH)**

This register is an 8-bit register that stores the A/D conversion result. It stores the higher 8 bits of a 10-bit resolution result.

ADCRH can be read by an 8-bit memory manipulation instruction.

Reset signal generation makes ADCRH undefined.

**Figure 10-7. Format of 8-Bit A/D Conversion Result Register (ADCRH)**

Address: FF1AH After reset: Undefined R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADCRH | | | | | | | | |

**(5) Port mode control register 2 (PMC2) and port mode register 2 (PM2)**

When using the P20/ANI0 to P23/ANI3 pins for analog input, set PMC20 to PMC23 and PM20 to PM23 to 1. At this time, the output latches of P20 to P23 may be 0 or 1.

PMC2 and PM2 are set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears PMC2 to 00H and sets PM2 to FFH.

**Figure 10-8. Format of Port Mode Control Register 2 (PMC2)**

Address: FF84H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMC2 | 0 | 0 | 0 | 0 | PMC23 | PMC22 | PMC21 | PMC20 |

| PMC2n | Operation mode specification (n = 0 to 3) |
|---|---|
| 0 | Port mode |
| 1 | A/D converter mode |

**Figure 10-9. Format of Port Mode Register 2 (PM2)**

Address: FF22H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM2 | 1 | 1 | 1 | 1 | PM23 | PM22 | PM21 | PM20 |

| PM2n | Pmn pin I/O mode selection (n = 0 to 3) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

**Caution When PMC20 to PMC23 are set to 1, the P20/ANI0 to P23/ANI3 pins cannot be used as port pins.**

## 10.4 A/D Converter Operations

### 10.4.1 Basic operations of A/D converter

<1> Select one channel for A/D conversion using the analog input channel specification register (ADS).

<2> Set ADCE to 1 and wait for 1 $\mu$s or longer.

<3> Execute two NOP instructions or an instruction equivalent to two machine cycles.

<4> Set ADCS to 1 and start the conversion operation.
(<5> to <11> are operations performed by hardware.)

<5> The voltage input to the selected analog input channel is sampled by the sample & hold circuit.

<6> When sampling has been done for a certain time, the sample & hold circuit is placed in the hold state and the input analog voltage is held until the A/D conversion operation has ended.

<7> Bit 9 of the successive approximation register (SAR) is set. The D/A converter voltage tap is set to (1/2) $AV_{REF}$ by the tap selector.

<8> The voltage difference between the D/A converter voltage tap and analog input is compared by the voltage comparator. If the analog input is greater than (1/2) $AV_{REF}$, the MSB of SAR remains set to 1. If the analog input is smaller than (1/2) $AV_{REF}$, the MSB is reset to 0.

<9> Next, bit 8 of SAR is automatically set to 1, and the operation proceeds to the next comparison. The D/A converter voltage tap is selected according to the preset value of bit 9, as described below.

- Bit 9 = 1: (3/4) $AV_{REF}$
- Bit 9 = 0: (1/4) $AV_{REF}$

The voltage tap and analog input voltage are compared and bit 8 of SAR is manipulated as follows.

- Analog input voltage ≥ Voltage tap: Bit 8 = 1
- Analog input voltage < Voltage tap: Bit 8 = 0

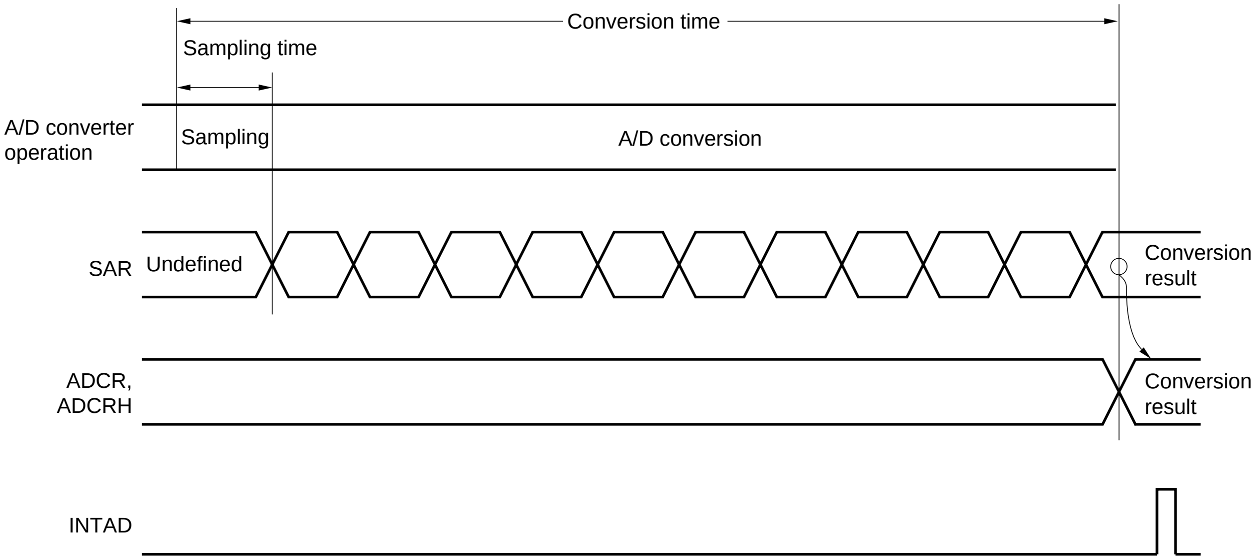
<10> Comparison is continued in this way up to bit 0 of SAR.

<11> Upon completion of the comparison of 10 bits, an effective digital result value remains in SAR, and the result value is transferred to the A/D conversion result register (ADCR, ADCRH) and then latched.
At the same time, the A/D conversion end interrupt request (INTAD) can also be generated.

<12> Repeat steps <5> to <11>, until ADCS is cleared to 0.
To stop the A/D converter, clear ADCS to 0.
To restart A/D conversion from the status of ADCE = 1, start from <3>. To restart A/D conversion from the status of ADCE = 0, however, start from <2>.

**Remark** The following two types of A/D conversion result registers can be used.
<1> ADCR (16 bits): Stores a 10-bit A/D conversion value.
<2> ADCRH (8 bits): Stores an 8-bit A/D conversion value.

**Figure 10-10. Basic Operation of A/D Converter**



A/D conversion operations are performed continuously until bit 7 (ADCS) of the A/D converter mode register (ADM) is reset (0) by software.

If a write operation is performed to ADM or the analog input channel specification register (ADS) during an A/D conversion operation, the conversion operation is initialized, and if the ADCS bit is set (1), conversion starts again from the beginning.

Reset input makes the A/D conversion result register (ADCR, ADCRH) undefined.

### 10.4.2 Input voltage and conversion results

The relationship between the analog input voltage input to the analog input pins (ANI0 to ANI3) and the theoretical A/D conversion result (stored in the 10-bit A/D conversion result register (ADCR)) is shown by the following expression.

$$\text{SAR} = \text{INT}\left(\frac{V_{\text{AIN}}}{AV_{\text{REF}}} \times 1024 + 0.5\right)$$

or

$$(\text{ADCR} - 0.5) \times \frac{AV_{\text{REF}}}{1024} \leq V_{\text{AIN}} < (\text{ADCR} + 0.5) \times \frac{AV_{\text{REF}}}{1024}$$

where, INT( ): Function which returns integer part of value in parentheses

$V_{\text{AIN}}$: Analog input voltage

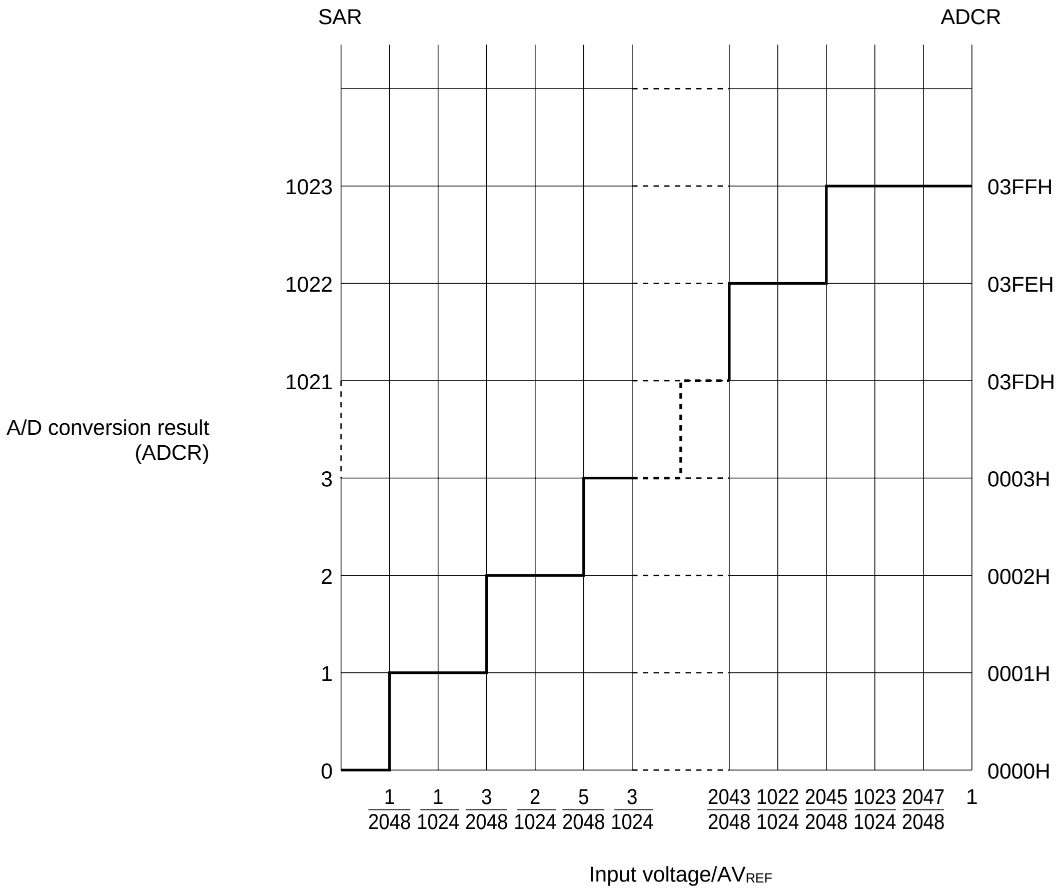
$AV_{\text{REF}}$: $AV_{\text{REF}}$ pin voltage

ADCR: 10-bit A/D conversion result register (ADCR) value

SAR: Successive approximation register

Figure 10-11 shows the relationship between the analog input voltage and the A/D conversion result.

**Figure 10-11. Relationship Between Analog Input Voltage and A/D Conversion Result**



### 10.4.3 A/D converter operation mode

The operation mode of the A/D converter is the select mode. One channel of analog input is selected from ANI0 to ANI3 by the analog input channel specification register (ADS) and A/D conversion is executed.

**(1) A/D conversion operation**

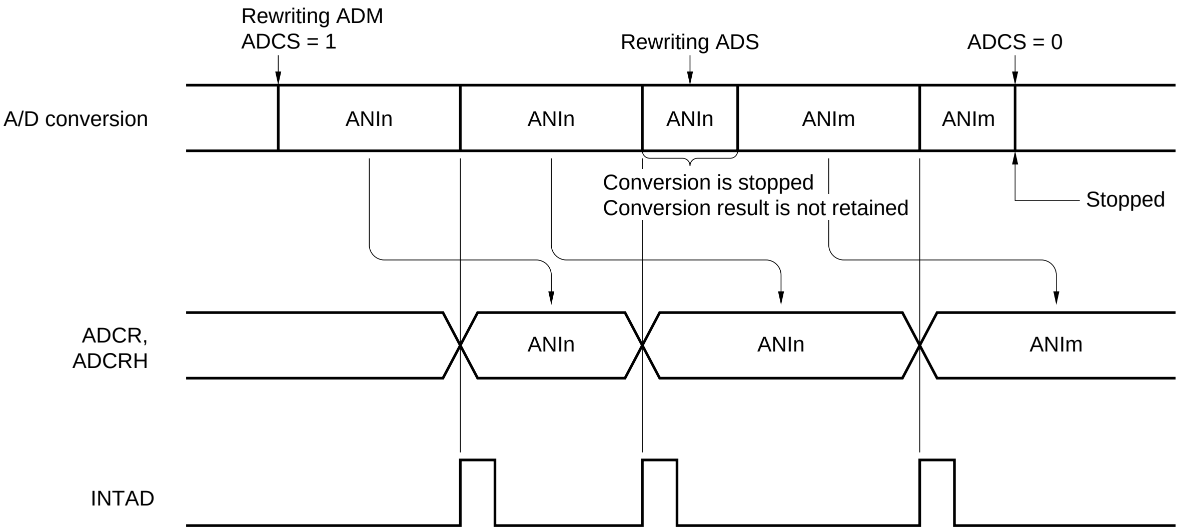
By setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 1, the A/D conversion operation of the voltage, which is applied to the analog input pin specified by the analog input channel specification register (ADS), is started.

When A/D conversion has been completed, the result of the A/D conversion is stored in the A/D conversion result register (ADCR, ADCRH), and an interrupt request signal (INTAD) is generated. Once the A/D conversion has started and when one A/D conversion has been completed, the next A/D conversion operation is immediately started. The A/D conversion operations are repeated until new data is written to ADS.

If ADM or ADS is written during A/D conversion, the A/D conversion operation under execution is stopped and restarted from the beginning.

If 0 is written to ADCS during A/D conversion, A/D conversion is immediately stopped. At this time, the conversion result is undefined.

**Figure 10-12. A/D Conversion Operation**



**Remarks 1.** n = 0 to 3

**2.** m = 0 to 3

The setting method is described below.

<1> Set bit 0 (ADCE) of the A/D converter mode register (ADM) to 1.
<2> Select the channel and conversion time using bits 1 and 0 (ADS1, ADS0) of the analog input channel specification register (ADS) and bits 5 to 3 (FR2 to FR0) of ADM.
<3> Execute two NOP instructions or an instruction equivalent to two machine cycles.
<4> Set bit 7 (ADCS) of ADM to 1 to start A/D conversion.
<5> An interrupt request signal (INTAD) is generated.
<6> Transfer the A/D conversion data to the A/D conversion result register (ADCR, ADCRH).

<Change the channel>

<7> Change the channel using bits 1 and 0 (ADS1, ADS0) of ADS.
<8> An interrupt request signal (INTAD) is generated.
<9> Transfer the A/D conversion data to the A/D conversion result register (ADCR, ADCRH).

<Complete A/D conversion>

<10> Clear ADCS to 0.
<11> Clear ADCE to 0.

**Cautions 1. Make sure the period of <1> to <4> is 1 $\mu$s or more.**
**2. It is no problem if the order of <1> and <2> is reversed.**
**3. <1> can be omitted. However, ignore the data resulting from the first conversion after <4> in this case.**
**4. The period from <5> to <8> differs from the conversion time set using bits 5 to 3 (FR2 to FR0) of ADM. The period from <7> to <8> is the conversion time set using FR2 to FR0.**

## 10.5 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 1LSB (Least Significant Bit). The percentage of 1LSB with respect to the full scale is expressed by %FSR (Full Scale Range).

1LSB is as follows when the resolution is 10 bits.

$1\text{LSB} = 1/2^{10} = 1/1024$
$= 0.098\%\text{FSR}$

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, integral linearity error, and differential linearity 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.

**(3) Quantization error**

When analog values are converted to digital values, a ±1/2LSB error naturally occurs. In an A/D converter, an analog input voltage in a range of ±1/2LSB 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 10-13. Overall Error**

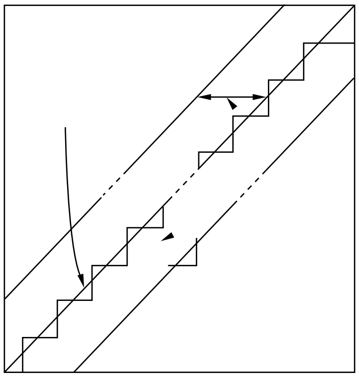


**Figure 10-14. Quantization Error**

**(5) Full-scale error**

This shows the difference between the actual measurement value of the analog input voltage and the theoretical value (Full-scale − 3/2LSB) when the digital output changes from 1......110 to 1......111.

**(6) 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.

**(7) Differential linearity error**

While the ideal width of code output is 1LSB, this indicates the difference between the actual measurement value and the ideal value.

**Figure 10-15. Zero-Scale Error**

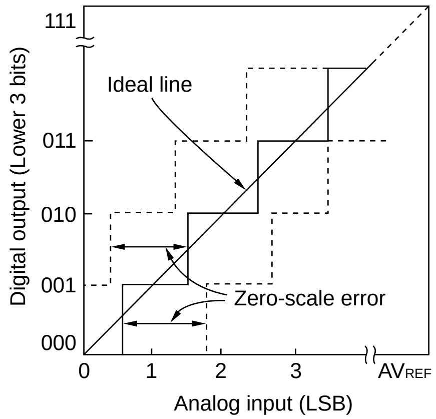


**Figure 10-16. Full-Scale Error**

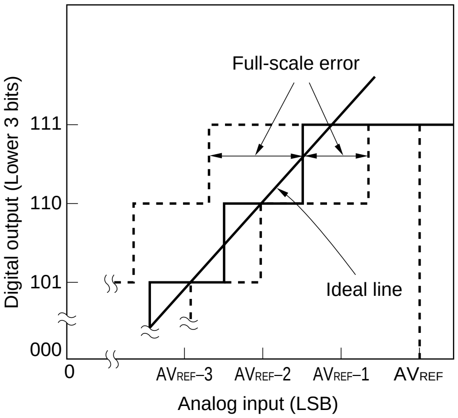


**Figure 10-17. Integral Linearity Error**

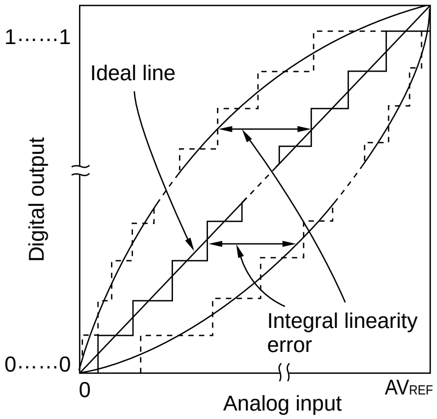
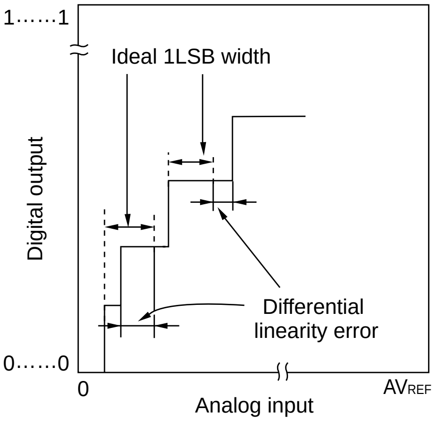


**Figure 10-18. Differential Linearity Error**



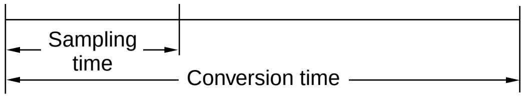
**(8) Conversion time**

This expresses the time from the start of sampling to when the digital output is 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.



## 10.6 Cautions for A/D Converter

**(1) Operating current in STOP mode**

The A/D converter stops operating in the STOP mode. At this time, the operating current can be reduced by clearing bit 7 (ADCS) and bit 0 (ADCE) of the A/D converter mode register (ADM) to 0.

**(2) Input range of ANI0 to ANI3**

Observe the rated range of the ANI0 to ANI3 input voltage. If a voltage of $AV_{REF}$ or higher and $AV_{SS}$ or lower (even in the range of absolute maximum ratings) is input to an analog input channel, the converted value of that channel becomes undefined. In addition, the converted values of the other channels may also be affected.

**(3) Conflicting operations**

<1> Conflict between A/D conversion result register (ADCR, ADCRH) write and ADCR, ADCRH read by instruction upon the end of conversion
ADCR, ADCRH read has priority. After the read operation, the new conversion result is written to ADCR, ADCRH.

<2> Conflict between ADCR, ADCRH write and A/D converter mode register (ADM) write or analog input channel specification register (ADS) write upon the end of conversion
ADM or ADS write has priority. ADCR, ADCRH write is not performed, nor is the conversion end interrupt signal (INTAD) generated.

**(4) Noise countermeasures**

To maintain the 10-bit resolution, attention must be paid to noise input to the $AV_{REF}$ pin and pins ANI0 to ANI3.

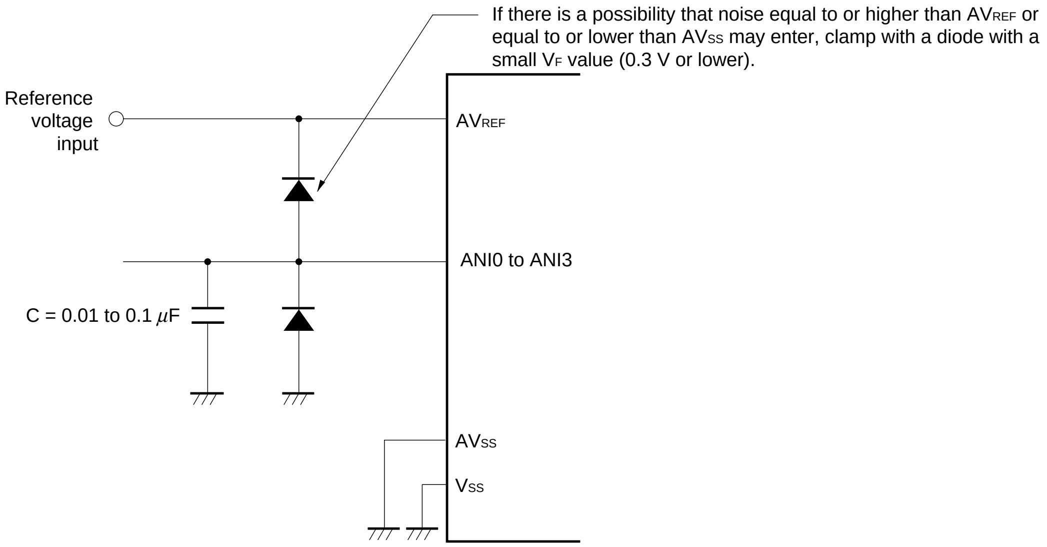
<1> Connect a capacitor with a low equivalent resistance and a high frequency response to the power supply.

<2> Because the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally, as shown in Figure 10-19, to reduce noise.

<3> Do not switch the A/D conversion function of the ANI0 to ANI3 pins to their alternate functions during conversion.

<4> The conversion accuracy can be improved by setting HALT mode immediately after the conversion starts.

**Figure 10-19. Analog Input Pin Connection**



If there is a possibility that noise equal to or higher than $AV_{REF}$ or equal to or lower than $AV_{SS}$ may enter, clamp with a diode with a small $V_F$ value (0.3 V or lower).
Reference voltage input
$AV_{REF}$
ANI0 to ANI3
C = 0.01 to 0.1 μF
$AV_{SS}$
$V_{SS}$

### (5) ANI0/P20 to ANI3/P23

<1> The analog input pins (ANI0 to ANI3) are also used as input port pins (P20 to P23).
When A/D conversion is performed with any of ANI0 to ANI3 selected, do not access port 2 (P20 to P23) while conversion is in progress; otherwise the conversion resolution may be degraded.

<2> If a digital pulse is applied to the pins adjacent to the pins currently used for A/D conversion, the expected value of the A/D conversion may not be obtained due to coupling noise. Therefore, do not apply a pulse to the pins adjacent to the pin undergoing A/D conversion.

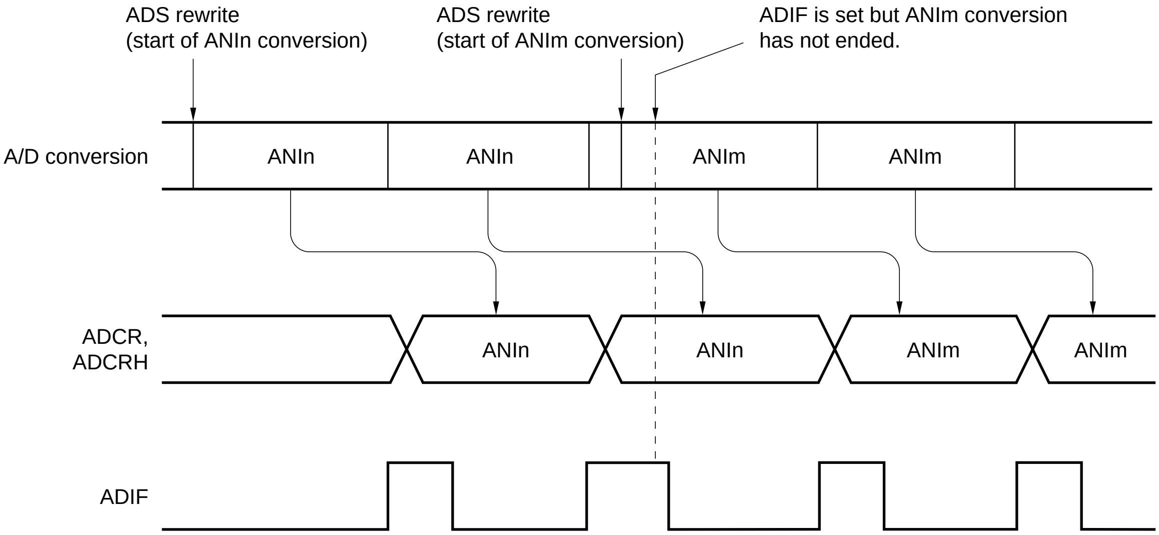
### (6) Input impedance of ANI0 to ANI3 pins

In this A/D converter, the internal sampling capacitor is charged and sampling is performed for approx. one sixth of the conversion time.
Since only the leakage current flows other than during sampling and the current for charging the capacitor also flows during sampling, the input impedance fluctuates during sampling and in the other state.
If the shortest conversion time of the reference voltage is used, to perform sufficient sampling, it is recommended to make the output impedance of the analog input source 1 kΩ or lower, or attach a capacitor of around 0.01 μF to 0.1 μF to the ANI0 to ANI3 pins (see **Figure 10-19**).

### (7) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the analog input channel specification register (ADS) is changed.
Therefore, if an analog input pin is changed during A/D conversion, the A/D conversion result and ADIF for the pre-change analog input may be set just before the ADS rewrite. Caution is therefore required since, at this time, when ADIF is read immediately after the ADS rewrite, ADIF is set despite the fact A/D conversion for the post-change analog input has not ended.
When A/D conversion is stopped and then resumed, clear ADIF before the A/D conversion operation is resumed.

**Figure 10-20. Timing of A/D Conversion End Interrupt Request Generation**



**Remarks 1.** n = 0 to 3
**2.** m = 0 to 3

### (8) Conversion results just after A/D conversion start

The first A/D conversion value immediately after A/D conversion starts may not fall within the rating range if the ADCS bit is set to 1 within 1 μs after the ADCE bit was set to 1, or if the ADCS bit is set to 1 with the ADCE bit = 0. Take measures such as polling the A/D conversion end interrupt request (INTAD) and removing the first conversion result.

### (9) A/D conversion result register (ADCR, ADCRH) read operation

When a write operation is performed to the A/D converter mode register (ADM) and analog input channel specification register (ADS), the contents of ADCR and ADCRH may become undefined. Read the conversion result following conversion completion before writing to ADM and ADS. Using a timing other than the above may cause an incorrect conversion result to be read.

### (10) Internal equivalent circuit

The equivalent circuit of the analog input block is shown below.

**Figure 10-21. Internal Equivalent Circuit of ANIn Pin**

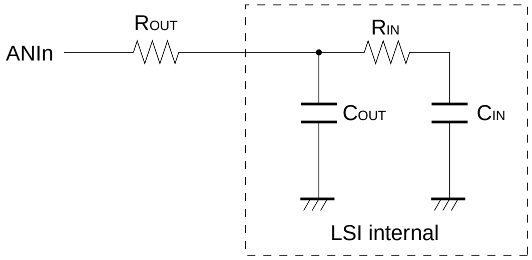


**Table 10-4. Resistance and Capacitance Values (Reference Values) of Equivalent Circuit**

| AVREF | ROUT | RIN | COUT | CIN |
|---|---|---|---|---|
| 4.5 V ≤ AVREF ≤ 5.5 V | 1 kΩ | 3 kΩ | 8 pF | 15 pF |
| 2.7 V ≤ AVREF < 4.5 V | 1 kΩ | 60 kΩ | 8 pF | 15 pF |

**Remarks 1.** The resistance and capacitance values shown in Table 10-4 are not guaranteed values.

**2.** n = 0 to 3

**3.** ROUT: Allowable signal source impedance

RIN: Analog input equivalent resistance

CIN: Analog input equivalent capacitance

COUT: Internal pin capacitance

# CHAPTER 11 SERIAL INTERFACE UART6

## 11.1 Functions of Serial Interface UART6

Serial interface UART6 has the following two modes.

**(1) Operation stop mode**

This mode is used when serial communication is not executed and can enable a reduction in the power consumption.

For details, see **11.4.1 Operation stop mode**.

**(2) Asynchronous serial interface (UART) mode**

This mode supports the LIN (Local Interconnect Network)-bus. The functions of this mode are outlined below.

For details, see **11.4.2 Asynchronous serial interface (UART) mode** and **11.4.3 Dedicated baud rate generator**.

- Two-pin configuration TxD6: Transmit data output pin
  RxD6: Receive data input pin
- Data length of communication data can be selected from 7 or 8 bits.
- Dedicated internal 8-bit baud rate generator allowing any baud rate to be set
- Transmission and reception can be performed independently.
- Twelve operating clock inputs selectable
- MSB- or LSB-first communication selectable
- Inverted transmission operation
- Synchronous break field transmission from 13 to 20 bits
- More than 11 bits can be identified for synchronous break field reception (SBF reception flag provided).

**Cautions 1. The TxD6 output inversion function inverts only the transmission side and not the reception side. To use this function, the reception side must be ready for reception of inverted data.**

**2. If clock supply to serial interface UART6 is not stopped (e.g., in the HALT mode), normal operation continues. If clock supply to serial interface UART6 is stopped (e.g., in the STOP mode), each register stops operating, and holds the value immediately before clock supply was stopped. The TxD6 pin also holds the value immediately before clock supply was stopped and outputs it. However, the operation is not guaranteed after clock supply is resumed. Therefore, reset the circuit so that POWER6 = 0, RXE6 = 0, and TXE6 = 0.**

**3. If data is continuously transmitted, the communication timing from the stop bit to the next start bit is extended two operating clocks of the macro. However, this does not affect the result of communication because the reception side initializes the timing when it has detected a start bit. Do not use the continuous transmission function if the interface is incorporated in LIN.**

**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.

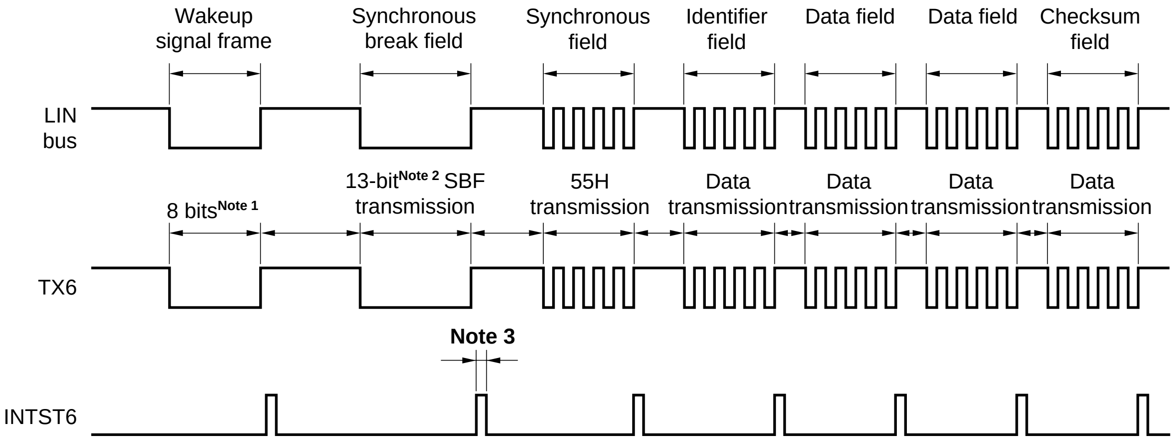
Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

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.

Figures 11-1 and 11-2 outline the transmission and reception operations of LIN.

**Figure 11-1. LIN Transmission Operation**



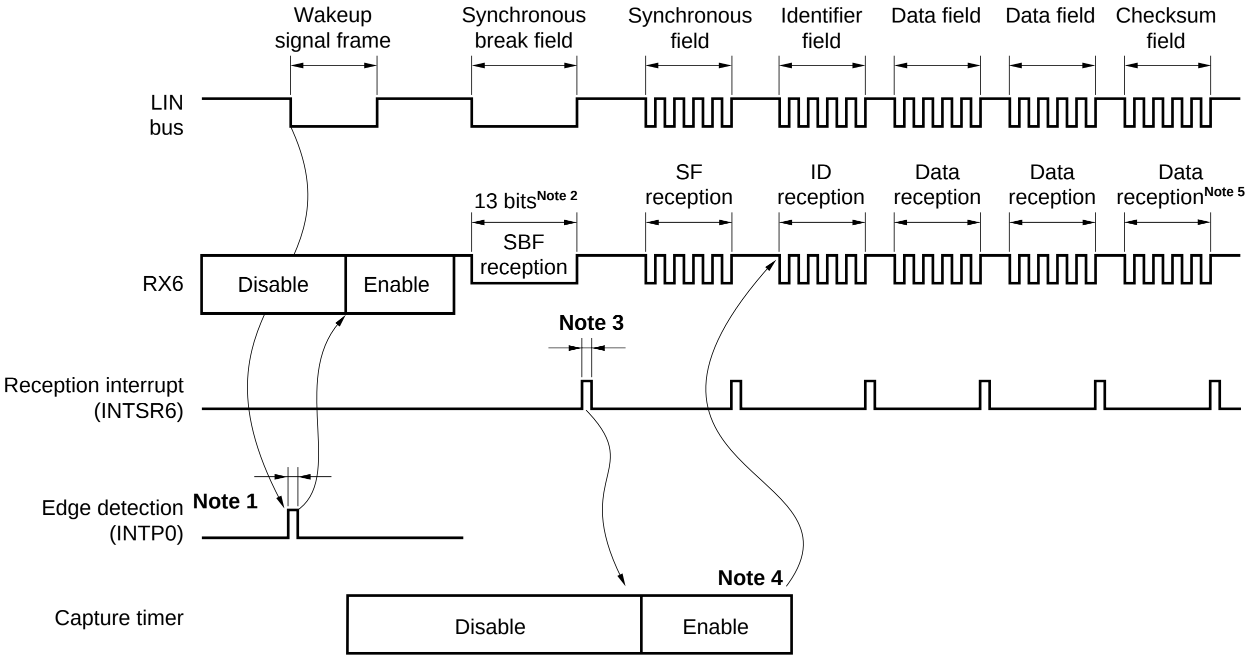
**Notes 1.** The wakeup signal frame is substituted by 80H transmission in the 8-bit mode.

**2.** The synchronous break field is output by hardware. The output width is equal to the bit length set by bits 4 to 2 (SBL62 to SBL60) of asynchronous serial interface control register 6 (ASICL6) (see **11.4.2 (2) (h) SBF transmission**).

**3.** INTST6 is output on completion of each transmission. It is also output when SBF is transmitted.

**Remark** The interval between each field is controlled by software.

Figure 11-2. LIN Reception Operation



**Notes 1.** The wakeup signal is detected at the edge of the pin, and enables UART6 and sets the SBF reception mode.

**2.** Reception continues until the STOP bit is detected. When an SBF with low-level data of 11 bits or more has been detected, it is assumed that SBF reception has been completed correctly, and an interrupt request signal is output. If an SBF with low-level data of less than 11 bits has been detected, it is assumed that an SBF reception error has occurred. The interrupt request signal is not output and the SBF reception mode is restored.

**3.** If SBF reception has been completed correctly, an interrupt request signal is output. This SBF reception completion interrupt enables the capture timer. Detection of errors OVE6, PE6, and FE6 is suppressed, and error detection processing of UART communication and data transfer of the shift register and RXB6 is not performed. The shift register holds the reset value FFH.

**4.** Calculate the baud rate error from the bit length of the synchronous field, disable UART6 after SF reception, and then re-set baud rate generator control register 6 (BRGC6).

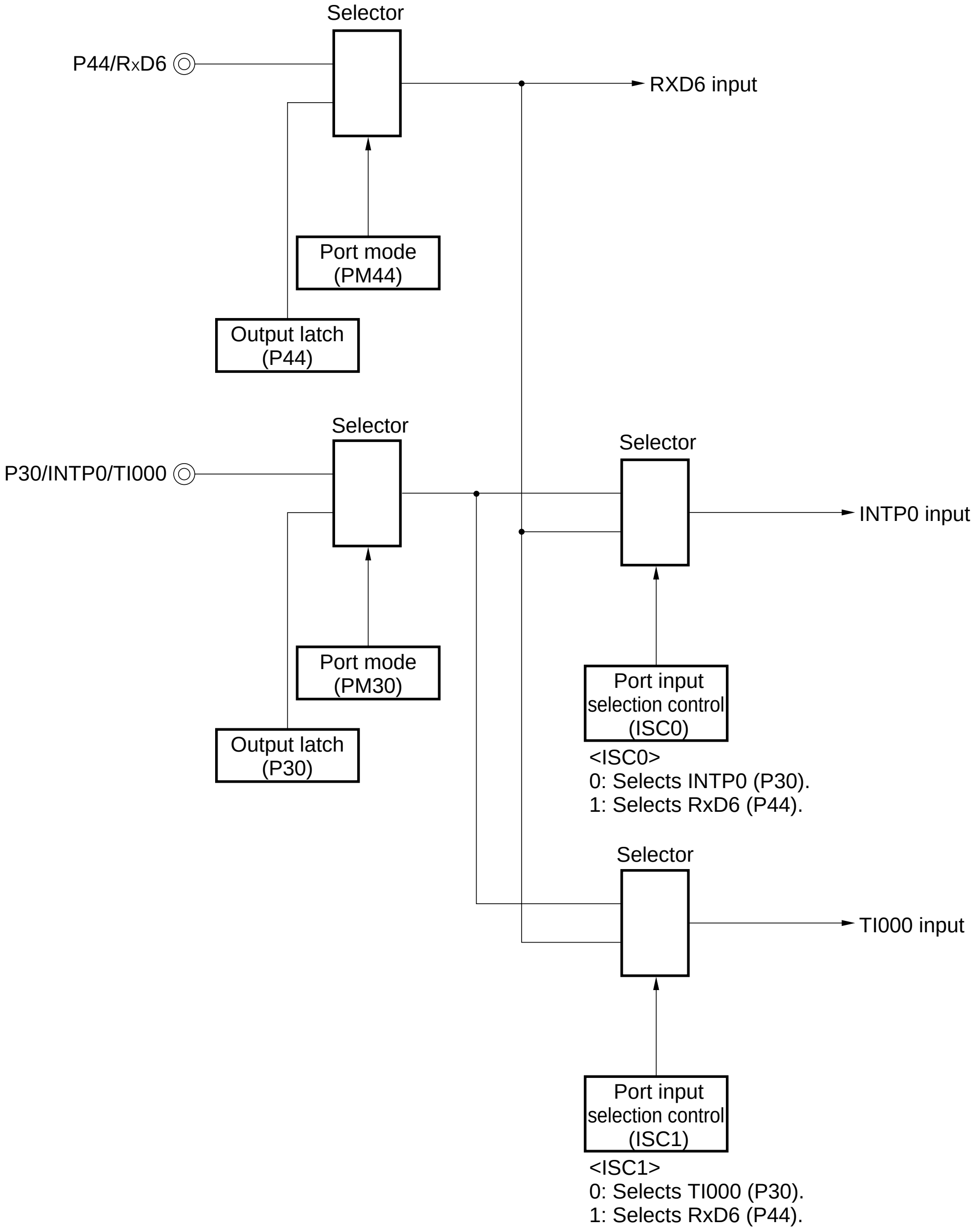
**5.** Distinguish the checksum field by software. Also perform processing by software to initialize UART6 after reception of the checksum field and to set the SBF reception mode again.

To perform a LIN receive operation, use a configuration like the one shown in Figure 11-3.

The wakeup signal transmitted from the LIN master is received by detecting the edge of the external interrupt (INTP0). The length of the synchronous field transmitted from the LIN master can be measured using the external event capture operation of 16-bit timer/event counter 00, and the baud rate error can be calculated.

The input signal of the reception port input (RxD6) can be input to the external interrupt (INTP0) and 16-bit timer/event counter 00 by port input switch control (ISC0/ISC1), without connecting RxD6 and INTP0/TI000 externally.

**Figure 11-3. Port Configuration for LIN Reception Operation**



**Remark** ISC0, ISC1: Bits 0 and 1 of the input switch control register (ISC) (see **Figure 11-11**)

The peripheral functions used in the LIN communication operation are shown below.

<Peripheral functions used>

- External interrupt (INTP0); wakeup signal detection

  Use: Detects the wakeup signal edges and detects start of communication.
- 16-bit timer/event counter 00 (TI000); baud rate error detection

  Use: Detects the baud rate error (measures the TI000 input edge interval in the capture mode) by detecting the synchronous field (SF) length and divides it by the number of bits.
- Serial interface UART6

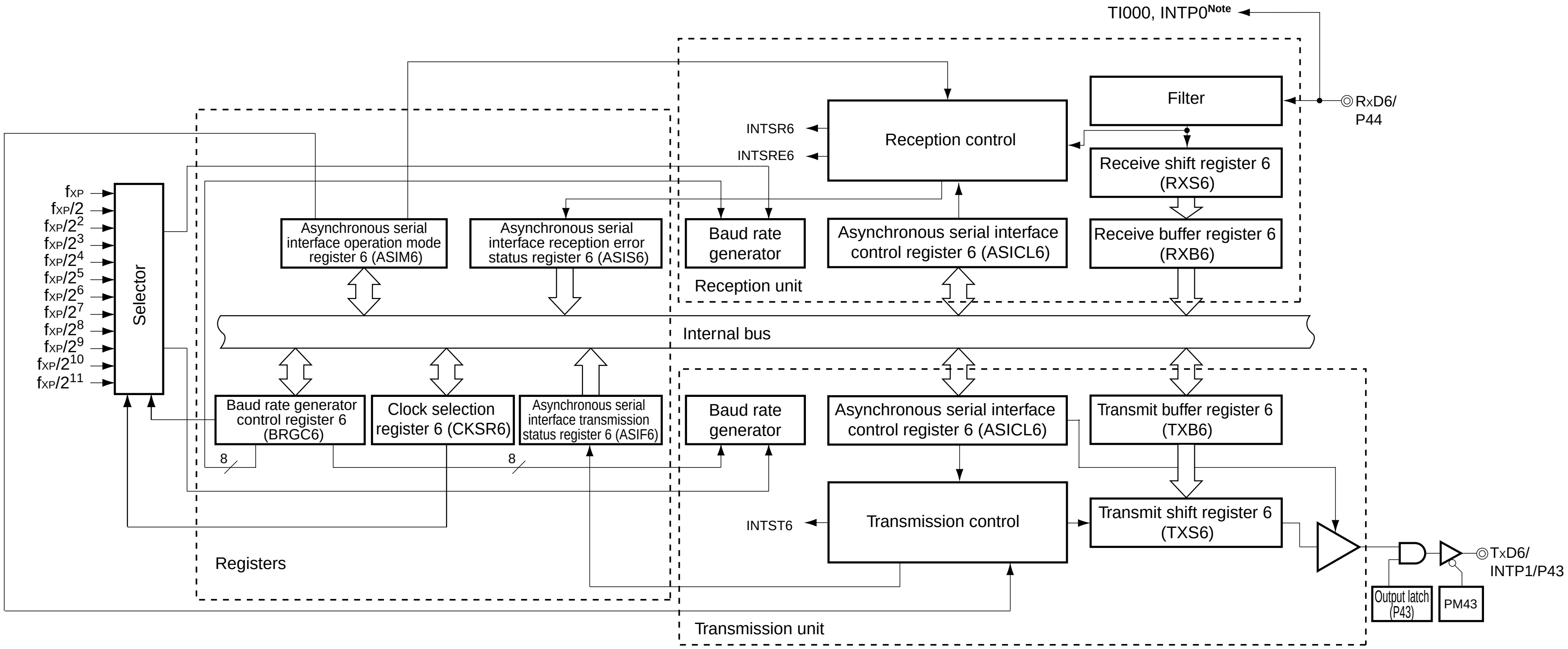
## 11.2 Configuration of Serial Interface UART6

Serial interface UART6 consists of the following hardware.

**Table 11-1. Configuration of Serial Interface UART6**

| Item | Configuration |
|---|---|
| Registers | Receive buffer register 6 (RXB6)<br>Receive shift register 6 (RXS6)<br>Transmit buffer register 6 (TXB6)<br>Transmit shift register 6 (TXS6) |
| Control registers | Asynchronous serial interface operation mode register 6 (ASIM6)<br>Asynchronous serial interface reception error status register 6 (ASIS6)<br>Asynchronous serial interface transmission status register 6 (ASIF6)<br>Clock selection register 6 (CKSR6)<br>Baud rate generator control register 6 (BRGC6)<br>Asynchronous serial interface control register 6 (ASICL6)<br>Input switch control register (ISC)<br>Port mode register 4 (PM4)<br>Port register 4 (P4) |

**Figure 11-4. Block Diagram of Serial Interface UART6**


TI000, INTP0Note
Filter
RxD6/
P44
INTSR6
INTSRE6
Reception control
Receive shift register 6
(RXS6)
fXP
fXP/2
fXP/2^2
fXP/2^3
fXP/2^4
fXP/2^5
fXP/2^6
fXP/2^7
fXP/2^8
fXP/2^9
fXP/2^10
fXP/2^11
Selector
Asynchronous serial interface operation mode register 6 (ASIM6)
Asynchronous serial interface reception error status register 6 (ASIS6)
Baud rate generator
Asynchronous serial interface control register 6 (ASICL6)
Receive buffer register 6 (RXB6)
Reception unit
Internal bus
Baud rate generator control register 6 (BRGC6)
Clock selection register 6 (CKSR6)
Asynchronous serial interface transmission status register 6 (ASIF6)
Baud rate generator
Asynchronous serial interface control register 6 (ASICL6)
Transmit buffer register 6 (TXB6)
8
8
INTST6
Transmission control
Transmit shift register 6 (TXS6)
TxD6/
INTP1/P43
Registers
Output latch (P43)
PM43
Transmission unit

**Note** Selectable with input switch control register (ISC).

**(1) Receive buffer register 6 (RXB6)**

This 8-bit register stores parallel data converted by receive shift register 6 (RXS6).

Each time 1 byte of data has been received, new receive data is transferred to this register from receive shift register 6 (RXS6). If the data length is set to 7 bits, data is transferred as follows.

- In LSB-first reception, the receive data is transferred to bits 0 to 6 of RXB6 and the MSB of RXB6 is always 0.
- In MSB-first reception, the receive data is transferred to bits 7 to 1 of RXB6 and the LSB of RXB6 is always 0.

If an overrun error (OVE6) occurs, the receive data is not transferred to RXB6.

RXB6 can be read by an 8-bit memory manipulation instruction. No data can be written to this register.

Generation of reset signal sets this register to FFH.

**(2) Receive shift register 6 (RXS6)**

This register converts the serial data input to the RxD6 pin into parallel data.

RXS6 cannot be directly manipulated by a program.

**(3) Transmit buffer register 6 (TXB6)**

This buffer register is used to set transmit data. Transmission is started when data is written to TXB6.

If the data length is set to 7 bits:

- In LSB-fast transmission, data is transferred to bits 0 to 6 of TXB6, and the MSB of TXB6 is not transmitted.
- In MSB-fast transmission, data is transferred to bits 7 to 1 of TXB6, and the LSB of TXB6 is not transmitted.

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

Generation of reset signal sets this register to FFH.

**Cautions 1. Do not write data to TXB6 when bit 1 (TXBF6) of asynchronous serial interface transmission status register 6 (ASIF6) is 1.**

**2. Do not refresh (write the same value to) TXB6 by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of asynchronous serial interface operation mode register 6 (ASIM6) are 1 or when bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 are 1).**

**(4) Transmit shift register 6 (TXS6)**

This register transmits the data transferred from TXB6 from the TxD6 pin as serial data. Data is transferred from TXB6 immediately after TXB6 is written for the first transmission, or immediately before INTST6 occurs after one frame was transmitted for continuous transmission. Data is transferred from TXB6 and transmitted from the TxD6 pin at the falling edge of the base clock.

TXS6 cannot be directly manipulated by a program.

## 11.3 Registers Controlling Serial Interface UART6

Serial interface UART6 is controlled by the following nine registers.

- Asynchronous serial interface operation mode register 6 (ASIM6)
- Asynchronous serial interface reception error status register 6 (ASIS6)
- Asynchronous serial interface transmission status register 6 (ASIF6)
- Clock selection register 6 (CKSR6)
- Baud rate generator control register 6 (BRGC6)
- Asynchronous serial interface control register 6 (ASICL6)
- Input switch control register (ISC)
- Port mode register 4 (PM4)
- Port register 4 (P4)

### (1) Asynchronous serial interface operation mode register 6 (ASIM6)

This 8-bit register controls the serial communication operations of serial interface UART6.
This register can be set by a 1-bit or 8-bit memory manipulation instruction.
Generation of reset signal sets this register to 01H.

**Remark** ASIM6 can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1 or bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1).

**Figure 11-5. Format of Asynchronous Serial Interface Operation Mode Register 6 (ASIM6) (1/2)**

Address: FF90H After reset: 01H R/W

| Symbol | <7> | <6> | <5> | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIM6 | POWER6 | TXE6 | RXE6 | PS61 | PS60 | CL6 | SL6 | ISRM6 |

| POWER6 | Enabling/disabling operation of internal operation clock |
|---|---|
| 0[Note 1] | Disable operation of the internal operation clock (fixes the clock to low level) and asynchronously resets the internal circuit[Note 2]. |
| 1[Note 3] | Enable operation of the internal operation clock |

| TXE6 | Enabling/disabling transmission |
|---|---|
| 0 | Disable transmission (synchronously reset the transmission circuit). |
| 1 | Enable transmission |

**Notes 1.** The output of the TxD6 pin goes high and the input from the RxD6 pin is fixed to the high level when POWER6 is cleared to 0 during a transmission 0.

**2.** Asynchronous serial interface reception error status register 6 (ASIS6), asynchronous serial interface transmission status register 6 (ASIF6), bit 7 (SBRF6) and bit 6 (SBRT6) of asynchronous serial interface control register 6 (ASICL6), and receive buffer register 6 (RXB6) are reset.

**3.** Operation of the 8-bit counter output is enabled at the second base clock after 1 is written to the POWER6 bit.

Figure 11-5. Format of Asynchronous Serial Interface Operation Mode Register 6 (ASIM6) (2/2)

| RXE6 | Enabling/disabling reception |
|---|---|
| 0 | Disable reception (synchronously reset the reception circuit). |
| 1 | Enable reception |

| PS61 | PS60 | Transmission operation | Reception operation |
|---|---|---|---|
| 0 | 0 | Parity bit not output. | Reception without parity |
| 0 | 1 | Output 0 parity. | Reception as 0 parity[Note] |
| 1 | 0 | Output odd parity. | Judge as odd parity. |
| 1 | 1 | Output even parity. | Judge as even parity. |

| CL6 | Specification of character length of transmit/receive data |
|---|---|
| 0 | Character length of data = 7 bits |
| 1 | Character length of data = 8 bits |

| SL6 | Specification of number of stop bits of transmit data |
|---|---|
| 0 | Number of stop bits = 1 |
| 1 | Number of stop bits = 2 |

| ISRM6 | Enabling/disabling occurrence of reception completion interrupt in case of error |
|---|---|
| 0 | “INTSRE6” occurs in case of error (at this time, INTSR6 does not occur). |
| 1 | “INTSR6” occurs in case of error (at this time, INTSRE6 does not occur). |

**Note** If “reception as 0 parity” is selected, the parity is not judged. Therefore, bit 2 (PE6) of asynchronous serial interface reception error status register 6 (ASIS6) is not set and the error interrupt does not occur.

**Cautions 1. At startup, set POWER6 to 1 and then set TXE6 to 1. To stop the operation, clear TXE6 to 0, and then clear POWER6 to 0.**

**2. At startup, set POWER6 to 1 and then set RXE6 to 1. To stop the operation, clear RXE6 to 0, and then clear POWER6 to 0.**

**3. Set POWER6 to 1 and then set RXE6 to 1 while a high level is input to the RxD6 pin. If POWER6 is set to 1 and RXE6 is set to 1 while a low level is input, reception is started.**

**4. Clear the TXE6 and RXE6 bits to 0 before rewriting the PS61, PS60, and CL6 bits.**

**5. Fix the PS61 and PS60 bits to 0 when mounting the device on LIN.**

**6. Make sure that TXE6 = 0 when rewriting the SL6 bit. Reception is always performed with “the number of stop bits = 1”, and therefore, is not affected by the set value of the SL6 bit.**

**7. Make sure that RXE6 = 0 when rewriting the ISRM6 bit.**

### (2) Asynchronous serial interface reception error status register 6 (ASIS6)

This register indicates an error status on completion of reception by serial interface UART6. It includes three error flag bits (PE6, FE6, OVE6).

This register is read-only by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H if bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 0. 00H is read when this register is read.

**Figure 11-6. Format of Asynchronous Serial Interface Reception Error Status Register 6 (ASIS6)**

Address: FF93H After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIS6 | 0 | 0 | 0 | 0 | 0 | PE6 | FE6 | OVE6 |

| PE6 | Status flag indicating parity error |
|---|---|
| 0 | If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read |
| 1 | If the parity of transmit data does not match the parity bit on completion of reception |

| FE6 | Status flag indicating framing error |
|---|---|
| 0 | If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read |
| 1 | If the stop bit is not detected on completion of reception |

| OVE6 | Status flag indicating overrun error |
|---|---|
| 0 | If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read |
| 1 | If receive data is set to the RXB register and the next reception operation is completed before the data is read. |

**Cautions 1. The operation of the PE6 bit differs depending on the set values of the PS61 and PS60 bits of asynchronous serial interface operation mode register 6 (ASIM6).**

**2. The first bit of the receive data is checked as the stop bit, regardless of the number of stop bits.**

**3. If an overrun error occurs, the next receive data is not written to receive buffer register 6 (RXB6) but discarded.**

### (3) Asynchronous serial interface transmission status register 6 (ASIF6)

This register indicates the status of transmission by serial interface UART6. It includes two status flag bits (TXBF6 and TXSF6).

Transmission can be continued without disruption even during an interrupt period, by writing the next data to the TXB6 register after data has been transferred from the TXB6 register to the TXS6 register.

This register is read-only by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H if bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 0.

**Figure 11-7. Format of Asynchronous Serial Interface Transmission Status Register 6 (ASIF6)**

Address: FF95H After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIF6 | 0 | 0 | 0 | 0 | 0 | 0 | TXBF6 | TXSF6 |

| TXBF6 | Transmit buffer data flag |
|---|---|
| 0 | If POWER6 = 0 or TXE6 = 0, or if data is transferred to transmit shift register 6 (TXS6) |
| 1 | If data is written to transmit buffer register 6 (TXB6) (if data exists in TXB6) |

| TXSF6 | Transmit shift register data flag |
|---|---|
| 0 | If POWER6 = 0 or TXE6 = 0, or if the next data is not transferred from transmit buffer register 6 (TXB6) after completion of transfer |
| 1 | If data is transferred from transmit buffer register 6 (TXB6) (if data transmission is in progress) |

**Cautions 1. To transmit data continuously, write the first transmit data (first byte) to the TXB6 register. Be sure to check that the TXBF6 flag is “0”. If so, write the next transmit data (second byte) to the TXB6 register. If data is written to the TXB6 register while the TXBF6 flag is “1”, the transmit data cannot be guaranteed.**

**2. To initialize the transmission unit upon completion of continuous transmission, be sure to check that the TXSF6 flag is “0” after generation of the transmission completion interrupt, and then execute initialization. If initialization is executed while the TXSF6 flag is “1”, the transmit data cannot be guaranteed.**

### (4) Clock selection register 6 (CKSR6)

This register selects the base clock of serial interface UART6.

CKSR6 can be set by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Remark** CKSR6 can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1 or bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1).

**Figure 11-8. Format of Clock Selection Register 6 (CKSR6)**

Address: FF96H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CKSR6 | 0 | 0 | 0 | 0 | TPS63 | TPS62 | TPS61 | TPS60 |

| TPS63 | TPS62 | TPS61 | TPS60 | Base clock ($f_{XCLK6}$) selection |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $f_{XP}$ (10 MHz) |
| 0 | 0 | 0 | 1 | $f_{XP}/2$ (5 MHz) |
| 0 | 0 | 1 | 0 | $f_{XP}/2^2$ (2.5 MHz) |
| 0 | 0 | 1 | 1 | $f_{XP}/2^3$ (1.25 MHz) |
| 0 | 1 | 0 | 0 | $f_{XP}/2^4$ (625 kHz) |
| 0 | 1 | 0 | 1 | $f_{XP}/2^5$ (312.5 kHz) |
| 0 | 1 | 1 | 0 | $f_{XP}/2^6$ (156.25 kHz) |
| 0 | 1 | 1 | 1 | $f_{XP}/2^7$ (78.13 kHz) |
| 1 | 0 | 0 | 0 | $f_{XP}/2^8$ (39.06 kHz) |
| 1 | 0 | 0 | 1 | $f_{XP}/2^9$ (19.53 kHz) |
| 1 | 0 | 1 | 0 | $f_{XP}/2^{10}$ (9.77 kHz) |
| 1 | 0 | 1 | 1 | $f_{XP}/2^{11}$ (4.89 kHz) |
| Other than above | | | | Setting prohibited |

**Caution Make sure POWER6 = 0 when rewriting TPS63 to TPS60.**

**Remarks 1.** Figures in parentheses are for operation with $f_{XP}$ = 10 MHz

**2.** $f_{XP}$: Oscillation frequency of clock to peripheral hardware

### (5) Baud rate generator control register 6 (BRGC6)

This register sets the division value of the 8-bit counter of serial interface UART6.

BRGC6 can be set by an 8-bit memory manipulation instruction.

Generation of reset signal sets this register to FFH.

**Remark** BRGC6 can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1 or bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1).

**Figure 11-9. Format of Baud Rate Generator Control Register 6 (BRGC6)**

Address: FF97H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| BRGC6 | MDL67 | MDL66 | MDL65 | MDL64 | MDL63 | MDL62 | MDL61 | MDL60 |

| MDL67 | MDL66 | MDL65 | MDL64 | MDL63 | MDL62 | MDL61 | MDL60 | k | Output clock selection of 8-bit counter |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | × | × | × | × | Setting prohibited |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | $f_{XCLK6}$/8 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 9 | $f_{XCLK6}$/9 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 10 | $f_{XCLK6}$/10 |
| •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• | •<br>•<br>•<br>•<br>• |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 252 | $f_{XCLK6}$/252 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 253 | $f_{XCLK6}$/253 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 254 | $f_{XCLK6}$/254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 | $f_{XCLK6}$/255 |

**Cautions 1. Make sure that bit 6 (TXE6) and bit 5 (RXE6) of the ASIM6 register = 0 when rewriting the MDL67 to MDL60 bits.**

**2. The baud rate is the output clock of the 8-bit counter divided by 2.**

**Remarks 1.** $f_{XCLK6}$: Frequency of base clock selected by the TPS63 to TPS60 bits of CKSR6 register

**2.** k: Value set by MDL67 to MDL60 bits (k = 8, 9, 10, ..., 255)

**3.** ×: Don’t care

### (6) Asynchronous serial interface control register 6 (ASICL6)

This register controls the serial communication operations of serial interface UART6.

ASICL6 can be set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets this register to 16H.

**Caution ASICL6 can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1 or bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1). However, if the SBRT6 = 1 and SBTT = 1 are set in the refresh operation during the SBF reception (SBRF6 = 1) or SBF transmission (between the SBTT6 setting (1) and the INTST6 occurrence), it triggers the SBF reception and SBF transmission again, so do not set.**

**Figure 11-10. Format of Asynchronous Serial Interface Control Register 6 (ASICL6) (1/2)**

Address: FF98H After reset: 16H R/W[Note]

| Symbol | <7> | <6> | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASICL6 | SBRF6 | SBRT6 | SBTT6 | SBL62 | SBL61 | SBL60 | DIR6 | TXDLV6 |

| SBRF6 | SBF reception status flag |
|---|---|
| 0 | If POWER6 = 0 and RXE6 = 0 or if SBF reception has been completed correctly |
| 1 | SBF reception in progress |

| SBRT6 | SBF reception trigger |
|---|---|
| 0 | – |
| 1 | SBF reception trigger |

| SBTT6 | SBF transmission trigger |
|---|---|
| 0 | – |
| 1 | SBF transmission trigger |

**Note** Bit 7 is read-only.

**Figure 11-10. Format of Asynchronous Serial Interface Control Register 6 (ASICL6) (2/2)**

| SBL62 | SBL61 | SBL60 | SBF transmission output width control |
|---|---|---|---|
| 1 | 0 | 1 | SBF is output with 13-bit length. |
| 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. |

| DIR6 | Specification of first bit |
|---|---|
| 0 | MSB |
| 1 | LSB |

| TXDLV6 | Enabling/disabling inverting TxD6 output |
|---|---|
| 0 | Normal output of TxD6 |
| 1 | Inverted output of TxD6 |

**Cautions**

1. **In the case of an SBF reception error, return the mode to the SBF reception mode again and hold (1) the status of the SBRF6 flag.**
2. **Before setting the SBRT6 bit to 1, make sure that bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1. Moreover, after setting the SBRT6 bit to 1, do not clear the SBRT6 bit to 0 before the SBF reception ends (an interrupt request signal is generated).**
3. **The read value of the SBRT6 bit is always 0. SBRT6 is automatically cleared to 0 after SBF reception has been correctly completed.**
4. **Before setting the SBTT6 bit to 1, make sure that bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1. Moreover, after setting the SBTT6 bit to 1, do not clear the SBTT6 bit to 0 before the SBF transmission ends (an interrupt request signal is generated).**
5. **The read value of the SBTT6 bit is always 0. SBTT6 is automatically cleared to 0 at the end of SBF transmission.**
6. **Before rewriting the DIR6 and TXDLV6 bits, clear the TXE6 and RXE6 bits to 0.**

### (7) Input switch control register (ISC)

The input switch control register (ISC) is used to receive a status signal transmitted from the master during LIN (Local Interconnect Network) reception. The input signal is switched by setting ISC.

This register can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Figure 11-11. Format of Input Switch Control Register (ISC)**

Address: FF8CH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ISC | 0 | 0 | 0 | 0 | 0 | 0 | ISC1 | ISC0 |

| ISC1 | TI000 input source selection |
|---|---|
| 0 | TI000 (P30) |
| 1 | RxD6 (P44) |

| ISC0 | INTP0 input source selection |
|---|---|
| 0 | INTP0 (P30) |
| 1 | RxD6 (P44) |

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

This register sets port 4 input/output in 1-bit units.

When using the P43/TxD6/INTP1 pin for serial interface data output, clear PM43 to 0 and set the output latch of P43 to 1.

When using the P44/RxD6 pin for serial interface data input, set PM44 to 1. The output latch of P44 at this time may be 0 or 1.

PM4 can be set by a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets this register to FFH.

**Figure 11-12. Format of Port Mode Register 4 (PM4)**

Address: FF24H After reset: FFH R/W

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

| PM4n | P4n pin I/O mode selection (n = 0 to 7) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

## 11.4 Operation of Serial Interface UART6

Serial interface UART6 has the following two modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode

### 11.4.1 Operation stop mode

In this mode, serial communication cannot be executed; therefore, the power consumption can be reduced. In addition, the pins can be used as ordinary port pins in this mode. To set the operation stop mode, clear bits 7, 6, and 5 (POWER6, TXE6, and RXE6) of ASIM6 to 0.

**(1) Register used**

The operation stop mode is set by asynchronous serial interface operation mode register 6 (ASIM6).
ASIM6 can be set by a 1-bit or 8-bit memory manipulation instruction.
Generation of reset signal sets this register to 01H.

Address: FF90H After reset: 01H R/W

| Symbol | <7> | <6> | <5> | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIM6 | POWER6 | TXE6 | RXE6 | PS61 | PS60 | CL6 | SL6 | ISRM6 |

| POWER6 | Enabling/disabling operation of internal operation clock |
|---|---|
| 0[Note 1] | Disable operation of the internal operation clock (fix the clock to low level) and asynchronously reset the internal circuit[Note 2]. |

| TXE6 | Enabling/disabling transmission |
|---|---|
| 0 | Disable transmission operation (synchronously reset the transmission circuit). |

| RXE6 | Enabling/disabling reception |
|---|---|
| 0 | Disable reception (synchronously reset the reception circuit). |

**Notes 1.** The output of the TxD6 pin goes high and the input from the RxD6 pin is fixed to high level when POWER6 = 0 during a transmission.

**2.** Asynchronous serial interface reception error status register 6 (ASIS6), asynchronous serial interface transmission status register 6 (ASIF6), bit 7 (SBRF6) and bit 6 (SBRT6) of asynchronous serial interface control register 6 (ASICL6), and receive buffer register 6 (RXB6) are reset.

**Caution Clear POWER6 to 0 after clearing TXE6 and RXE6 to 0 to set the operation stop mode.**
**To start the operation, set POWER6 to 1, and then set TXE6 and RXE6 to 1.**

**Remark** To use the RxD6/P44 and TxD6/INTP1/P43 pins as general-purpose port pins, see **CHAPTER 4 PORT FUNCTIONS**.

### 11.4.2 Asynchronous serial interface (UART) mode

In this mode, data of 1 byte is transmitted/received following a start bit, and a full-duplex operation can be performed.

A dedicated UART baud rate generator is incorporated, so that communication can be executed at a wide range of baud rates.

**(1) Registers used**

- Asynchronous serial interface operation mode register 6 (ASIM6)
- Asynchronous serial interface reception error status register 6 (ASIS6)
- Asynchronous serial interface transmission status register 6 (ASIF6)
- Clock selection register 6 (CKSR6)
- Baud rate generator control register 6 (BRGC6)
- Asynchronous serial interface control register 6 (ASICL6)
- Input switch control register (ISC)
- Port mode register 4 (PM4)
- Port register 4 (P4)

The basic procedure of setting an operation in the UART mode is as follows.

<1> Set the CKSR6 register (see **Figure 11-8**).
<2> Set the BRGC6 register (see **Figure 11-9**).
<3> Set bits 0 to 4 (ISRM6, SL6, CL6, PS60, PS61) of the ASIM6 register (see **Figure 11-5**).
<4> Set bits 0 and 1 (TXDLV6, DIR6) of the ASICL6 register (see **Figure 11-10**).
<5> Set bit 7 (POWER6) of the ASIM6 register to 1.
<6> Set bit 6 (TXE6) of the ASIM6 register to 1. → Transmission is enabled.
Set bit 5 (RXE6) of the ASIM6 register to 1. → Reception is enabled.
<7> Write data to transmit buffer register 6 (TXB6). → Data transmission is started.

**Caution Take relationship with the other party of communication into consideration when setting the port mode register and port register.**

The relationship between the register settings and pins is shown below.

**Table 11-2. Relationship Between Register Settings and Pins**

| POWER6 | TXE6 | RXE6 | PM43 | P43 | PM44 | P44 | UART6 Operation | Pin Function | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | TxD6/INTP1/P43 | RxD6/P44 |
| 0 | 0 | 0 | ×Note | ×Note | ×Note | ×Note | Stop | P43 | P44 |
| 1 | 0 | 1 | ×Note | ×Note | 1 | × | Reception | P43 | RxD6 |
| | 1 | 0 | 0 | 1 | ×Note | ×Note | Transmission | TxD6 | P44 |
| | 1 | 1 | 0 | 1 | 1 | × | Transmission/ reception | TxD6 | RxD6 |

**Note** Can be set as port function.

**Remark**

| | |
|---|---|
| ×: | don’t care |
| POWER6: | Bit 7 of asynchronous serial interface operation mode register 6 (ASIM6) |
| TXE6: | Bit 6 of ASIM6 |
| RXE6: | Bit 5 of ASIM6 |
| PM4×: | Port mode register |
| P4×: | Port output latch |

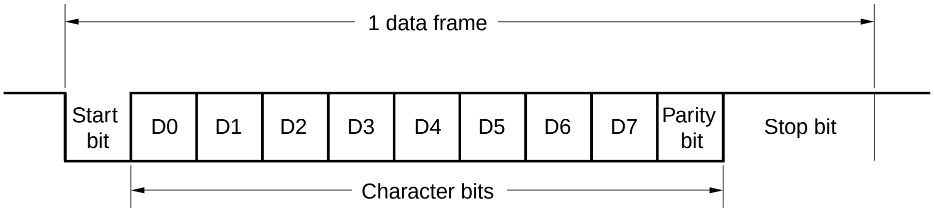
### (2) Communication operation

#### (a) Format and waveform example of normal transmit/receive data

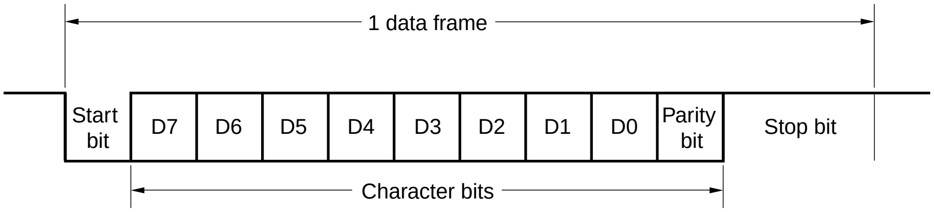
Figures 11-13 and 11-14 show the format and waveform example of the normal transmit/receive data.

**Figure 11-13. Format of Normal UART Transmit/Receive Data**

**1. LSB-first transmission/reception**



**2. MSB-first transmission/reception**



One data frame consists of the following bits.

- Start bit ... 1 bit
- Character bits ... 7 or 8 bits
- Parity bit ... Even parity, odd parity, 0 parity, or no parity
- Stop bit ... 1 or 2 bits

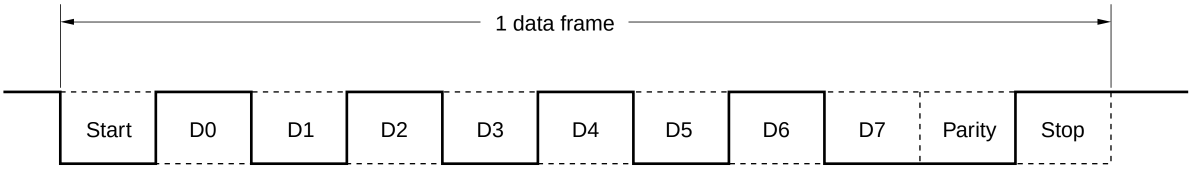
The character bit length, parity, and stop bit length in one data frame are specified by asynchronous serial interface operation mode register 6 (ASIM6).

Whether data is communicated with the LSB or MSB first is specified by bit 1 (DIR6) of asynchronous serial interface control register 6 (ASICL6).

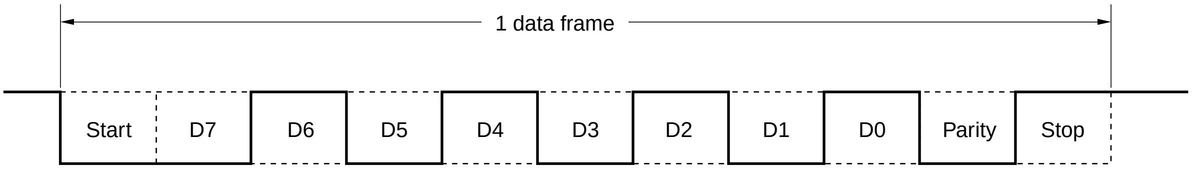
Whether the TxD6 pin outputs normal or inverted data is specified by bit 0 (TXDLV6) of ASICL6.

**Figure 11-14. Example of Normal UART Transmit/Receive Data Waveform**

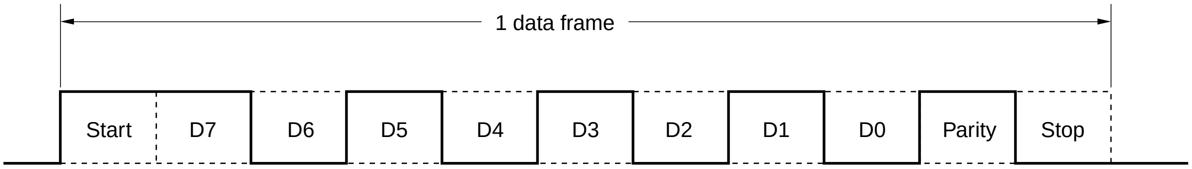
1. **Data length: 8 bits, LSB first, Parity: Even parity, Stop bit: 1 bit, Communication data: 55H**



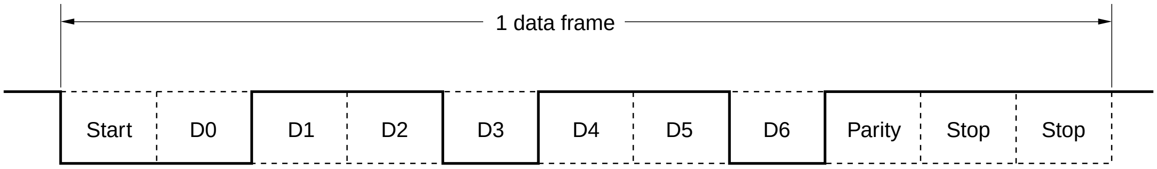
2. **Data length: 8 bits, MSB first, Parity: Even parity, Stop bit: 1 bit, Communication data: 55H**



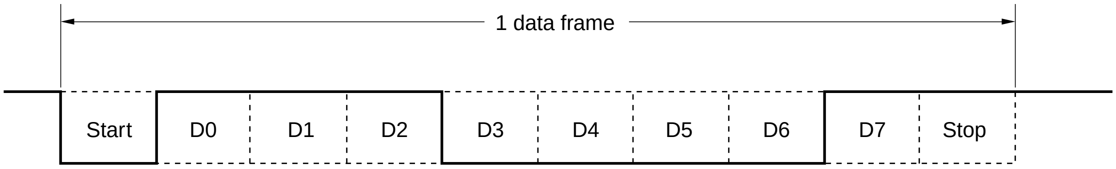
3. **Data length: 8 bits, MSB first, Parity: Even parity, Stop bit: 1 bit, Communication data: 55H, TxD6 pin inverted output**



4. **Data length: 7 bits, LSB first, Parity: Odd parity, Stop bit: 2 bits, Communication data: 36H**



5. **Data length: 8 bits, LSB first, Parity: None, Stop bit: 1 bit, Communication data: 87H**



### (b) Parity types and operation

The parity bit is used to detect a bit error in communication data. Usually, the same type of parity bit is used on both the transmission and reception sides. With even parity and odd parity, a 1-bit (odd number) error can be detected. With zero parity and no parity, an error cannot be detected.

**Caution Fix the PS61 and PS60 bits to 0 when the device is incorporated in LIN.**

#### (i) Even parity

- Transmission

  Transmit data, including the parity bit, is controlled so that the number of bits that are “1” is even.
  The value of the parity bit is as follows.

  If transmit data has an odd number of bits that are “1”: 1
  If transmit data has an even number of bits that are “1”: 0

- Reception

  The number of bits that are “1” in the receive data, including the parity bit, is counted. If it is odd, a parity error occurs.

#### (ii) Odd parity

- Transmission

  Unlike even parity, transmit data, including the parity bit, is controlled so that the number of bits that are “1” is odd.

  If transmit data has an odd number of bits that are “1”: 0
  If transmit data has an even number of bits that are “1”: 1

- Reception

  The number of bits that are “1” in the receive data, including the parity bit, is counted. If it is even, a parity error occurs.

#### (iii) 0 parity

The parity bit is cleared to 0 when data is transmitted, regardless of the transmit data.
The parity bit is not detected when the data is received. Therefore, a parity error does not occur regardless of whether the parity bit is “0” or “1”.

#### (iv) No parity

No parity bit is appended to the transmit data.
Reception is performed assuming that there is no parity bit when data is received. Because there is no parity bit, a parity error does not occur.

### (c) Normal transmission

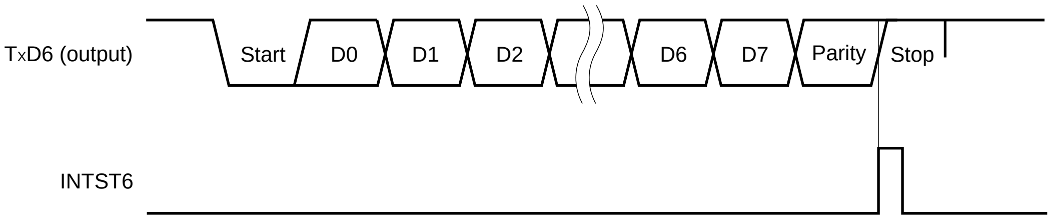
The TxD6 pin outputs a high level when bit 7 (POWER6) of asynchronous serial interface operation mode register 6 (ASIM6) is set to 1. If bit 6 (TXE6) of ASIM6 is then set to 1, transmission is enabled. Transmission can be started by writing transmit data to transmit buffer register 6 (TXB6). The start bit, parity bit, and stop bit are automatically appended to the data.

When transmission is started, the data in TXB6 is transferred to transmit shift register 6 (TXS6). After that, the data is sequentially output from TXS6 to the TxD6 pin. When transmission is completed, the parity and stop bits set by ASIM6 are appended and a transmission completion interrupt request (INTST6) is generated. Transmission is stopped until the data to be transmitted next is written to TXB6.

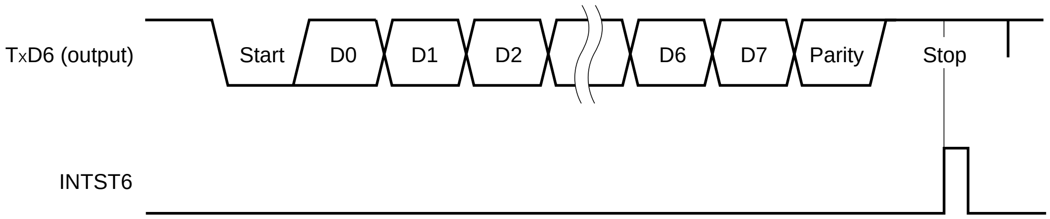
Figure 11-15 shows the timing of the transmission completion interrupt request (INTST6). This interrupt occurs as soon as the last stop bit has been output.

**Figure 11-15. Normal Transmission Completion Interrupt Request Timing**

**1. Stop bit length: 1**



**2. Stop bit length: 2**



**(d) Continuous transmission**

The next transmit data can be written to transmit buffer register 6 (TXB6) as soon as transmit shift register 6 (TXS6) has started its shift operation. Consequently, even while the INTST6 interrupt is being serviced after transmission of one data frame, data can be continuously transmitted and an efficient communication rate can be realized. In addition, the TXB6 register can be efficiently written twice (2 bytes) without having to wait for the transmission time of one data frame, by reading bit 0 (TXSF6) of asynchronous serial interface transmission status register 6 (ASIF6) when the transmission completion interrupt has occurred.

To transmit data continuously, be sure to reference the ASIF6 register to check the transmission status and whether the TXB6 register can be written, and then write the data.

**Cautions 1. The TXBF6 and TXSF6 flags of the ASIF6 register change from "10" to "11", and to "01" during continuous transmission. To check the status, therefore, do not use a combination of the TXBF6 and TXSF6 flags for judgment. Judge whether continuous transmission is possible or not by reading only the TXBF flag.**

**2. When the device is incorporated in a LIN, the continuous transmission function cannot be used. Make sure that asynchronous serial interface transmission status register 6 (ASIF6) is 00H before writing transmit data to transmit buffer register 6 (TXB6).**

| TXBF6 | Writing to TXB6 Register |
|---|---|
| 0 | Writing enabled |
| 1 | Writing disabled |

**Caution To transmit data continuously, write the first transmit data (first byte) to the TXB6 register. Be sure to check that the TXBF6 flag is "0". If so, write the next transmit data (second byte) to the TXB6 register. If data is written to the TXB6 register while the TXBF6 flag is "1", the transmit data cannot be guaranteed.**

The communication status can be checked using the TXSF6 flag.

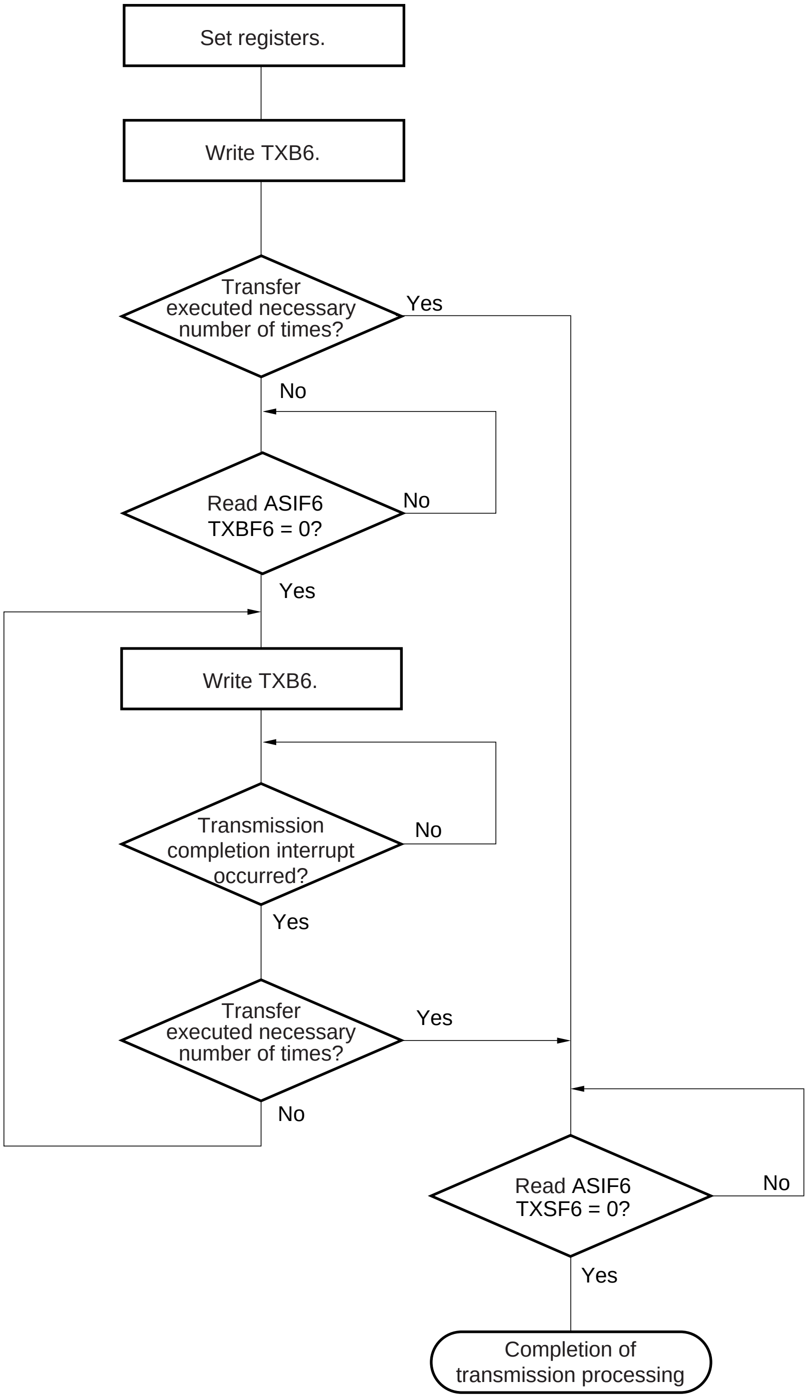
| TXSF6 | Transmission Status |
|---|---|
| 0 | Transmission is completed. |
| 1 | Transmission is in progress. |

**Cautions 1. To initialize the transmission unit upon completion of continuous transmission, be sure to check that the TXSF6 flag is "0" after generation of the transmission completion interrupt, and then execute initialization. If initialization is executed while the TXSF6 flag is "1", the transmit data cannot be guaranteed.**

**2. During continuous transmission, an overrun error may occur, which means that the next transmission was completed before execution of INTST6 interrupt servicing after transmission of one data frame. An overrun error can be detected by developing a program that can count the number of transmit data and by referencing the TXSF6 flag.**

Figure 11-16 shows an example of the continuous transmission processing flow.

**Figure 11-16. Example of Continuous Transmission Processing Flow**


Set registers.
Write TXB6.
Transfer executed necessary number of times?
Yes
No
Read ASIF6 TXBF6 = 0?
No
Yes
Write TXB6.
Transmission completion interrupt occurred?
No
Yes
Transfer executed necessary number of times?
Yes
No
Read ASIF6 TXSF6 = 0?
No
Yes
Completion of transmission processing

**Remark** TXB6: Transmit buffer register 6

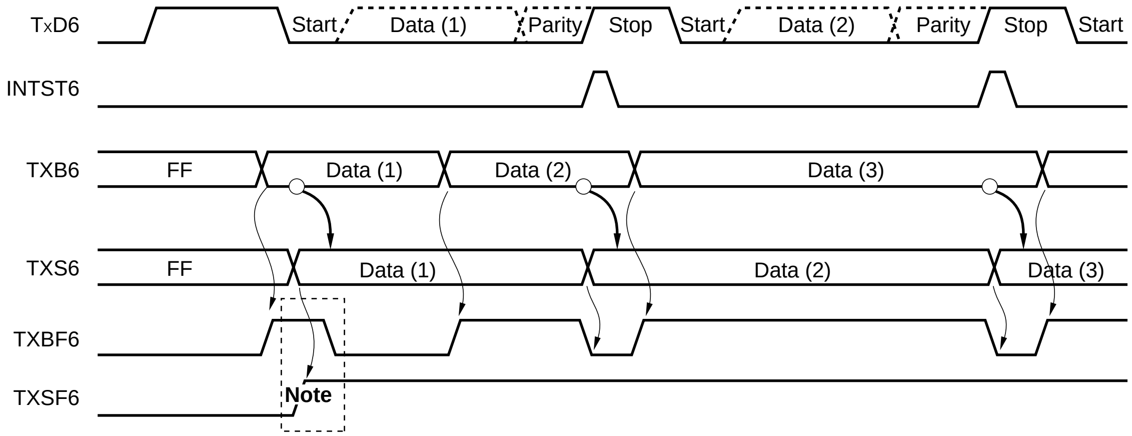
ASIF6: Asynchronous serial interface transmission status register 6

TXBF6: Bit 1 of ASIF6 (transmit buffer data flag)

TXSF6: Bit 0 of ASIF6 (transmit shift register data flag)

Figure 11-17 shows the timing of starting continuous transmission, and Figure 11-18 shows the timing of ending continuous transmission.

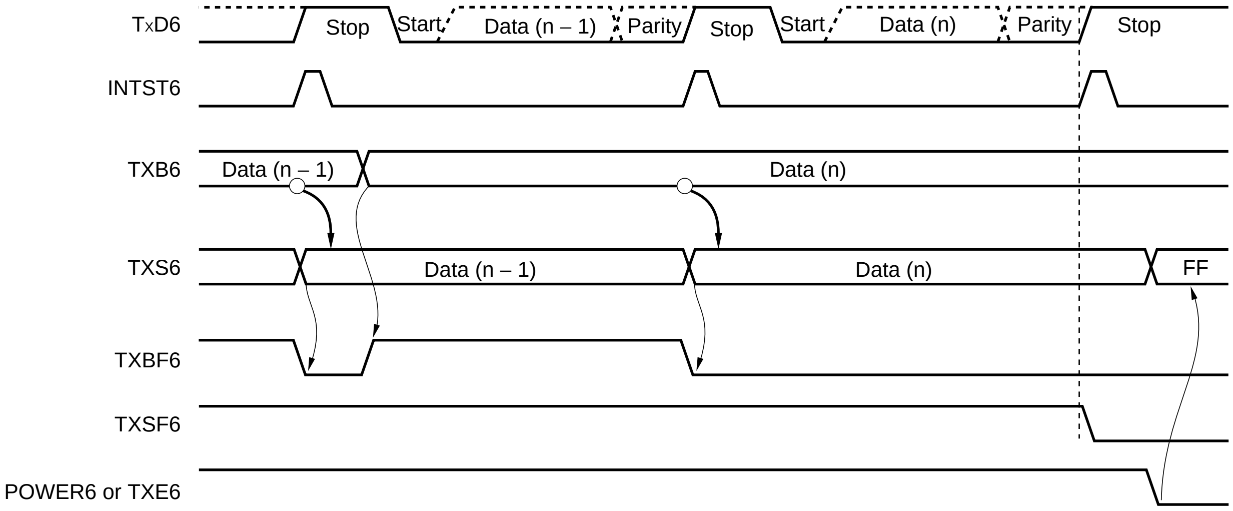
**Figure 11-17. Timing of Starting Continuous Transmission**



**Note** When ASIF6 is read, there is a period in which TXBF6 and TXSF6 = 1, 1. Therefore, judge whether writing is enabled using only the TXBF6 bit.

**Remark** TxD6: TxD6 pin (output)
INTST6: Interrupt request signal
TXB6: Transmit buffer register 6
TXS6: Transmit shift register 6
ASIF6: Asynchronous serial interface transmission status register 6
TXBF6: Bit 1 of ASIF6
TXSF6: Bit 0 of ASIF6

Figure 11-18. Timing of Ending Continuous Transmission



**Remark**

| | |
|---|---|
| TxD6: | TxD6 pin (output) |
| INTST6: | Interrupt request signal |
| TXB6: | Transmit buffer register 6 |
| TXS6: | Transmit shift register 6 |
| ASIF6: | Asynchronous serial interface transmission status register 6 |
| TXBF6: | Bit 1 of ASIF6 |
| TXSF6: | Bit 0 of ASIF6 |
| POWER6: | Bit 7 of asynchronous serial interface operation mode register (ASIM6) |
| TXE6: | Bit 6 of asynchronous serial interface operation mode register (ASIM6) |

**(e) Normal reception**

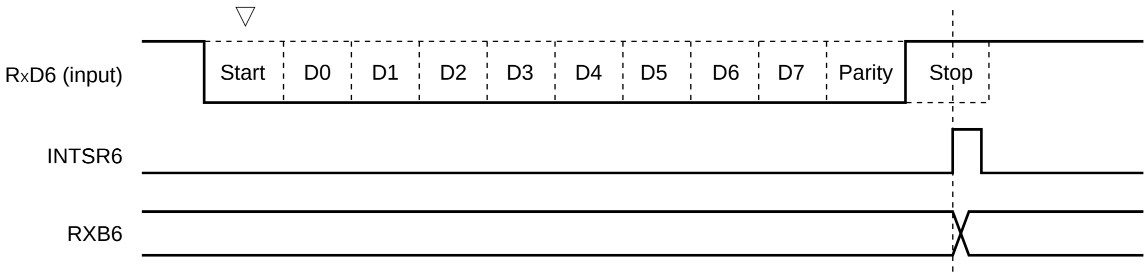
Reception is enabled and the RxD6 pin input is sampled when bit 7 (POWER6) of asynchronous serial interface operation mode register 6 (ASIM6) is set to 1 and then bit 5 (RXE6) of ASIM6 is set to 1.

The 8-bit counter of the baud rate generator starts counting when the falling edge of the RxD6 pin input is detected. When the set value of baud rate generator control register 6 (BRGC6) has been counted, the RxD6 pin input is sampled again (▽ in Figure 11-19). If the RxD6 pin is low level at this time, it is recognized as a start bit.

When the start bit is detected, reception is started, and serial data is sequentially stored in the receive shift register (RXS6) at the set baud rate. When the stop bit has been received, the reception completion interrupt (INTSR6) is generated and the data of RXS6 is written to receive buffer register 6 (RXB6). If an overrun error (OVE6) occurs, however, the receive data is not written to RXB6.

Even if a parity error (PE6) occurs while reception is in progress, reception continues to the reception position of the stop bit, and an error interrupt (INTSR6/INTSRE6) is generated on completion of reception.

**Figure 11-19. Reception Completion Interrupt Request Timing**



**Cautions 1. Be sure to read receive buffer register 6 (RXB6) even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the reception error status will persist.**

**2. Reception is always performed with the "number of stop bits = 1". The second stop bit is ignored.**

**3. Be sure to read asynchronous serial interface reception error status register 6 (ASIS6) before reading RXB6.**

**(f) Reception error**

Three types of errors may occur during reception: a parity error, framing error, or overrun error. If the error flag of asynchronous serial interface reception error status register 6 (ASIS6) is set as a result of data reception, a reception error interrupt request (INTSR6/INTSRE6) is generated.

Which error has occurred during reception can be identified by reading the contents of ASIS6 in the reception error interrupt servicing (INTSR6/INTSRE6) (see **Figure 11-6**).

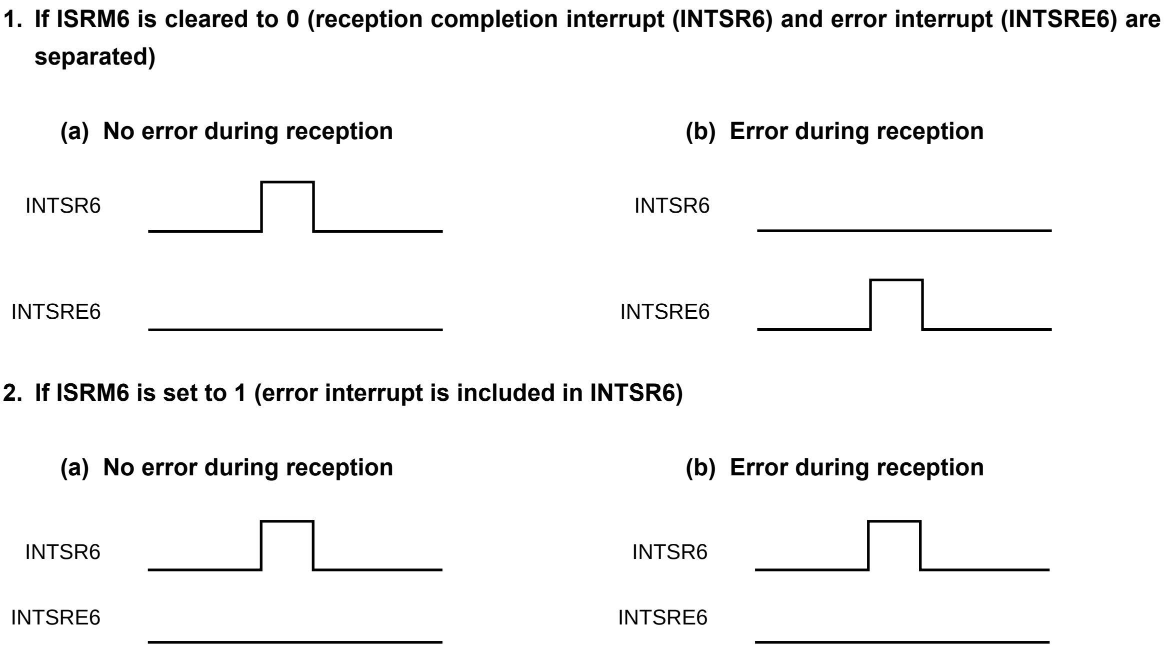
The contents of ASIS6 are reset to 0 when ASIS6 is read.

**Table 11-3. Cause of Reception Error**

| Reception Error | Cause |
|---|---|
| Parity error | The parity specified for transmission does not match the parity of the receive data. |
| Framing error | Stop bit is not detected. |
| Overrun error | Reception of the next data is completed before data is read from receive buffer register 6 (RXB6). |

The error interrupt can be separated into reception completion interrupt (INTSR6) and error interrupt (INTSRE6) by clearing bit 0 (ISRM6) of asynchronous serial interface operation mode register 6 (ASIM6) to 0.

**Figure 11-20. Reception Error Interrupt**



**1. If ISRM6 is cleared to 0 (reception completion interrupt (INTSR6) and error interrupt (INTSRE6) are separated)**

**(a) No error during reception**

**(b) Error during reception**

**2. If ISRM6 is set to 1 (error interrupt is included in INTSR6)**

**(a) No error during reception**

**(b) Error during reception**

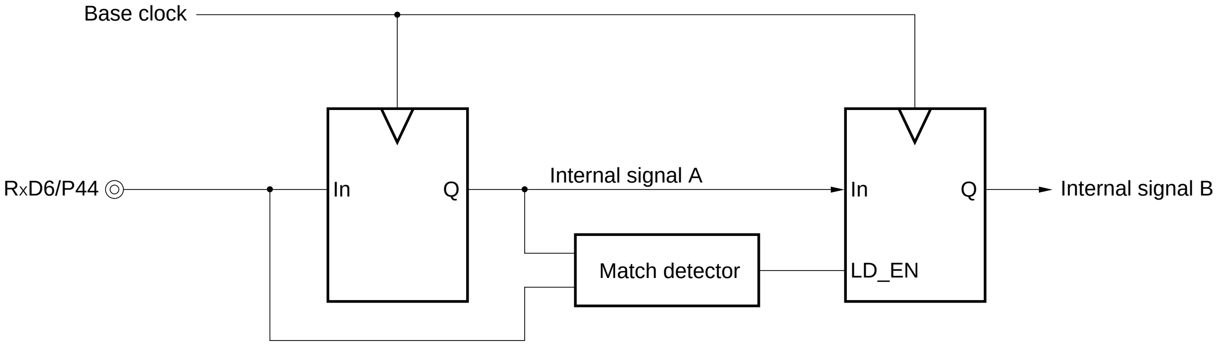
**(g) Noise filter of receive data**

The RxD6 signal is sampled with the base clock output by the prescaler block.

If two sampled values are the same, the output of the match detector changes, and the data is sampled as input data.

Because the circuit is configured as shown in Figure 11-21, the internal processing of the reception operation is delayed by two clocks from the external signal status.

**Figure 11-21. Noise Filter Circuit**



**(h) SBF transmission**

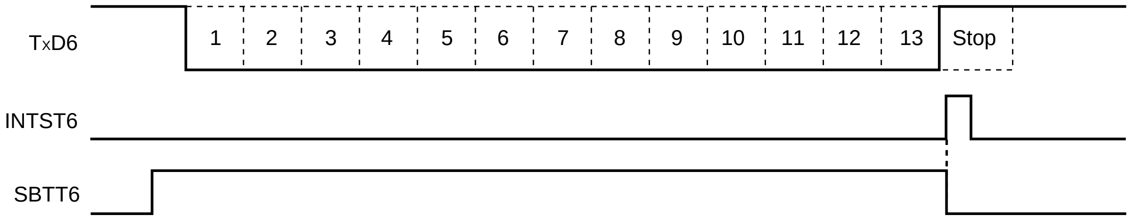
When the device is incorporated in LIN, the SBF (Synchronous Break Field) transmission control function is used for transmission. For the transmission operation of LIN, see **Figure 11-1 LIN Transmission Operation**.

When bit 7 (POWER6) of asynchronous serial interface mode register 6 (ASIM6) is set to 1, the TxD6 pin outputs high level. Next, when bit 6 (TXE6) of ASIM6 is set to 1, the transmission enabled status is entered, and SBF transmission is started by setting bit 5 (SBTT6) of asynchronous serial interface control register 6 (ASICL6) to 1.

Thereafter, a low level of bits 13 to 20 (set by bits 4 to 2 (SBL62 to SBL60) of ASICL6) is output. Following the end of SBF transmission, the transmission completion interrupt request (INTST6) is generated and SBTT6 is automatically cleared. Thereafter, the normal transmission mode is restored.

Transmission is suspended until the data to be transmitted next is written to transmit buffer register 6 (TXB6), or until SBTT6 is set to 1.

**Figure 11-22. SBF Transmission**



**Remark** TxD6: TxD6 pin (output)
INTST6: Transmission completion interrupt request
SBTT6: Bit 5 of asynchronous serial interface control register 6 (ASICL6)

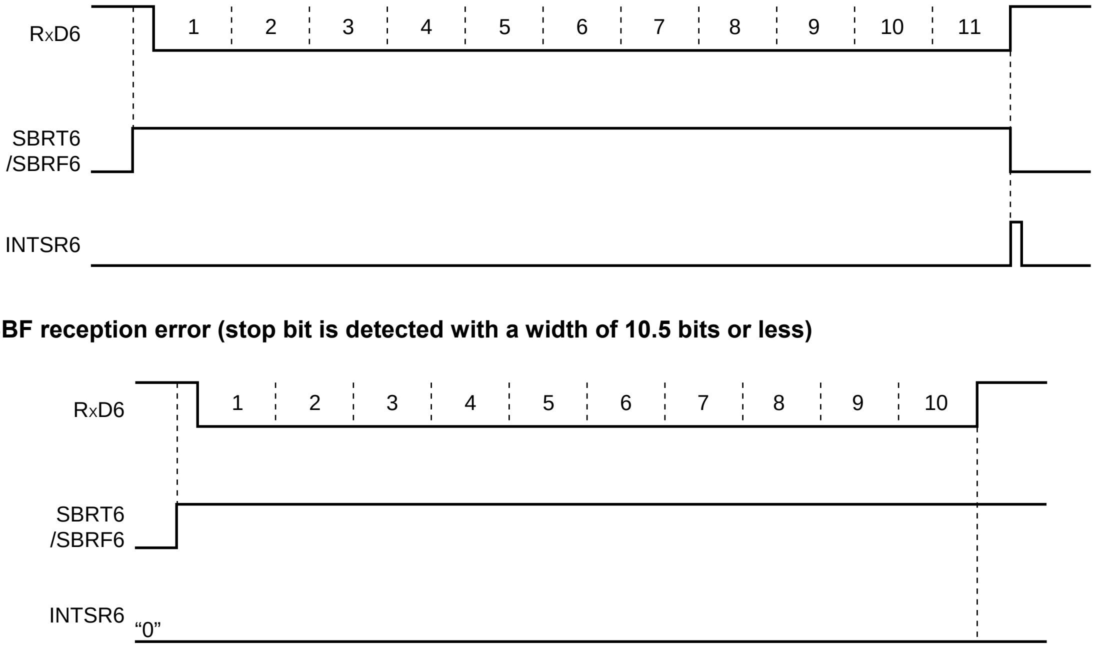
### (i) SBF reception

When the device is incorporated in LIN, the SBF (Synchronous Break Field) reception control function is used for reception. For the reception operation of LIN, see **Figure 11-2 LIN Reception Operation**.

Reception is enabled when bit 7 (POWER6) of asynchronous serial interface operation mode register 6 (ASIM6) is set to 1 and then bit 5 (RXE6) of ASIM6 is set to 1. SBF reception is enabled when bit 6 (SBRT6) of asynchronous serial interface control register 6 (ASICL6) is set to 1. In the SBF reception enabled status, the RxD6 pin is sampled and the start bit is detected in the same manner as the normal reception enable status.

When the start bit has been detected, reception is started, and serial data is sequentially stored in the receive shift register 6 (RXS6) at the set baud rate. When the stop bit is received and if the width of SBF is 11 bits or more, a reception completion interrupt request (INTSR6) is generated as normal processing. At this time, the SBRF6 and SBRT6 bits are automatically cleared, and SBF reception ends. Detection of errors, such as OVE6, PE6, and FE6 (bits 0 to 2 of asynchronous serial interface reception error status register 6 (ASIS6)) is suppressed, and error detection processing of UART communication is not performed. In addition, data transfer between receive shift register 6 (RXS6) and receive buffer register 6 (RXB6) is not performed, and the reset value of FFH is retained. If the width of SBF is 10 bits or less, an interrupt does not occur as error processing after the stop bit has been received, and the SBF reception mode is restored. In this case, the SBRF6 and SBRT6 bits are not cleared.

**Figure 11-23. SBF Reception**



**1. Normal SBF reception (stop bit is detected with a width of more than 10.5 bits)**

**2. SBF reception error (stop bit is detected with a width of 10.5 bits or less)**

**Remark** RxD6: RxD6 pin (input)
SBRT6: Bit 6 of asynchronous serial interface control register 6 (ASICL6)
SBRF6: Bit 7 of ASICL6
INTSR6: Reception completion interrupt request

### 11.4.3 Dedicated baud rate generator

The dedicated baud rate generator consists of a source clock selector and an 8-bit programmable counter, and generates a serial clock for transmission/reception of UART6.

Separate 8-bit counters are provided for transmission and reception.

#### (1) Configuration of baud rate generator

- Base clock

  The clock selected by bits 3 to 0 (TPS63 to TPS60) of clock selection register 6 (CKSR6) is supplied to each module when bit 7 (POWER6) of asynchronous serial interface operation mode register 6 (ASIM6) is 1. This clock is called the base clock and its frequency is called $f_{XCLK6}$. The base clock is fixed to low level when POWER6 = 0.

- Transmission counter

  This counter stops operation, cleared to 0, when bit 7 (POWER6) or bit 6 (TXE6) of asynchronous serial interface operation mode register 6 (ASIM6) is 0.

  It starts counting when POWER6 = 1 and TXE6 = 1.

  The counter is cleared to 0 when the first data transmitted is written to transmit buffer register 6 (TXB6).

  If data are continuously transmitted, the counter is cleared to 0 again when one frame of data has been completely transmitted. If there is no data to be transmitted next, the counter is not cleared to 0 and continues counting until POWER6 or TXE6 is cleared to 0.

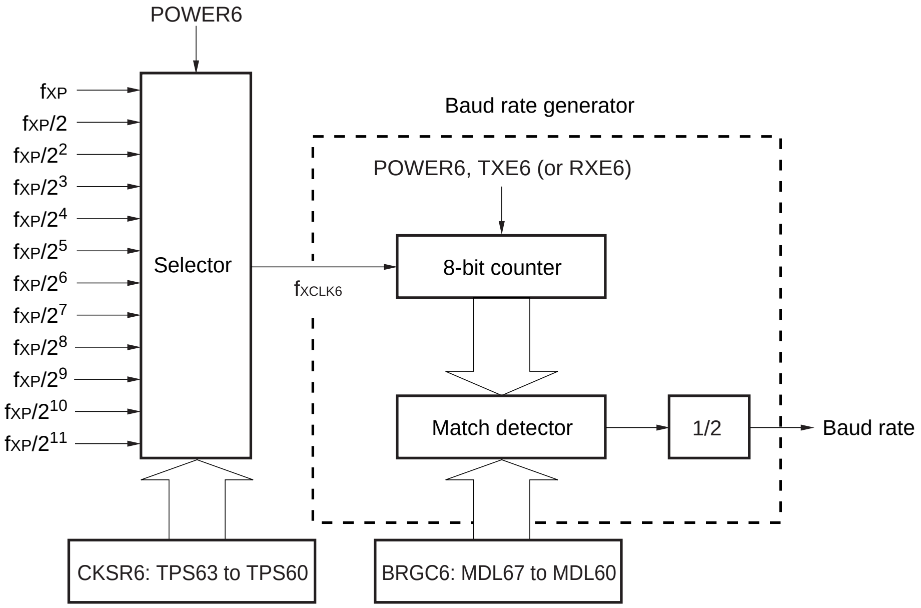
- Reception counter

  This counter stops operation, cleared to 0, when bit 7 (POWER6) or bit 5 (RXE6) of asynchronous serial interface operation mode register 6 (ASIM6) is 0.

  It starts counting when the start bit has been detected.

  The counter stops operation after one frame has been received, until the next start bit is detected.

**Figure 11-24. Configuration of Baud Rate Generator**



**Remark** POWER6: Bit 7 of asynchronous serial interface operation mode register 6 (ASIM6)

TXE6: Bit 6 of ASIM6

RXE6: Bit 5 of ASIM6

CKSR6: Clock selection register 6

BRGC6: Baud rate generator control register 6

### (2) Generation of serial clock

A serial clock can be generated by using clock selection register 6 (CKSR6) and baud rate generator control register 6 (BRGC6).

Select the clock to be input to the 8-bit counter by using bits 3 to 0 (TPS63 to TPS60) of CKSR6.

Bits 7 to 0 (MDL67 to MDL60) of BRGC6 can be used to select the division value of the 8-bit counter.

#### (a) Baud rate

The baud rate can be calculated by the following expression.

- Baud rate = $\frac{f_{XCLK6}}{2 \times k}$ [bps]

$f_{XCLK6}$: Frequency of base clock selected by TPS63 to TPS60 bits of CKSR6 register

k: Value set by MDL67 to MDL60 bits of BRGC6 register (k = 8, 9, 10, ..., 255)

#### (b) Error of baud rate

The baud rate error can be calculated by the following expression.

- Error (%) = $\left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Desired baud rate (correct baud rate)}} - 1\right) \times 100$ [%]

**Cautions 1. Keep the baud rate error during transmission to within the permissible error range at the reception destination.**

**2. Make sure that the baud rate error during reception satisfies the range shown in (4) Permissible baud rate range during reception.**

**Example:** Frequency of base clock = 10 MHz = 10,000,000 Hz
Set value of MDL67 to MDL60 bits of BRGC6 register = 00100001B (k = 33)
Target baud rate = 153600 bps

Baud rate = 10 M/(2 × 33)
= 10000000/(2 × 33) = 151,515 [bps]

Error = (151515/153600 − 1) × 100
= −1.357 [%]

### (3) Example of setting baud rate

**Table 11-4. Set Data of Baud Rate Generator**

| Baud Rate [bps] | $f_{XP}$ = 10.0 MHz | | | | $f_{XP}$ = 8.38 MHz | | | | $f_{XP}$ = 4.19 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPS63 to TPS60 | k | Calculated Value | ERR[%] | TPS63 to TPS60 | k | Calculated Value | ERR[%] | TPS63 to TPS60 | k | Calculated Value | ERR[%] |
| 600 | 6H | 130 | 601 | 0.16 | 6H | 109 | 601 | 0.11 | 5H | 109 | 601 | 0.11 |
| 1200 | 5H | 130 | 1202 | 0.16 | 5H | 109 | 1201 | 0.11 | 4H | 109 | 1201 | 0.11 |
| 2400 | 4H | 130 | 2404 | 0.16 | 4H | 109 | 2403 | 0.11 | 3H | 109 | 2403 | 0.11 |
| 4800 | 3H | 130 | 4808 | 0.16 | 3H | 109 | 4805 | 0.11 | 2H | 109 | 4805 | 0.11 |
| 9600 | 2H | 130 | 9615 | 0.16 | 2H | 109 | 9610 | 0.11 | 1H | 109 | 9610 | 0.11 |
| 10400 | 2H | 120 | 10417 | 0.16 | 2H | 101 | 10371 | 0.28 | 1H | 101 | 10475 | –0.28 |
| 19200 | 1H | 130 | 19231 | 0.16 | 1H | 109 | 19220 | 0.11 | 0H | 109 | 19220 | 0.11 |
| 31250 | 1H | 80 | 31250 | 0.00 | 0H | 134 | 31268 | 0.06 | 0H | 67 | 31268 | 0.06 |
| 38400 | 0H | 130 | 38462 | 0.16 | 0H | 109 | 38440 | 0.11 | 0H | 55 | 38090 | –0.80 |
| 76800 | 0H | 65 | 76923 | 0.16 | 0H | 55 | 76182 | –0.80 | 0H | 27 | 77693 | 1.03 |
| 115200 | 0H | 43 | 116279 | 0.94 | 0H | 36 | 116389 | 1.03 | 0H | 18 | 116389 | 1.03 |
| 153600 | 0H | 33 | 151515 | –1.36 | 0H | 27 | 155185 | 1.03 | 0H | 14 | 149643 | –2.58 |
| 230400 | 0H | 22 | 227272 | –1.36 | 0H | 18 | 232778 | 1.03 | 0H | 9 | 232778 | 1.03 |

**Remark** TPS63 to TPS60: Bits 3 to 0 of clock selection register 6 (CKSR6) (setting of base clock ($f_{XCLK6}$))

k: Value set by MDL67 to MDL60 bits of baud rate generator control register 6 (BRGC6) (k = 8, 9, 10, ..., 255)

$f_{XP}$: Oscillation frequency of clock to peripheral hardware

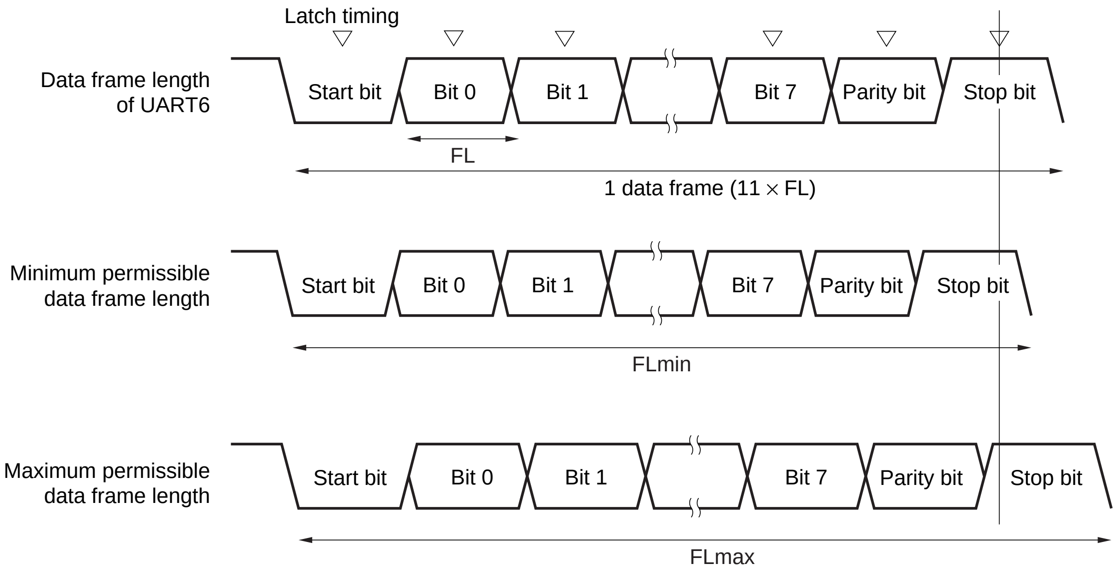
ERR: Baud rate error

### (4) Permissible baud rate range during reception

The permissible error from the baud rate at the transmission destination during reception is shown below.

**Caution Make sure that the baud rate error during reception is within the permissible error range, by using the calculation expression shown below.**

**Figure 11-25. Permissible Baud Rate Range During Reception**



As shown in Figure 11-25, the latch timing of the receive data is determined by the counter set by baud rate generator control register 6 (BRGC6) after the start bit has been detected. If the last data (stop bit) meets this latch timing, the data can be correctly received.

Assuming that 11-bit data is received, the theoretical values can be calculated as follows.

$FL = (Brate)^{-1}$

Brate: Baud rate of UART6

k: Set value of BRGC6

FL: 1-bit data length

Margin of latch timing: 2 clocks

Minimum permissible data frame length: $\text{FLmin} = 11 \times \text{FL} - \frac{k-2}{2k} \times \text{FL} = \frac{21k+2}{2k}\ \text{FL}$

Therefore, the maximum receivable baud rate at the transmission destination is as follows.

$$\text{BRmax} = (\text{FLmin}/11)^{-1} = \frac{22k}{21k+2}\ \text{Brate}$$

Similarly, the maximum permissible data frame length can be calculated as follows.

$$\frac{10}{11} \times \text{FLmax} = 11 \times \text{FL} - \frac{k+2}{2 \times k} \times \text{FL} = \frac{21k-2}{2 \times k}\ \text{FL}$$

$$\text{FLmax} = \frac{21k-2}{20k}\ \text{FL} \times 11$$

Therefore, the minimum receivable baud rate at the transmission destination is as follows.

$$\text{BRmin} = (\text{FLmax}/11)^{-1} = \frac{20k}{21k-2}\ \text{Brate}$$

The permissible baud rate error between UART6 and the transmission destination can be calculated from the above minimum and maximum baud rate expressions, as follows.

**Table 11-5. Maximum/Minimum Permissible Baud Rate Error**

| Division Ratio (k) | Maximum Permissible Baud Rate Error | Minimum Permissible 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% |

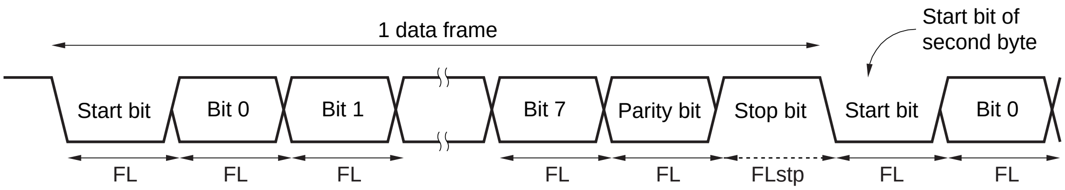
**Remarks 1.** The permissible error of reception depends on the number of bits in one frame, input clock frequency, and division ratio (k). The higher the input clock frequency and the higher the division ratio (k), the higher the permissible error.

**2.** k: Set value of BRGC6

### (5) Data frame length during continuous transmission

When data is continuously transmitted, the data frame length from a stop bit to the next start bit is extended by two clocks of base clock from the normal value. However, the result of communication is not affected because the timing is initialized on the reception side when the start bit is detected.

**Figure 11-26. Data Frame Length During Continuous Transmission**



Where the 1-bit data length is FL, the stop bit length is FLstp, and base clock frequency is $f_{XCLK6}$, the following expression is satisfied.

$$FLstp = FL + 2/f_{XCLK6}$$

Therefore, the data frame length during continuous transmission is:

$$\text{Data frame length} = 11 \times FL + 2/f_{XCLK6}$$

# CHAPTER 12 MULTIPLIER

## 12.1 Multiplier Function

The multiplier has the following function.

- Calculation of 8 bits × 8 bits = 16 bits

## 12.2 Multiplier Configuration

**(1) 16-bit multiplication result storage register 0 (MUL0)**

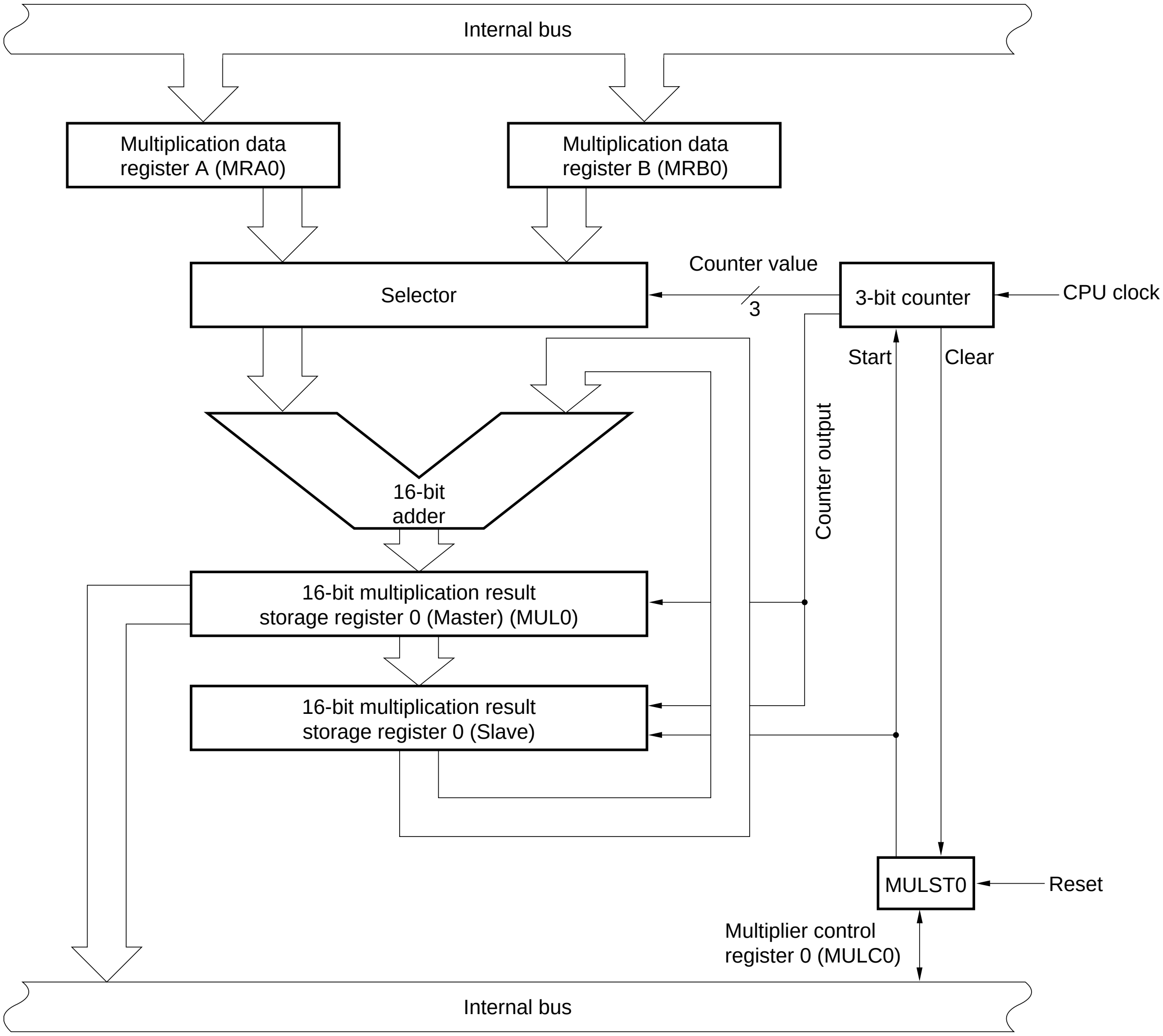
This register stores the 16-bit result of multiplication.
This register holds the result of multiplication after 16 CPU clocks have elapsed.
MUL0 can be read by a 16-bit memory manipulation instruction.
Reset signal generation makes MUL0 undefined.

**Caution Although this register is manipulated with a 16-bit memory manipulation instruction, it can be also manipulated with an 8-bit memory manipulation instruction. When using an 8-bit memory manipulation instruction, however, access the register by means of direct addressing.**

**(2) Multiplication data registers A and B (MRA0 and MRB0)**

These are 8-bit multiplication data storage registers. The multiplier multiplies the values of MRA0 and MRB0.
MRA0 and MRB0 can be written by an 8-bit memory manipulation instruction.
Reset signal generation makes these registers undefined.
Figure 12-1 shows the block diagram of the multiplier.

**Figure 12-1. Block Diagram of Multiplier**


Internal bus
Multiplication data register A (MRA0)
Multiplication data register B (MRB0)
Counter value
Selector
3
3-bit counter
CPU clock
Start
Clear
Counter output
16-bit adder
16-bit multiplication result storage register 0 (Master) (MUL0)
16-bit multiplication result storage register 0 (Slave)
MULST0
Reset
Multiplier control register 0 (MULC0)
Internal bus

## 12.3 Multiplier Control Register

The multiplier is controlled by the following register.

- Multiplier control register 0 (MULC0)

### (1) Multiplier control register 0 (MULC0)

This register indicates the operating status of the multiplier after operation, as well as controls the multiplier.
MULC0 can be set by a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation clears this register to 00H.

**Figure 12-2. Format of Multiplier Control Register 0 (MULC0)**

Address: FFD2H After reset: 00H RW

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| MULC0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MULST0 |

| MULST0 | Multiplier operation start control bit | Operating status of multiplier |
|---|---|---|
| 0 | Stops operation after resetting counter to 0. | Operation stops |
| 1 | Enables operation | Operation in progress |

**Caution Be sure to clear bits 1 to 7 to 0.**

## 12.4 Multiplier Operation

The multiplier of the 78K0S/KB1+ can execute the calculation of 8 bits × 8 bits = 16 bits. Figure 12-3 shows the operation timing of the multiplier where MRA0 is set to AAH and MRB0 is set to D3H.

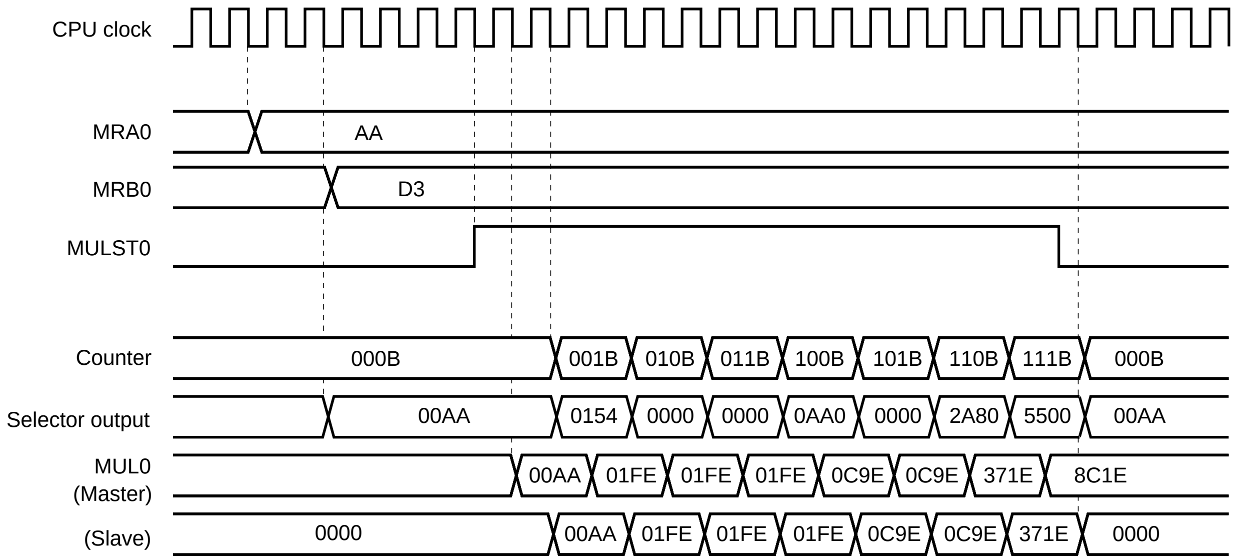
<1> Counting is started by setting MULST0.

<2> The data generated by the selector is added to the data of MUL0 at each CPU clock, and the counter value is incremented by one.

<3> If MULST0 is cleared when the counter value is 111B, the operation is stopped. At this time, MUL0 holds the data.

<4> While MULST0 is low, the counter and slave are cleared.

**Figure 12-3. Multiplier Operation Timing (Example of AAH × D3H)**



# CHAPTER 13 INTERRUPT FUNCTIONS

## 13.1 Interrupt Function Types

All interrupts are controlled as maskable interrupts.

- **Maskable interrupts**
  These interrupts undergo mask control. If two or more interrupt requests are simultaneously generated, each interrupt has a predetermined priority as shown in Table 13-1.
  A standby release signal is generated.
  There are nine internal sources and four external sources of maskable interrupts.

## 13.2 Interrupt Sources and Configuration

There are a total of 13 interrupt sources, and up to four reset sources (see **Table 13-1**).

**Table 13-1. Interrupt Sources**

| Interrupt Type | Priority[Note 1] | Interrupt Source | | Internal/ External | Vector Table Address | Basic Configuration Type[Note 2] |
|---|---|---|---|---|---|---|
| | | Name | Trigger | | | |
| Maskable | 1 | INTLVI | Low-voltage detection[Note 3] | Internal | 0006H | (A) |
| | 2 | INTP0 | Pin input edge detection | External | 0008H | (B) |
| | 3 | INTP1 | | | 000AH | |
| | 4 | INTTMH1 | Match between TMH1 and CMP01 (when compare register is specified) | Internal | 000CH | (A) |
| | 5 | INTTM000 | Match between TM00 and CR000 (when compare register is specified), TI010 pin valid edge detection (when capture register is specified) | | 000EH | |
| | 6 | INTTM010 | Match between TM00 and CR010 (when compare register is specified), TI000 pin valid edge detection (when capture register is specified) | | 0010H | |
| | 7 | INTAD | End of A/D conversion | | 0012H | |
| | 8 | INTP2 | Pin input edge detection | External | 0016H | (B) |
| | 9 | INTP3 | | | 0018H | |
| | 10 | INTTM80 | Match between TM80 and CR80 | Internal | 001AH | (A) |
| | 11 | INTSRE6 | UART6 reception error occurrence | | 001CH | |
| | 12 | INTSR6 | End of UART6 reception | | 001EH | |
| | 13 | INTST6 | End of UART6 transmission | | 0020H | |
| Reset | – | RESET | Reset input | – | 0000H | – |
| | | POC | Power-on-clear | | | |
| | | LVI | Low-voltage detection[Note 4] | | | |
| | | WDT | WDT overflow | | | |

**Notes 1.** Priority is the priority order when several maskable interrupt requests are generated at the same time. 1 is the highest and 13 is the lowest.

**2.** Basic configuration types (A) and (B) correspond to (A) and (B) in Figure 13-1.

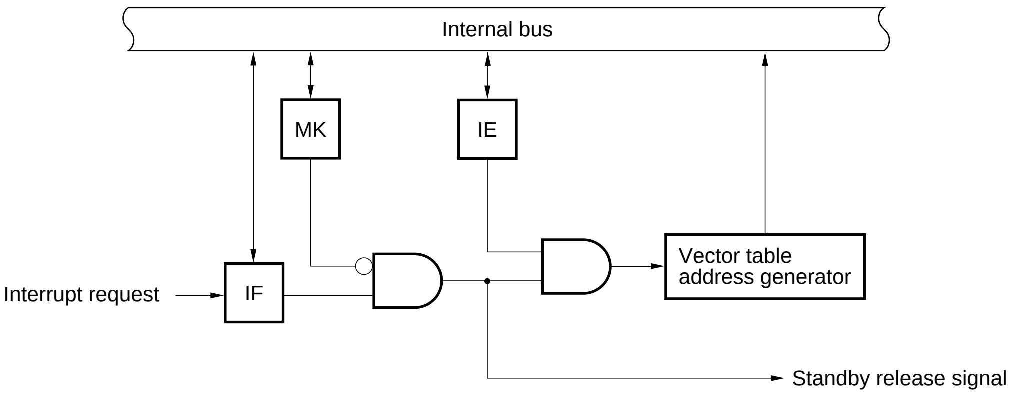
**3.** When bit 1 (LVIMD) of low-voltage detection register (LVIM) = 0 is selected.

**4.** When bit 1 (LVIMD) of low-voltage detection register (LVIM) = 1 is selected.

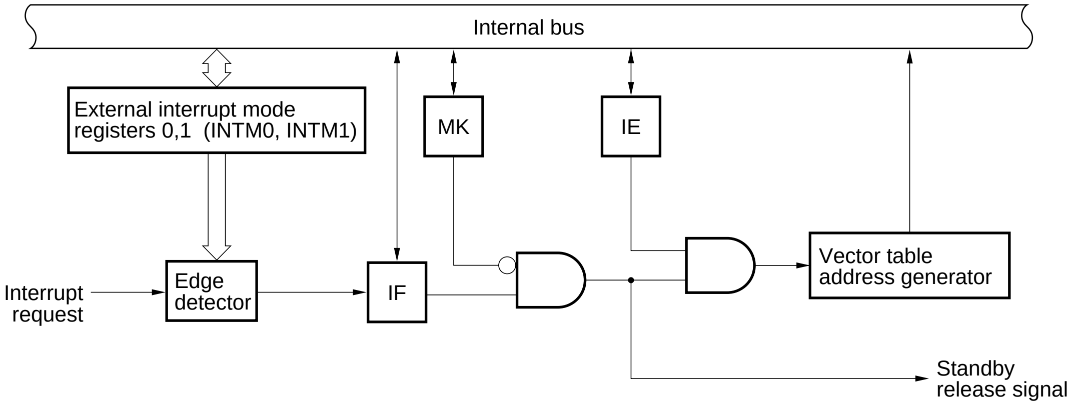
**Caution No interrupt sources correspond to the vector table address 0014H.**

### Figure 13-1. Basic Configuration of Interrupt Function

**(A) Internal maskable interrupt**



**(B) External maskable interrupt**


Internal bus
External interrupt mode registers 0,1 (INTM0, INTM1)
MK
IE
Interrupt request
Edge detector
IF
Vector table address generator
Standby release signal

IF: Interrupt request flag

IE: Interrupt enable flag

MK: Interrupt mask flag

## 13.3 Interrupt Function Control Registers

The interrupt functions are controlled by the following four types of registers.

- Interrupt request flag registers 0, 1 (IF0, IF1)
- Interrupt mask flag registers 0, 1 (MK0, MK1)
- External interrupt mode registers 0, 1 (INTM0, INTM1)
- Program status word (PSW)

Table 13-2 lists interrupt requests, the corresponding interrupt request flags, and interrupt mask flags.

**Table 13-2. Interrupt Request Signals and Corresponding Flags**

| Interrupt Request Signal | Interrupt Request Flag | Interrupt Mask Flag |
|---|---|---|
| INTLVI | LVIIF | LVIMK |
| INTP0 | PIF0 | PMK0 |
| INTP1 | PIF1 | PMK1 |
| INTTMH1 | TMIFH1 | TMMKH1 |
| INTTM000 | TMIF000 | TMMK000 |
| INTTM010 | TMIF010 | TMMK010 |
| INTAD | ADIF | ADMK |
| INTP2 | PIF2 | PMK2 |
| INTP3 | PIF3 | PMK3 |
| INTTM80 | TMIF80 | TMMK80 |
| INTSRE6 | SREIF6 | SREMK6 |
| INTSR6 | SRIF6 | SRMK6 |
| INTST6 | STIF6 | STMK6 |

**(2) Interrupt mask flag registers 0, 1 (MK0, MK1)**

The interrupt mask flag is used to enable and disable the corresponding maskable interrupts.

MK0 and MK1 are set with a 1-bit or 8-bit memory manipulation instruction.

Generation of reset signal sets MK0 and MK1 to FFH.

**Figure 13-3. Format of Interrupt Mask Flag Registers 0, 1 (MK0, MK1)**

Address: FFE4H After reset: FFH R/W

| Symbol | <7> | <6> | <5> | <4> | <3> | <2> | <1> | 0 |
|---|---|---|---|---|---|---|---|---|
| MK0 | ADMK | TMMK010 | TMMK000 | TMMKH1 | PMK1 | PMK0 | LVIMK | 1 |

Address: FFE5H After reset: FFH R/W

| Symbol | 7 | <6> | <5> | <4> | <3> | <2> | <1> | 0 |
|---|---|---|---|---|---|---|---|---|
| MK1 | 1 | STMK6 | SRMK6 | SREMK6 | TMMK80 | PMK3 | PMK2 | 1 |

| ××MK× | Interrupt servicing control |
|---|---|
| 0 | Enables interrupt servicing. |
| 1 | Disables interrupt servicing. |

**Caution Because P30, P31, P41, and P43 have an alternate function as external interrupt inputs, when the output level is changed by specifying the output mode of the port function, an interrupt request flag is set. Therefore, the interrupt mask flag should be set to 1 before using the output mode.**

### (3) External interrupt mode register 0 (INTM0)

This register is used to set the valid edge of INTP0 to INTP2.

INTM0 is set with an 8-bit memory manipulation instruction.

Reset signal generation clears INTM0 to 00H.

**Figure 13-4. Format of External Interrupt Mode Register 0 (INTM0)**

Address: FFECH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| INTM0 | ES21 | ES20 | ES11 | ES10 | ES01 | ES00 | 0 | 0 |

| ES21 | ES20 | INTP2 valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

| ES11 | ES10 | INTP1 valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

| ES01 | ES00 | INTP0 valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

**Cautions 1. Be sure to clear bits 0 and 1 to 0.**

**2. Before setting the INTM0 register, be sure to set the corresponding interrupt mask flag (××MK× = 1) to disable interrupts. After setting the INTM0 register, clear the interrupt request flag (××IF× = 0), then clear the interrupt mask flag (××MK× = 0), which will enable interrupts.**

### (4) External interrupt mode register 1 (INTM1)

INTM1 is used to specify the valid edge for INTP3.

INTM1 is set with an 8-bit memory manipulation instruction.

Reset signal generation clears INTM1 to 00H.

**Figure 13-5. Format of External Interrupt Mode Register 1 (INTM1)**

Address: FFEDH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| INTM1 | 0 | 0 | 0 | 0 | 0 | 0 | ES31 | ES30 |

| ES31 | ES30 | INTP3 valid edge selection |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

**Cautions 1. Be sure to clear bits 2 to 7 to 0.**

**2. Before setting INTM1, set PMK3 to 1 to disable interrupts. To enable interrupts, clear PIF3 to 0, then clear PMK3 to 0.**

### (5) Program status word (PSW)

The program status word is used to hold the instruction execution result and the current status of the interrupt requests. The IE flag, used to enable and disable maskable interrupts, is mapped to PSW.

PSW can be read- and write-accessed in 8-bit units, as well as using bit manipulation instructions and dedicated instructions (EI and DI). When a vectored interrupt is acknowledged, the PSW is automatically saved to a stack, and the IE flag is reset to 0.

Generation of reset signal sets PSW to 02H.

**Figure 13-6. Program Status Word (PSW) Configuration**

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | After reset |
|---|---|---|---|---|---|---|---|---|---|
| PSW | IE | Z | 0 | AC | 0 | 0 | 1 | CY | 02H |

Z, AC, CY → Used in the execution of ordinary instructions

| IE | Whether to enable/disable interrupt acknowledgment |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

## 13.4 Interrupt Servicing Operation

### 13.4.1 Maskable interrupt request acknowledgment operation

A maskable interrupt request can be acknowledged when the interrupt request flag is set to 1 and the corresponding interrupt mask flag is cleared to 0. A vectored interrupt request is acknowledged in the interrupt enabled status (when the IE flag is set to 1).

The time required to start the interrupt servicing after a maskable interrupt request has been generated is shown in Table 13-3.

See Figures 13-8 and 13-9 for the interrupt request acknowledgment timing.

**Table 13-3. Time from Generation of Maskable Interrupt Request to Servicing**

| Minimum Time | Maximum Time[Note] |
|---|---|
| 9 clocks | 19 clocks |

**Note** The wait time is maximum when an interrupt request is generated immediately before BT and BF instructions.

**Remark** 1 clock: $\frac{1}{f_{CPU}}$ ($f_{CPU}$: CPU clock)

When two or more maskable interrupt requests are generated at the same time, they are acknowledged starting from the interrupt request assigned the highest priority.

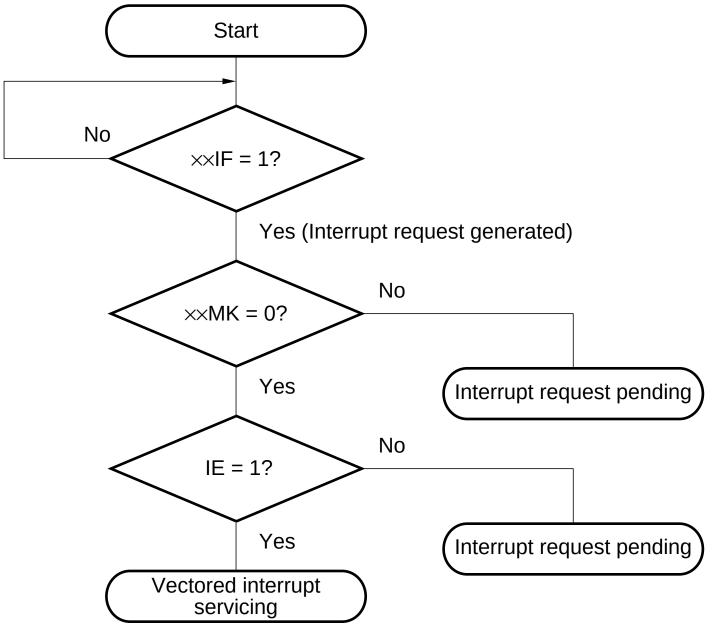
A pending interrupt is acknowledged when a status in which it can be acknowledged is set.

Figure 13-7 shows the algorithm of interrupt request acknowledgment.

When a maskable interrupt request is acknowledged, the contents of the PSW and PC are saved to the stack in that order, the IE flag is reset to 0, and the data in the vector table determined for each interrupt request is loaded to the PC, and execution branches.

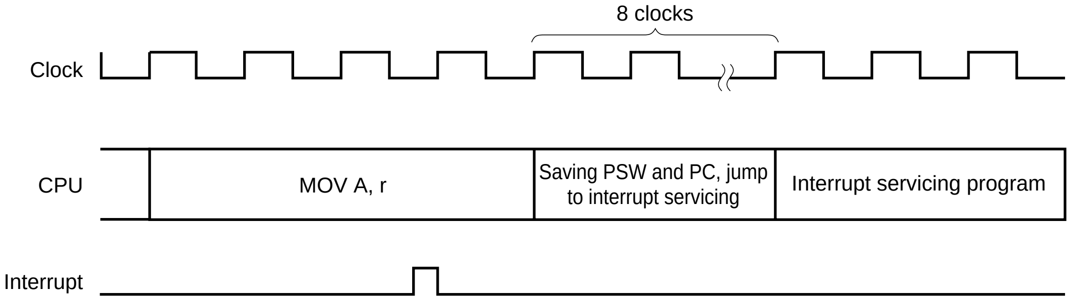
To return from interrupt servicing, use the RETI instruction.

**Figure 13-7. Interrupt Request Acknowledgment Processing Algorithm**


Start
No
××IF = 1?
Yes (Interrupt request generated)
××MK = 0?
No
Yes
Interrupt request pending
IE = 1?
No
Yes
Interrupt request pending
Vectored interrupt servicing

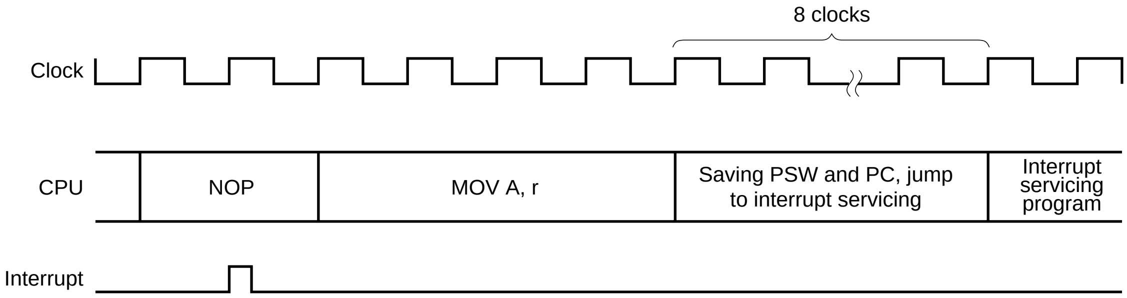
| | |
|---|---|
| ××IF: | Interrupt request flag |
| ××MK: | Interrupt mask flag |
| IE: | Flag to control maskable interrupt request acknowledgment (1 = enable, 0 = disable) |

**Figure 13-8. Interrupt Request Acknowledgment Timing (Example of MOV A, r)**


8 clocks
Clock
CPU
MOV A, r
Saving PSW and PC, jump to interrupt servicing
Interrupt servicing program
Interrupt

If an interrupt request flag (××IF) is set before an instruction clock n (n = 4 to 10) under execution becomes n – 1, the interrupt is acknowledged after the instruction under execution is complete. Figure 13-8 shows an example of the interrupt request acknowledgment timing for an 8-bit data transfer instruction MOV A, r. Since this instruction is executed for 4 clocks, if an interrupt occurs for 3 clocks after the execution starts, the interrupt acknowledgment processing is performed after the MOV A, r instruction is executed.

**Figure 13-9. Interrupt Request Acknowledgment Timing (When Interrupt Request Flag Is Set at Last Clock During Instruction Execution)**


8 clocks
Clock
CPU
NOP
MOV A, r
Saving PSW and PC, jump to interrupt servicing
Interrupt servicing program
Interrupt

If an interrupt request flag (××IF) is set at the last clock of the instruction, the interrupt acknowledgment processing starts after the next instruction is executed.

Figure 13-9 shows an example of the interrupt request acknowledgment timing for an interrupt request flag that is set at the second clock of NOP (2-clock instruction). In this case, the MOV A, r instruction after the NOP instruction is executed, and then the interrupt acknowledgment processing is performed.

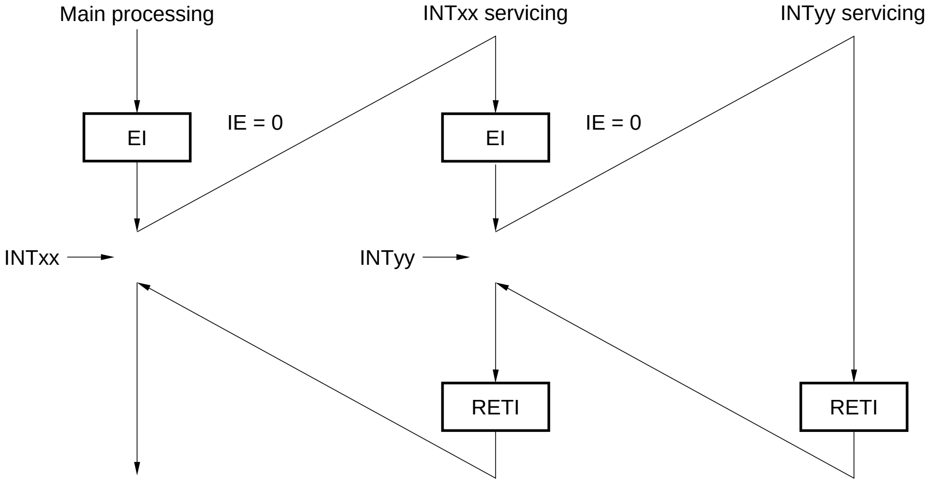
**Caution Interrupt requests will be held pending while the interrupt request flag registers 0, 1 (IF0, IF1) or interrupt mask flag registers 0, 1 (MK0, MK1) are being accessed.**

### 13.4.2 Multiple interrupt servicing

Multiple interrupt servicing in which another interrupt is acknowledged while an interrupt is being serviced can be performed using a priority order system. When two or more interrupts are generated at once, interrupt servicing is performed according to the priority assigned to each interrupt request in advance (see **Table 13-1**).

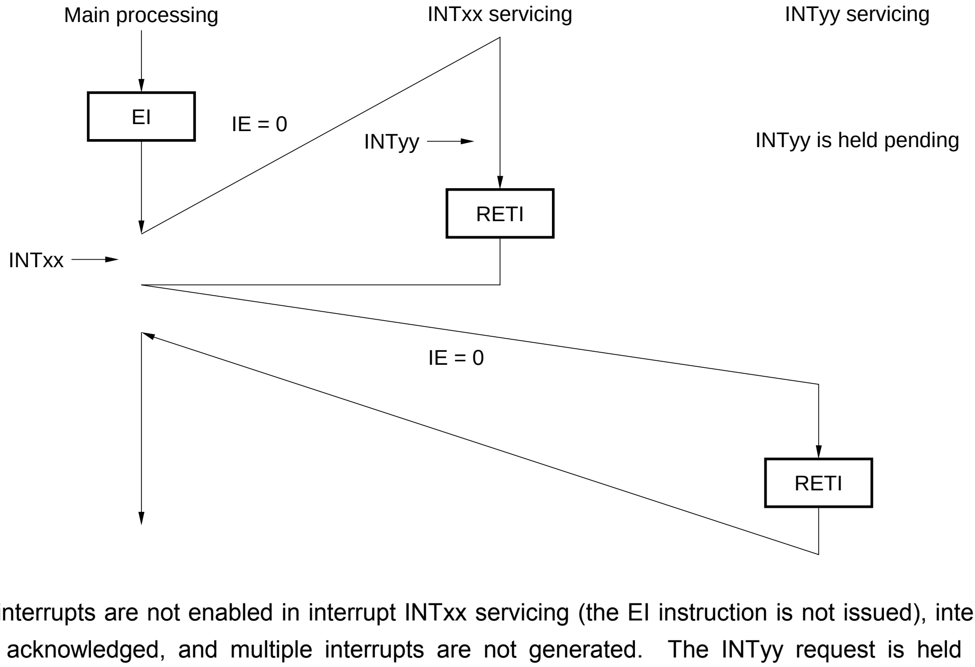
**Figure 13-10. Example of Multiple Interrupts**

**Example 1. Multiple interrupts are acknowledged**



During interrupt INTxx servicing, interrupt request INTyy is acknowledged, and multiple interrupts are generated. The EI instruction is issued before each interrupt request acknowledgment, and the interrupt request acknowledgment enable state is set.

**Example 2. Multiple interrupts are not generated because interrupts are not enabled**



Because interrupts are not enabled in interrupt INTxx servicing (the EI instruction is not issued), interrupt request INTyy is not acknowledged, and multiple interrupts are not generated. The INTyy request is held pending and acknowledged after the INTxx servicing is performed.

IE = 0: Interrupt request acknowledgment disabled

### 13.4.3 Interrupt request pending

Some instructions may keep pending the acknowledgment of an instruction request until the completion of the execution of the next instruction even if the interrupt request (maskable interrupt and external interrupt) is generated during the execution. The following shows such instructions (interrupt request pending instruction).

- Manipulation instruction for interrupt request flag registers 0, 1 (IF0, IF1)
- Manipulation instruction for interrupt mask flag registers 0, 1 (MK0, MK1)

# CHAPTER 14 STANDBY FUNCTION

## 14.1 Standby Function and Configuration

### 14.1.1 Standby function

**Table 14-1. Relationship Between Operation Clocks in Each Operation Status**

| Status / Operation Mode | Low-Speed Ring-OSC Oscillator | | | System Clock | Clock Supplied to Peripheral Hardware |
|---|---|---|---|---|---|
| | **Note 1** | **Note 2** | | | |
| | | LSRSTOP = 0 | LSRSTOP = 1 | | |
| Reset | Stopped | | | Stopped | Stopped |
| STOP | Oscillating | Oscillating[Note 3] | Stopped | | |
| HALT | | | | Oscillating | Oscillating |

**Notes 1.** When “Cannot be stopped” is selected for low-speed Ring-OSC by the option byte.

**2.** When it is selected that the low-speed Ring-OSC oscillator “can be stopped by software”, oscillation of the low-speed Ring-OSC oscillator can be stopped by LSRSTOP.

**3.** If the operating clock of the watchdog timer is the low-speed Ring-OSC clock, the watchdog timer is stopped.

**Caution The LSRSTOP setting is valid only when “Can be stopped by software” is set for the low-speed Ring-OSC oscillator by the option byte.**

**Remark** LSRSTOP: Bit 0 of the low-speed Ring-OSC mode register (LSRCM)

The standby function is designed to reduce the operating current of the system. The following two modes are available.

**(1) HALT mode**

HALT instruction execution sets the HALT mode. In the HALT mode, the CPU operation clock is stopped. Oscillation of the system clock oscillator continues. If the low-speed Ring-OSC oscillator is operating before the HALT mode is set, oscillation of the clock of the low-speed Ring-OSC oscillator continues (refer to **Table 14-1**. Oscillation of the low-speed Ring-OSC clock (whether it cannot be stopped or can be stopped by software) is set by the option byte). In this mode, the operating current is not decreased as much as in the STOP mode, but the HALT mode is effective for restarting operation immediately upon interrupt request generation and carrying out intermittent operations.

**(2) STOP mode**

STOP instruction execution sets the STOP mode. In the STOP mode, the system clock oscillator stops, stopping the whole system, thereby considerably reducing the CPU operating current.

Because this mode can be cleared by an interrupt request, it enables intermittent operations to be carried out. However, select the HALT mode if processing must be immediately started by an interrupt request when the operation stop time[Note] is generated after the STOP mode is released (because an additional wait time for stabilizing oscillation elapses when crystal/ceramic oscillation is used).

**Note** The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

In either of these two modes, all the contents of registers, flags and data memory just before the standby mode is set are held. The I/O port output latches and output buffer statuses are also held.

**Cautions 1. When shifting to the STOP mode, be sure to stop the peripheral hardware operation before executing STOP instruction (except the peripheral hardware that operates on the low-speed Ring-OSC clock).**

**2. The following sequence is recommended for operating current reduction of the A/D converter when the standby function is used: First clear bit 7 (ADCS) and bit 0 (ADCE) of the A/D converter mode register (ADM) to 0 to stop the A/D conversion operation, and then execute the HALT or STOP instruction.**

**3. If the low-speed Ring-OSC oscillator is operating before the STOP mode is set, oscillation of the low-speed Ring-OSC clock cannot be stopped in the STOP mode (refer to Table 14-1).**

### 14.1.2 Registers used during standby

## 14.2 Standby Function Operation

### 14.2.1 HALT mode

**(1) HALT mode**

The HALT mode is set by executing the HALT instruction.

The operating statuses in the HALT mode are shown below.

**Caution Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set.**

**Table 14-2. Operating Statuses in HALT Mode**

| Setting of HALT Mode<br>Item | | Low-Speed Ring-OSC Cannot Be Stopped[Note] | Low-Speed Ring-OSC Can Be Stopped[Note] | |
|---|---|---|---|---|
| | | | When Low-Speed Ring-OSC Oscillation Continues | When Low-Speed Ring-OSC Oscillation Stops |
| System clock | | Clock supply to CPU is stopped. | | |
| CPU | | Operation stops. | | |
| Port (latch) | | Holds status before HALT mode was set. | | |
| 16-bit timer/event counter 00 | | Operable | | |
| 8-bit timer 80 | | Operable | | |
| 8-bit timer H1 | Sets count clock to $f_{XP}$ to $f_{XP}/2^{12}$ | Operable | | |
| | Sets count clock to $f_{RL}/2^{7}$ | Operable | Operable | Operation stops. |
| Watchdog timer | System clock selected as operating clock | Setting prohibited | Operation stops. | |
| | “Low-speed Ring-OSC clock” selected as operating clock | Operable (Operation continues.) | Operation stops. | |
| A/D converter | | Operable | | |
| Serial interface UART6 | | Operable | | |
| Power-on-clear circuit | | Always operates. | | |
| Low-voltage detector | | Operable | | |
| External interrupt | | Operable | | |

**Note** “Cannot be stopped” or “Stopped by software” is selected for low-speed Ring-OSC by the option byte (for the option byte, see **CHAPTER 18 OPTION BYTE**).

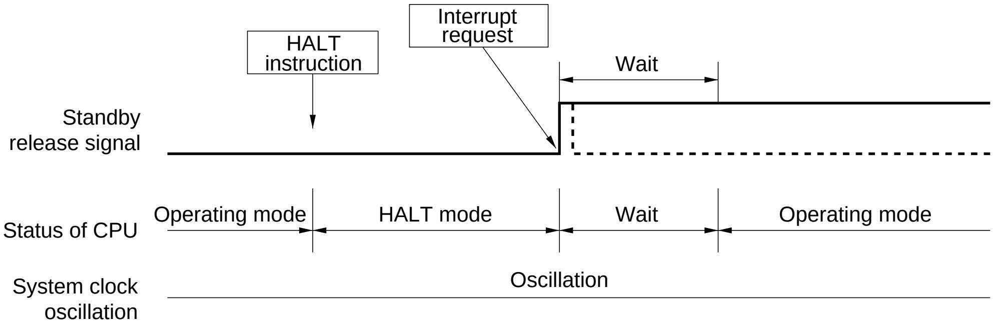
**(2) HALT mode release**

The HALT mode can be released by the following two sources.

**(a) Release by unmasked interrupt request**

When an unmasked interrupt request is generated, the HALT mode is released. If interrupt acknowledgement is enabled, vectored interrupt servicing is carried out. If interrupt acknowledgement is disabled, the next address instruction is executed.

**Figure 14-2. HALT Mode Release by Interrupt Request Generation**



**Remarks 1.** The broken lines indicate the case when the interrupt request which has released the standby mode is acknowledged.

**2.** The wait time is as follows:

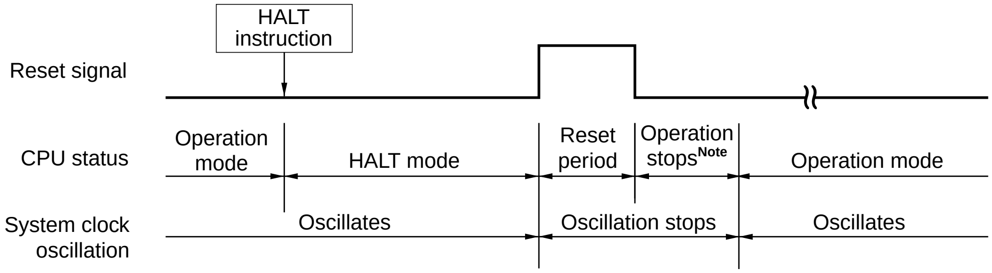
- When vectored interrupt servicing is carried out: 11 to 13 clocks
- When vectored interrupt servicing is not carried out: 3 to 5 clocks

**(b) Release by reset signal generation**

When the reset signal is generated, HALT mode is released, and then, as in the case with a normal reset operation, the program is executed after branching to the reset vector address.

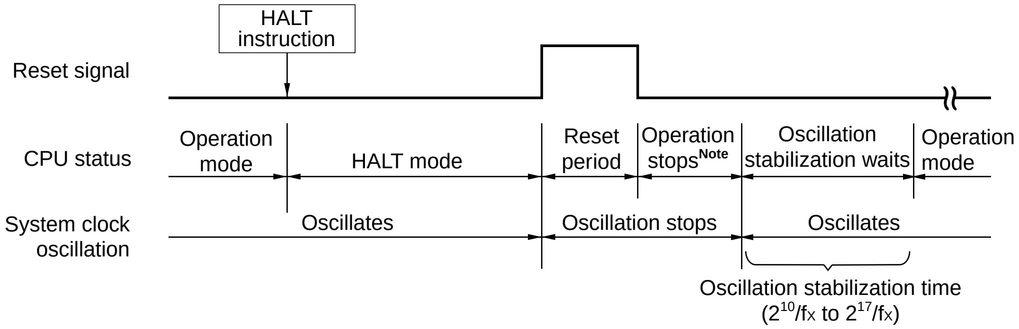
**Figure 14-3. HALT Mode Release by Reset Signal Generation**

**(1) When CPU clock is high-speed Ring-OSC clock or external input clock**



**Note** Operation is stopped (277 μs (MIN.), 544 μs (TYP.), 1.075 ms (MAX.)) because the option byte is referenced.

**(2) When CPU clock is crystal/ceramic oscillation clock**



**Note** Operation is stopped (276 μs (MIN.), 544 μs (TYP.), 1.074 ms (MAX.)) because the option byte is referenced.

**Remark** fx: System clock oscillation frequency

**Table 14-3. Operation in Response to Interrupt Request in HALT Mode**

| Release Source | MK×× | IE | Operation |
|---|---|---|---|
| Maskable interrupt request | 0 | 0 | Next address instruction execution |
| | 0 | 1 | Interrupt servicing execution |
| | 1 | × | HALT mode held |
| Reset signal generation | – | × | Reset processing |

×: don’t care

### 14.2.2 STOP mode

**(1) STOP mode setting and operating statuses**

The STOP mode is set by executing the STOP instruction.

**Caution Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. Thus, in the STOP mode, the normal operation mode is restored after the STOP instruction is executed and then the operation is stopped for 34 μs (TYP.) (after an additional wait time for stabilizing the oscillation set by the oscillation stabilization time select register (OSTS) has elapsed when crystal/ceramic oscillation is used).**

The operating statuses in the STOP mode are shown below.

**Table 14-4. Operating Statuses in STOP Mode**

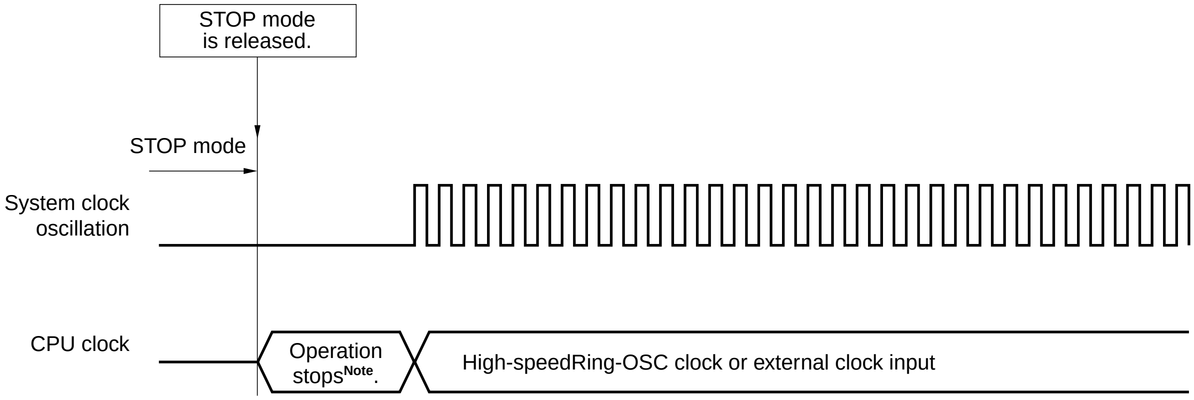
| Setting of HALT Mode<br>Item | | Low-Speed Ring-OSC Cannot Be Stopped[Note] | Low-Speed Ring-OSC Can Be Stopped[Note] | |
|---|---|---|---|---|
| | | | When Low-Speed Ring-OSC Oscillation Continues | When Low-Speed Ring-OSC Oscillation Stops |
| System clock | | Oscillation stops. | | |
| CPU | | Operation stops. | | |
| Port (latch) | | Holds status before STOP mode is set. | | |
| 16-bit timer/event counter 00 | | Operation stops. | | |
| 8-bit timer 80 | | Operation stops. | | |
| 8-bit timer H1 | Sets count clock to $f_{XP}$ to $f_{XP}/2^{12}$ | Operation stops. | | |
| | Sets count clock to $f_{RL}/2^{7}$ | Operable | Operable | Operation stops. |
| Watchdog timer | “Clock to peripheral hardware” selected as operating clock | Setting prohibited | Operation stops. | |
| | “Low-speed Ring-OSC clock” selected as operating clock | Operable (Operation continues.) | Operation stops. | |
| A/D converter | | Operation stops. | | |
| Serial interface UART6 | | Operation stops. | | |
| Power-on-clear circuit | | Always operates. | | |
| Low-voltage detector | | Operable | | |
| External interrupt | | Operable | | |

**Note** “Cannot be stopped” or “Stopped by software” is selected for low-speed Ring-OSC by the option byte (for the option byte, see **CHAPTER 18 OPTION BYTE**).

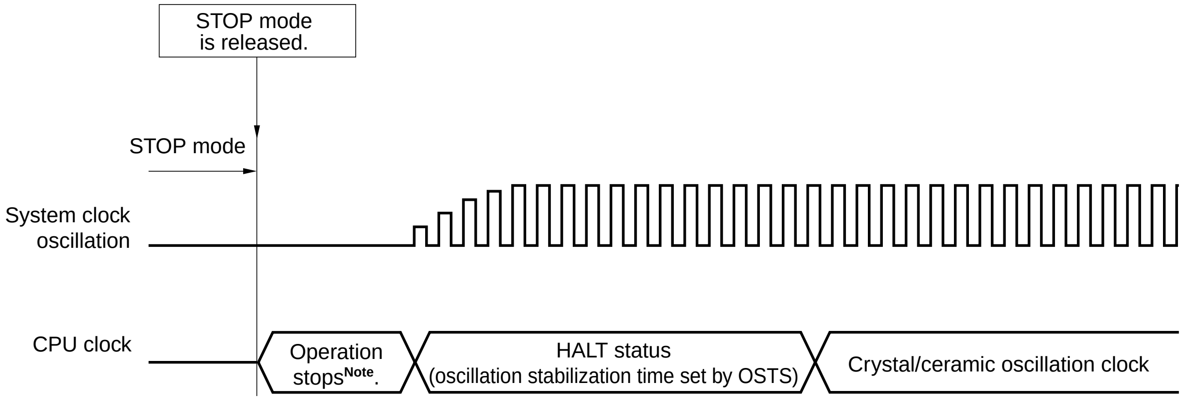
### (2) STOP mode release

**Figure 14-4. Operation Timing When STOP Mode Is Released**

**<1> If high-speed Ring-OSC clock or external input clock is selected as system clock to be supplied**



**<2> If crystal/ceramic oscillation clock is selected as system clock to be supplied**



**Note** The operation stop time is 17 $\mu$s (MIN.), 34 $\mu$s (TYP.), and 67 $\mu$s (MAX.).

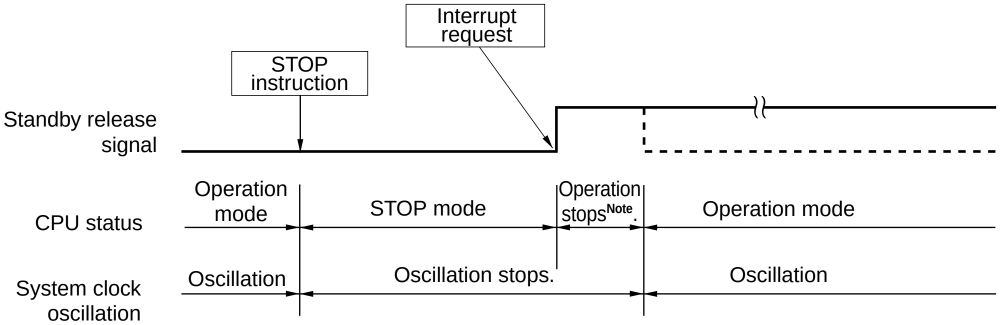
The STOP mode can be released by the following two sources.

**(a) Release by unmasked interrupt request**

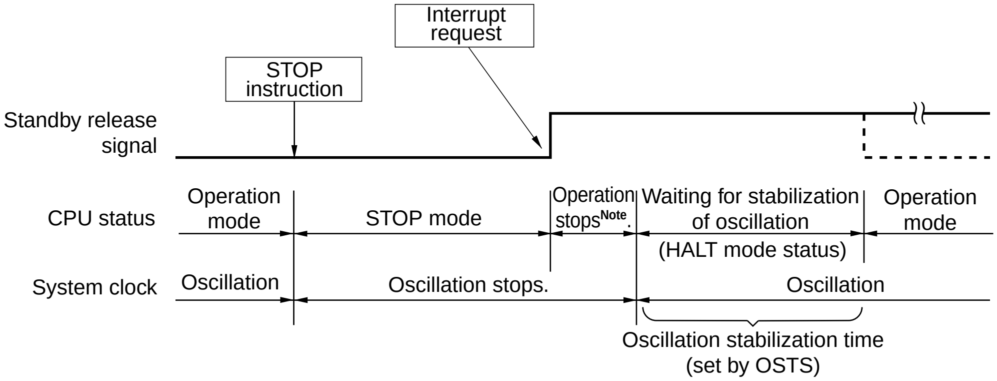
When an unmasked interrupt request is generated, the STOP mode is released. After the oscillation stabilization time has elapsed, if interrupt acknowledgment is enabled, vectored interrupt servicing is carried out. If interrupt acknowledgment is disabled, the next address instruction is executed.

**Figure 14-5. STOP Mode Release by Interrupt Request Generation**

**(1) If CPU clock is high-speed Ring-OSC clock or external input clock**



**(2) If CPU clock is crystal/ceramic oscillation clock**



**Note** The operation stop time is 17 $\mu$s (MIN.), 34 $\mu$s (TYP.), and 67 $\mu$s (MAX.).

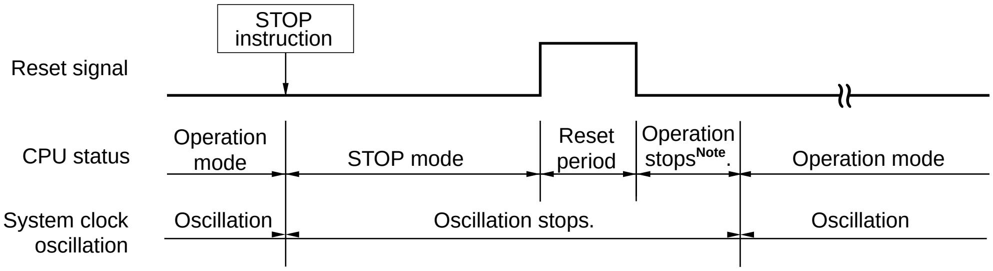
**Remark** The broken lines indicate the case when the interrupt request that has released the standby mode is acknowledged.

**(b) Release by reset signal generation**

When the reset signal is generated, STOP mode is released and a reset operation is performed after the oscillation stabilization time has elapsed.

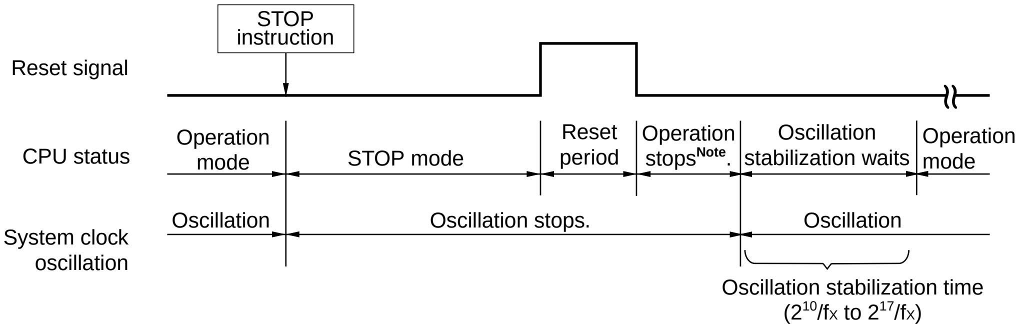
**Figure 14-6. STOP Mode Release by Reset Signal Generation**

**(1) If CPU clock is high-speed Ring-OSC clock or external input clock**



**Note** Operation is stopped (277 $\mu$s (MIN.), 544 $\mu$s (TYP.), 1.075 ms (MAX.)) because the option byte is referenced.

**(2) If CPU clock is crystal/ceramic oscillation clock**



**Note** Operation is stopped (276 $\mu$s (MIN.), 544 $\mu$s (TYP.), 1.074 ms (MAX.)) because the option byte is referenced.

**Remark** $f_X$: System clock oscillation frequency

**Table 14-5. Operation in Response to Interrupt Request in STOP Mode**

| Release Source | MK×× | IE | Operation |
|---|---|---|---|
| Maskable interrupt request | 0 | 0 | Next address instruction execution |
| | 0 | 1 | Interrupt servicing execution |

# CHAPTER 15 RESET FUNCTION

The following four operations are available to generate a reset signal.

(1) External reset input via $\overline{\text{RESET}}$ pin
(2) Internal reset by watchdog timer program loop detection
(3) Internal reset by comparison of supply voltage and detection voltage of power-on-clear (POC) circuit
(4) Internal reset by comparison of supply voltage and detection voltage of low-power-supply detector (LVI)

External and internal resets have no functional differences. In both cases, program execution starts at the address at 0000H and 0001H when the reset signal is input.

A reset is applied when a low level is input to the $\overline{\text{RESET}}$ pin, the watchdog timer overflows, or by POC and LVI circuit voltage detection, and each item of hardware is set to the status shown in Table 15-1. Each pin is high impedance during reset signal generation or during the oscillation stabilization time just after reset release, except for P130, which is low-level output.

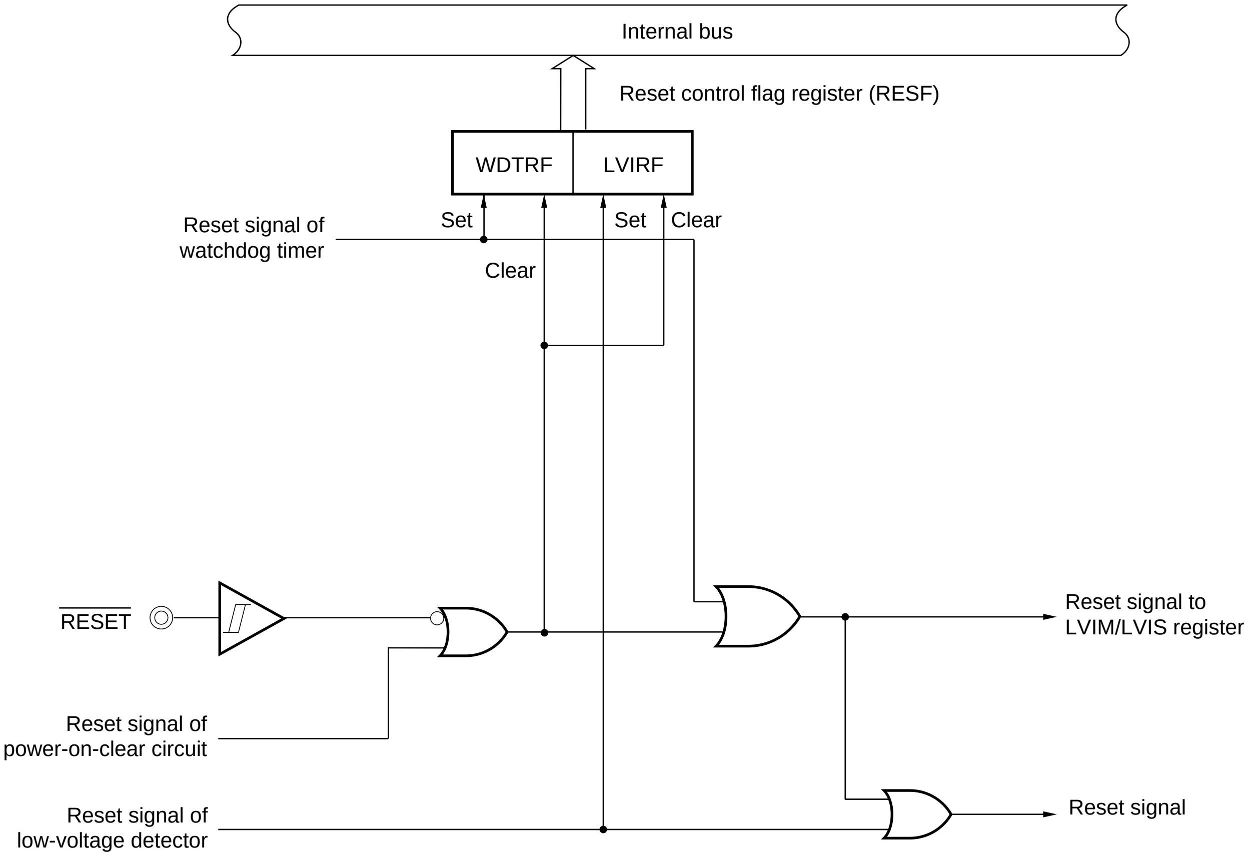
When a high level is input to the $\overline{\text{RESET}}$ pin, the reset is released and program execution starts using the CPU clock after referencing the option byte (after the option byte is referenced and the clock oscillation stabilization time elapses if crystal/ceramic oscillation is selected). A reset generated by the watchdog timer source is automatically released after the reset, and program execution starts using the CPU clock after referencing the option byte (after the option byte is referenced and the clock oscillation stabilization time elapses if crystal/ceramic oscillation is selected). (see **Figures 15-2** to **15-4**). Reset by POC and LVI circuit power supply detection is automatically released when $V_{DD} > V_{POC}$ or $V_{DD} > V_{LVI}$ after the reset, and program execution starts using the CPU clock after referencing the option byte (after the option byte is referenced and the clock oscillation stabilization time elapses if crystal/ceramic oscillation is selected) (see **CHAPTER 16 POWER-ON-CLEAR CIRCUIT** and **CHAPTER 17 LOW-VOLTAGE DETECTOR**).

**Cautions 1. For an external reset, input a low level for 2 μs or more to the $\overline{\text{RESET}}$ pin.**

**2. During reset signal generation, the system clock and low-speed Ring-OSC clock stop oscillating.**

**3. When the $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34), the 78K0S/KB1+ is reset if a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the POC circuit and before the option byte is referenced again. The reset status is retained until a high level is input to the $\overline{\text{RESET}}$ pin.**

**Figure 15-1. Block Diagram of Reset Function**



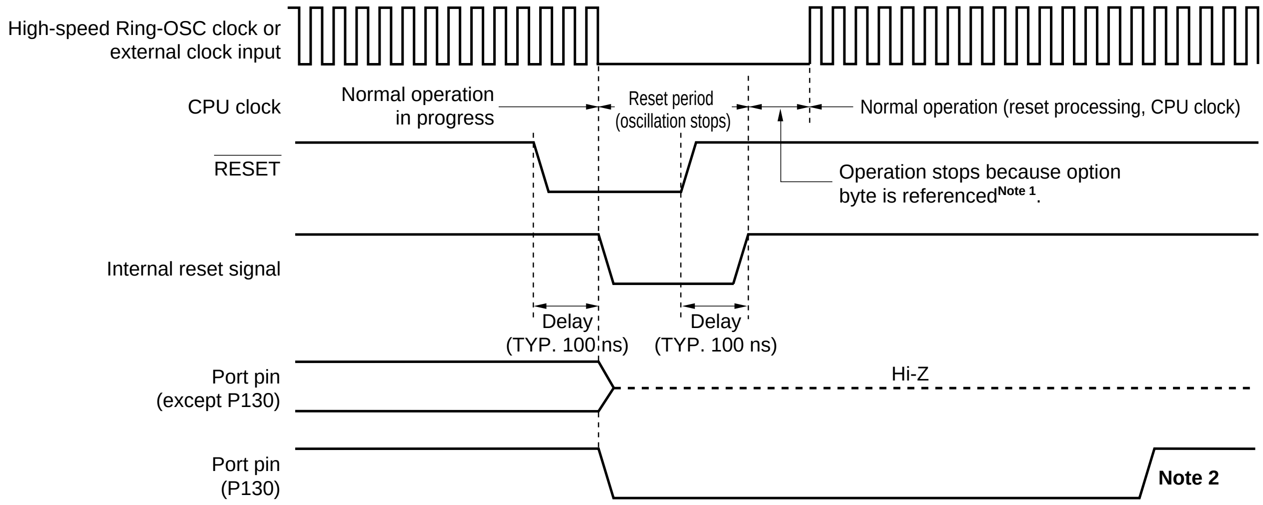
**Caution The LVI circuit is not reset by the internal reset signal of the LVI circuit.**

**Remarks 1.** LVIM: Low-voltage detect register

**2.** LVIS: Low-voltage detection level select register

### Figure 15-2. Timing of Reset by $\overline{\text{RESET}}$ Input

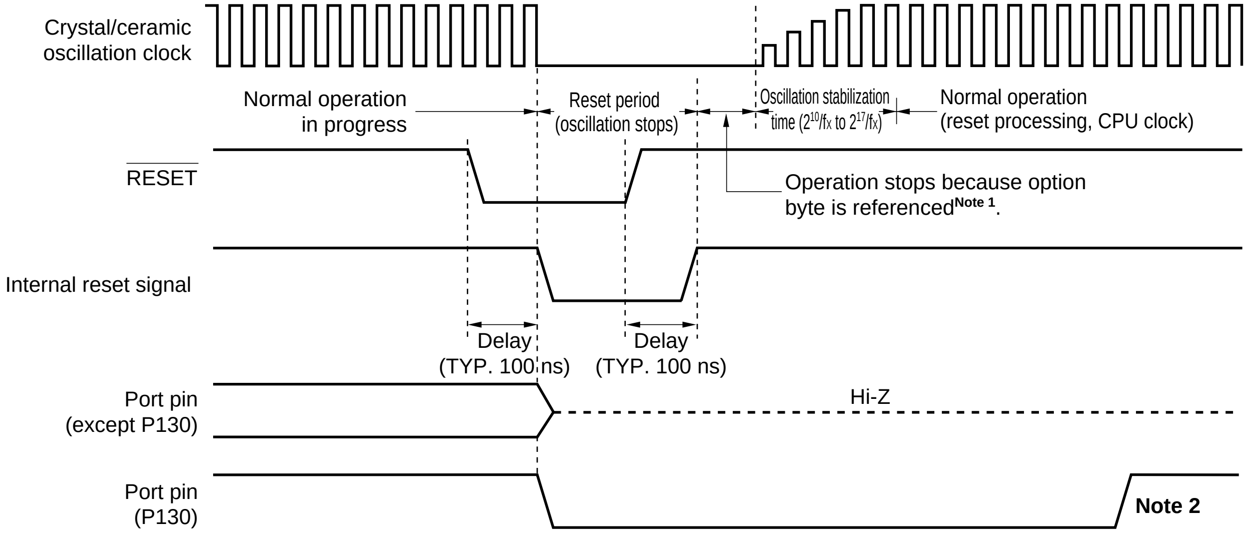
#### <1> With high-speed Ring-OSC clock or external clock input


High-speed Ring-OSC clock or external clock input
CPU clock
Normal operation in progress
Reset period (oscillation stops)
Normal operation (reset processing, CPU clock)
RESET
Operation stops because option byte is referenced Note 1.
Internal reset signal
Delay (TYP. 100 ns)
Delay (TYP. 100 ns)
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

**Notes 1.** The operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

#### <2> With crystal/ceramic oscillation clock


Crystal/ceramic oscillation clock
Normal operation in progress
Reset period (oscillation stops)
Oscillation stabilization time ($2^{10}$/fx to $2^{17}$/fx)
Normal operation (reset processing, CPU clock)
RESET
Operation stops because option byte is referenced Note 1.
Internal reset signal
Delay (TYP. 100 ns)
Delay (TYP. 100 ns)
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

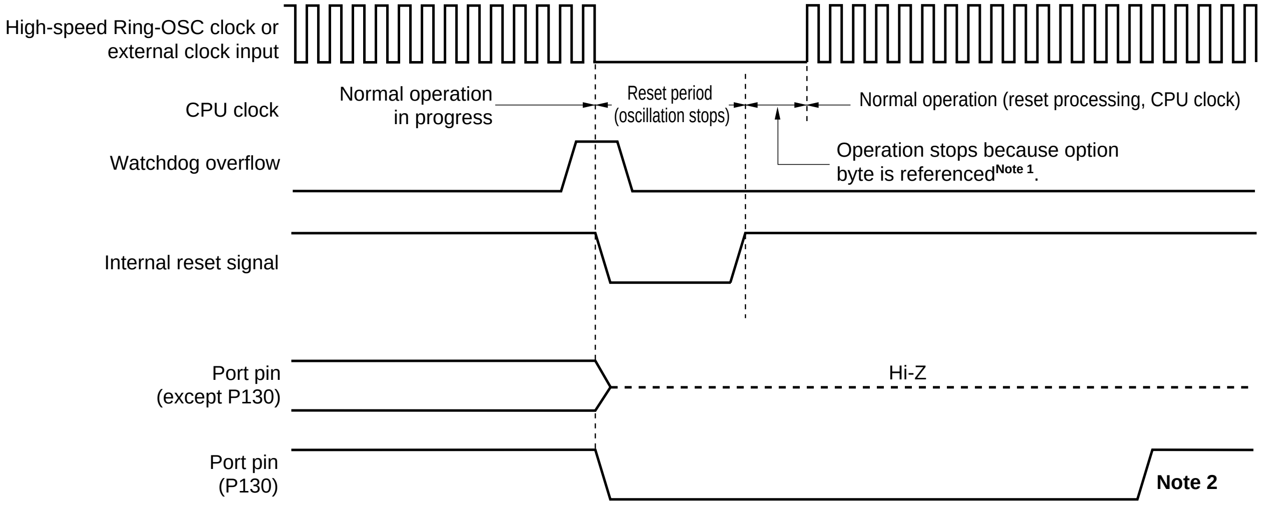
**Notes 1.** The operation stop time is 276 μs (MIN.), 544 μs (TYP.), and 1.074 ms (MAX.).

**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

**Remark** fx: System clock oscillation frequency

### Figure 15-3. Timing of Reset by Overflow of Watchdog Timer

#### <1> With high-speed Ring-OSC clock or external clock input

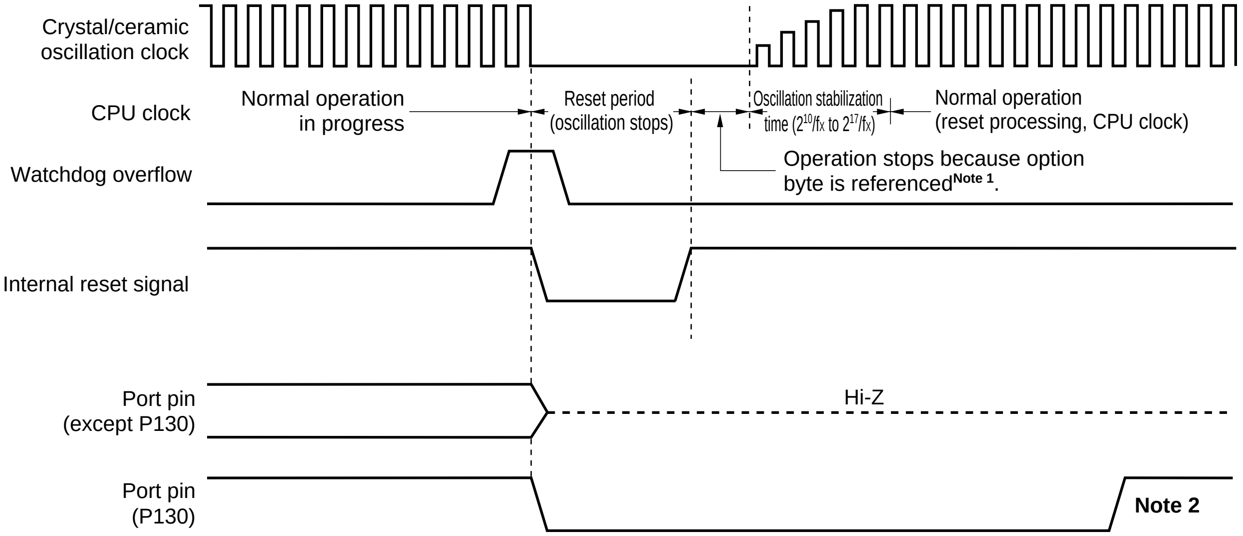

High-speed Ring-OSC clock or external clock input
CPU clock
Normal operation in progress
Reset period (oscillation stops)
Normal operation (reset processing, CPU clock)
Operation stops because option byte is referenced[Note 1].
Watchdog overflow
Internal reset signal
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

**Notes 1.** The operation stop time is 277 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.075 ms (MAX.).

**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

**Caution The watchdog timer is also reset in the case of an internal reset of the watchdog timer.**

#### <2> With crystal/ceramic oscillation clock


Crystal/ceramic oscillation clock
CPU clock
Normal operation in progress
Reset period (oscillation stops)
Oscillation stabilization time ($2^{10}$/fx to $2^{17}$/fx)
Normal operation (reset processing, CPU clock)
Operation stops because option byte is referenced[Note 1].
Watchdog overflow
Internal reset signal
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

**Notes 1.** The operation stop time is 276 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.074 ms (MAX.).

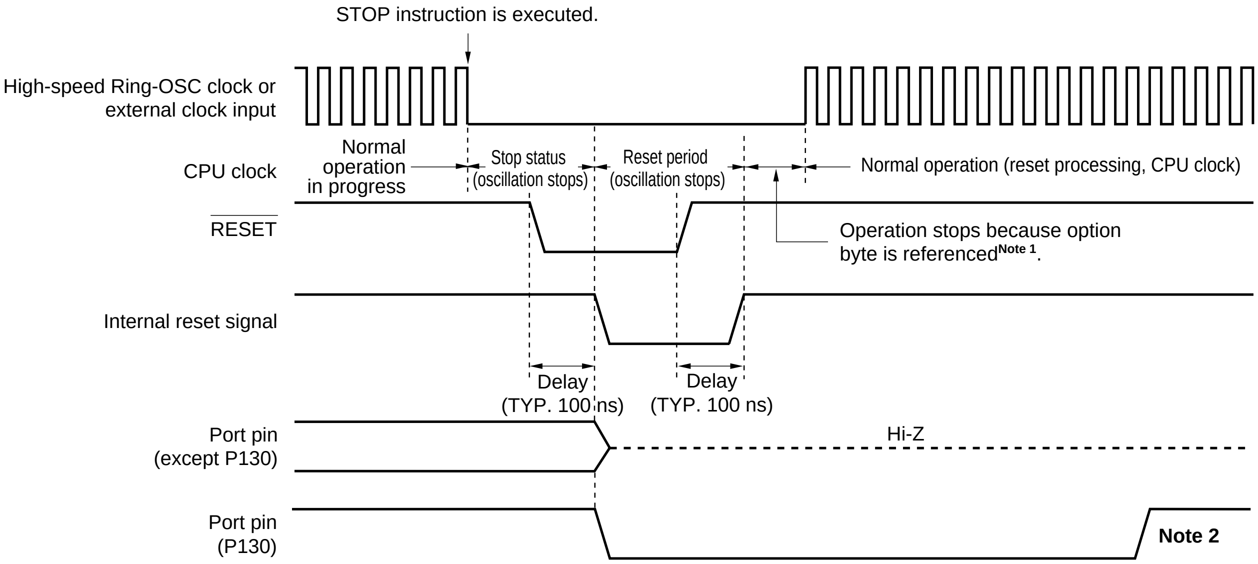
**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

**Caution The watchdog timer is also reset in the case of an internal reset of the watchdog timer.**

**Remark** fx: System clock oscillation frequency

### Figure 15-4. Reset Timing by $\overline{\text{RESET}}$ Input in STOP Mode

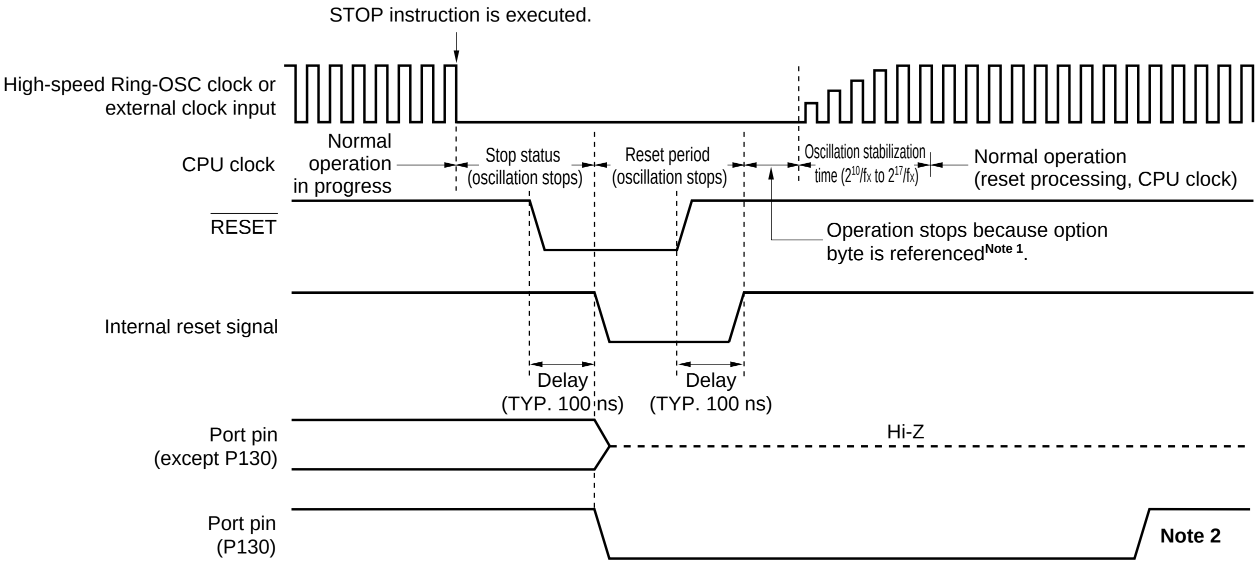
#### <1> With high-speed Ring-OSC clock or external clock input


STOP instruction is executed.
High-speed Ring-OSC clock or external clock input
CPU clock
Normal operation in progress
Stop status (oscillation stops)
Reset period (oscillation stops)
Normal operation (reset processing, CPU clock)
$\overline{\text{RESET}}$
Operation stops because option byte is referenced[Note 1].
Internal reset signal
Delay (TYP. 100 ns)
Delay (TYP. 100 ns)
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

**Notes 1.** The operation stop time is 277 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.075 ms (MAX.).

**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

#### <2> With crystal/ceramic oscillation clock


STOP instruction is executed.
High-speed Ring-OSC clock or external clock input
CPU clock
Normal operation in progress
Stop status (oscillation stops)
Reset period (oscillation stops)
Oscillation stabilization time ($2^{10}$/fx to $2^{17}$/fx)
Normal operation (reset processing, CPU clock)
$\overline{\text{RESET}}$
Operation stops because option byte is referenced[Note 1].
Internal reset signal
Delay (TYP. 100 ns)
Delay (TYP. 100 ns)
Port pin (except P130)
Hi-Z
Port pin (P130)
Note 2

**Notes 1.** The operation stop time is 276 $\mu$s (MIN.), 544 $\mu$s (TYP.), and 1.074 ms (MAX.).

**2.** When reset is effected, P130 outputs a low level. If P130 is set to output a high level before reset is effected, the output signal of P130 can be dummy-output as the reset signal to the CPU.

**Remarks 1.** For the reset timing of the power-on-clear circuit and low-voltage detector, refer to **CHAPTER 16 POWER-ON-CLEAR CIRCUIT** and **CHAPTER 17 LOW-VOLTAGE DETECTOR**.

**2.** fx: System clock oscillation frequency

**Table 15-1. Hardware Statuses After Reset Acknowledgment (1/2)**

| Hardware | | Status After Reset |
|---|---|---|
| Program counter (PC) [Note 1] | | Contents of reset vector table (0000H and 0001H) are set. |
| Stack pointer (SP) | | Undefined |
| Program status word (PSW) | | 02H |
| RAM | Data memory | Undefined [Note 2] |
| | General-purpose registers | Undefined [Note 2] |
| Ports (P0, P2 to P4, P12, P13) (output latches) | | 00H |
| Port mode registers (PM0, PM2 to PM4, PM12) | | FFH |
| Port mode control register (PMC2) | | 00H |
| Pull-up resistor option registers (PU0, PU2, PU3, PU4, PU12) | | 00H |
| Processor clock control register (PCC) | | 02H |
| Preprocessor clock control register (PPCC) | | 02H |
| Low-speed Ring-OSC mode register (LSRCM) | | 00H |
| Oscillation stabilization time select register (OSTS) | | Undefined |
| 16-bit timer 00 | Timer counter 00 (TM00) | 0000H |
| | Capture/compare registers 000, 010 (CR000, CR010) | 0000H |
| | Mode control register 00 (TMC00) | 00H |
| | Prescaler mode register 00 (PRM00) | 00H |
| | Capture/compare control register 00 (CRC00) | 00H |
| | Timer output control register 00 (TOC00) | 00H |
| 8-bit timer 80 | Timer counter 80 (TM80) | 00H |
| | Compare register (CR80) | Undefined |
| | Mode control register 80 (TMC80) | 00H |
| 8-bit timer H1 | Compare registers (CMP01, CMP11) | 00H |
| | Mode register 1 (TMHMD1) | 00H |
| Watchdog timer | Mode register (WDTM) | 67H |
| | Enable register (WDTE) | 9AH |
| A/D converter | Conversion result registers (ADCR, ADCRH) | Undefined |
| | Mode register (ADM) | 00H |
| | Analog input channel specification register (ADS) | 00H |

**Notes 1.** Only the contents of PC are undefined while reset is being generated and while the oscillation stabilization time elapses. The statuses of the other hardware units remain unchanged.

**2.** The status after reset is held in the standby mode.

Table 15-1. Hardware Statuses After Reset Acknowledgment (2/2)

| Hardware | | Status After Reset |
|---|---|---|
| Serial interface UART6 | Receive buffer register 6 (RXB6) | FFH |
| | Transmit buffer register 6 (TXB6) | FFH |
| | Asynchronous serial interface operation mode register 6 (ASIM6) | 01H |
| | Asynchronous serial interface reception error status register 6 (ASIS6) | 00H |
| | Asynchronous serial interface transmission error status register 6 (ASIF6) | 00H |
| | Clock select register 6 (CKSR6) | 00H |
| | Baud rate generator control register 6 (BRGC6) | FFH |
| | Asynchronous serial interface control register 6 (ASICL6) | 16H |
| | Input select control register (ISC) | 00H |
| Multiplier | 16-bit Multiplication result storage register (MUL0) | Undefined |
| | Data registers (MRA0, MRB0) | Undefined |
| | Control register (MULC0) | 00H |
| Reset function | Reset control flag register (RESF) | 00H<sup>Note</sup> |
| Low-voltage detector | Low-voltage detection register (LVIM) | 00H<sup>Note</sup> |
| | Low-voltage detection level select register (LVIS) | 00H<sup>Note</sup> |
| Interrupt | Request flag registers (IF0, IF1) | 00H |
| | Mask flag registers (MK0, MK1) | FFH |
| | External interrupt mode registers (INTM0, INTM1) | 00H |
| Flash memory | Flash protect command register (PFCMD) | Undefined |
| | Flash status register (PFS) | 00H |
| | Flash programming mode control register (FLPMC) | Undefined |
| | Flash programming command register (FLCMD) | 00H |
| | Flash address pointer L (FLAPL) | Undefined |
| | Flash address pointer H (FLAPH) | |
| | Flash address pointer H compare register (FLAPHC) | 00H |
| | Flash address pointer L compare register (FLAPLC) | 00H |
| | Flash write buffer register (FLW) | 00H |

**Note** These values change as follows depending on the reset source.

| Reset Source / Register | $\overline{\text{RESET}}$ Input | Reset by POC | Reset by WDT | Reset by LVI |
|---|---|---|---|---|
| RESF | See **Table 15-2**. | | | |
| LVIM | Cleared (00H) | Cleared (00H) | Cleared (00H) | Held |
| LVIS | | | | |

## 15.1 Register for Confirming Reset Source

Many internal reset generation sources exist in the 78K0S/KB1+. The reset control flag register (RESF) is used to store which source has generated the reset request.

RESF can be read by an 8-bit memory manipulation instruction.

Reset signal generation by $\overline{\text{RESET}}$ input or power-on-clear (POC) circuit, and reading RESF clear RESF to 00H.

**Figure 15-5. Format of Reset Control Flag Register (RESF)**

Address: FF54H After reset: 00H[Note] R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RESF | 0 | 0 | 0 | WDTRF | 0 | 0 | 0 | LVIRF |

| WDTRF | Internal reset request by watchdog timer (WDT) |
|---|---|
| 0 | Internal reset request is not generated, or RESF is cleared. |
| 1 | Internal reset request is generated. |

| LVIRF | Internal reset request by low-voltage detector (LVI) |
|---|---|
| 0 | Internal reset request is not generated, or RESF is cleared. |
| 1 | Internal reset request is generated. |

**Note** The value after reset varies depending on the reset source.

**Caution Do not read data by a 1-bit memory manipulation instruction.**

The status of RESF when a reset request is generated is shown in Table 15-2.

**Table 15-2. RESF Status When Reset Request Is Generated**

| Reset Source<br>Flag | $\overline{\text{RESET}}$ Input | Reset by POC | Reset by WDT | Reset by LVI |
|---|---|---|---|---|
| WDTRF | Cleared (0) | Cleared (0) | Set (1) | Held |
| LVIRF | | | Held | Set (1) |

# CHAPTER 16 POWER-ON-CLEAR CIRCUIT

## 16.1 Functions of Power-on-Clear Circuit

The power-on-clear circuit (POC) has the following functions.

- Generates internal reset signal at power on.
- Compares supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$ = 2.1 V ±0.1 V), and generates internal reset signal when $V_{DD} < V_{POC}$.
- Compares supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$ = 2.1 V ±0.1 V), and releases internal reset signal when $V_{DD} \geq V_{POC}$.

**Cautions 1. If an internal reset signal is generated in the POC circuit, the reset control flag register (RESF) is cleared to 00H.**

**2. Because the detection voltage ($V_{POC}$) of the POC circuit is in a range of 2.1 V ±0.1 V, use a voltage in the range of 2.2 to 5.5 V.**

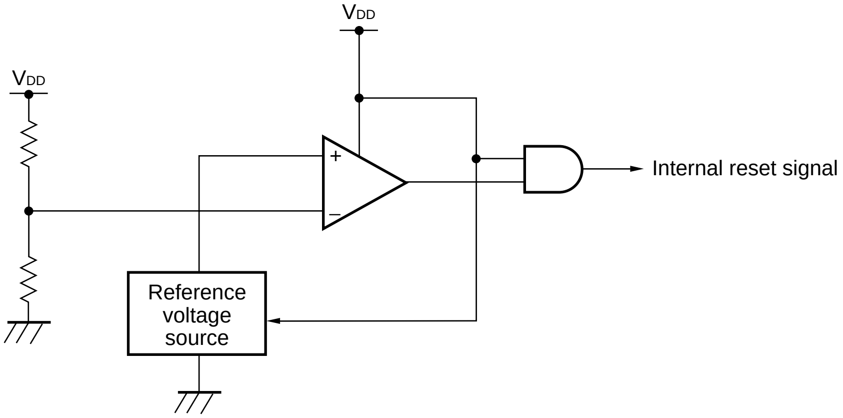
**Remark** This product incorporates multiple hardware functions that generate an internal reset signal. A flag that indicates the reset cause is located in the reset control flag register (RESF) for when an internal reset signal is generated by the watchdog timer (WDT) or low-voltage-detection (LVI) circuit. RESF is not cleared to 00H and the flag is set to 1 when an internal reset signal is generated by WDT or LVI.

For details of RESF, see **CHAPTER 15 RESET FUNCTION**.

## 16.2 Configuration of Power-on-Clear Circuit

The block diagram of the power-on-clear circuit is shown in Figure 16-1.

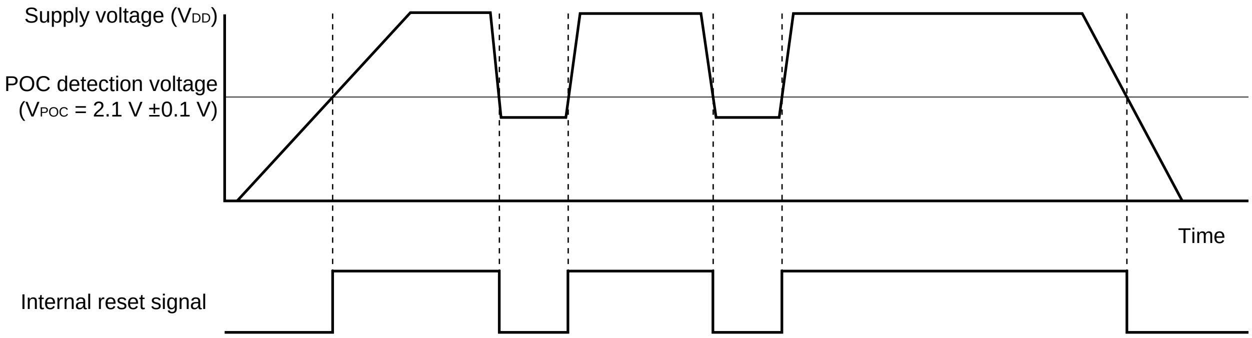
**Figure 16-1. Block Diagram of Power-on-Clear Circuit**



## 16.3 Operation of Power-on-Clear Circuit

In the power-on-clear circuit, the supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$ = 2.1 V ±0.1 V) are compared, and an internal reset signal is generated when $V_{DD} < V_{POC}$, and an internal reset is released when $V_{DD} \geq V_{POC}$.

**Figure 16-2. Timing of Internal Reset Signal Generation in Power-on-Clear Circuit**



**Remark** The internal reset signal is active-low.

## 16.4 Cautions for Power-on-Clear Circuit

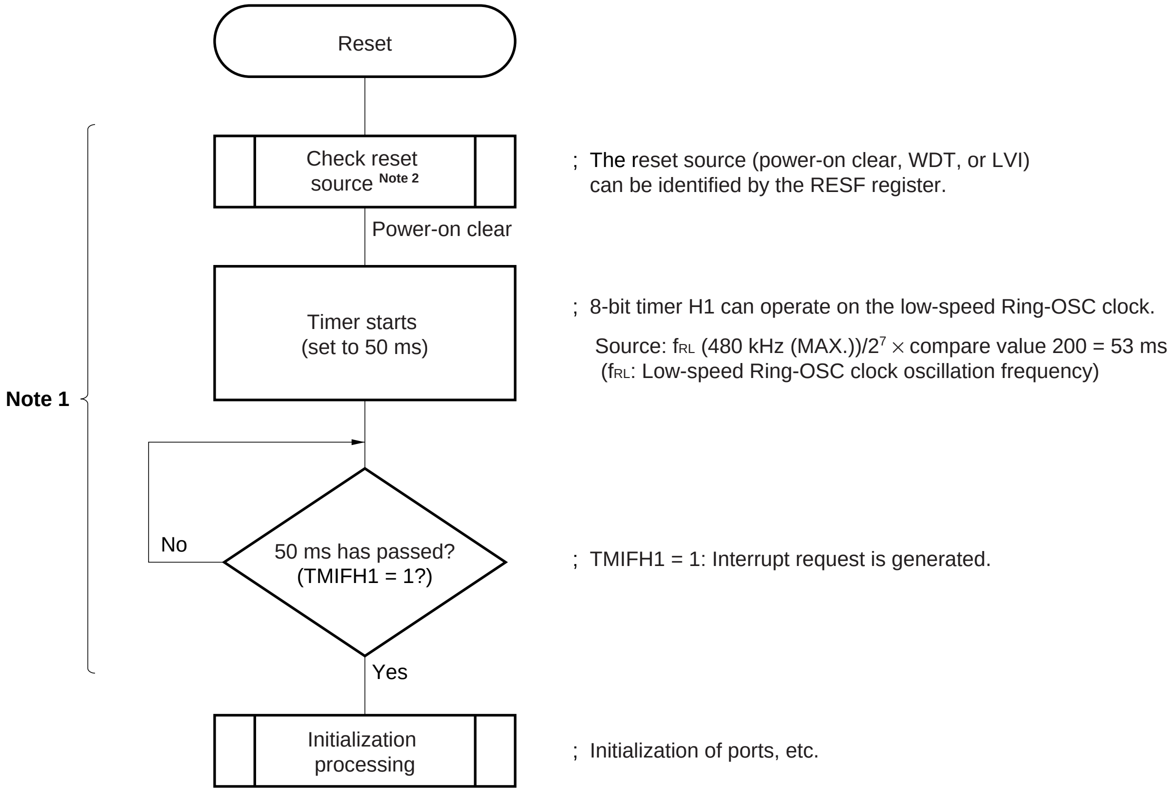
In a system where the supply voltage ($V_{DD}$) fluctuates for a certain period in the vicinity of the POC detection voltage ($V_{POC}$), the system may be repeatedly reset and released from the reset status. In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking the following action.

<Action>

After releasing the reset signal, wait for the supply voltage fluctuation period of each system by means of a software counter that uses a timer, and then initialize the ports.

**Figure 16-3. Example of Software Processing After Release of Reset (1/2)**

- If supply voltage fluctuation is 50 ms or less in vicinity of POC detection voltage

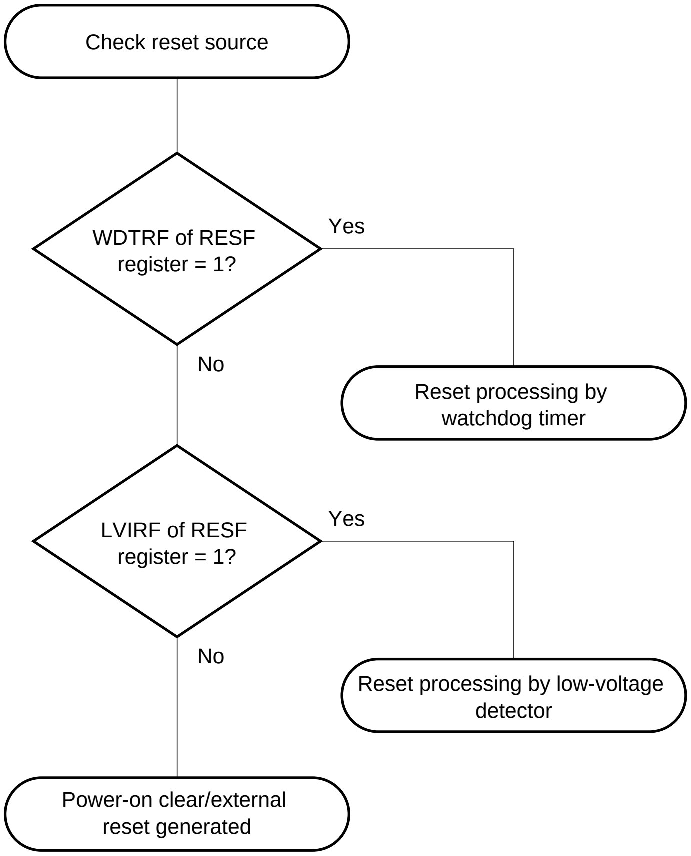

Reset
Check reset source Note 2
Power-on clear
Timer starts (set to 50 ms)
Note 1
No
50 ms has passed? (TMIFH1 = 1?)
Yes
Initialization processing
; The reset source (power-on clear, WDT, or LVI) can be identified by the RESF register.
; 8-bit timer H1 can operate on the low-speed Ring-OSC clock.
Source: fRL (480 kHz (MAX.))/2^7 × compare value 200 = 53 ms
(fRL: Low-speed Ring-OSC clock oscillation frequency)
; TMIFH1 = 1: Interrupt request is generated.
; Initialization of ports, etc.

**Notes 1.** If reset is generated again during this period, initialization processing is not started.

**2.** A flowchart is shown on the next page.

**Figure 16-3. Example of Software Processing After Release of Reset (2/2)**

- Checking reset cause


Check reset source
WDTRF of RESF register = 1?
Yes
No
Reset processing by watchdog timer
LVIRF of RESF register = 1?
Yes
No
Reset processing by low-voltage detector
Power-on clear/external reset generated

# CHAPTER 17 LOW-VOLTAGE DETECTOR

## 17.1 Functions of Low-Voltage Detector

The low-voltage detector (LVI) has following functions.

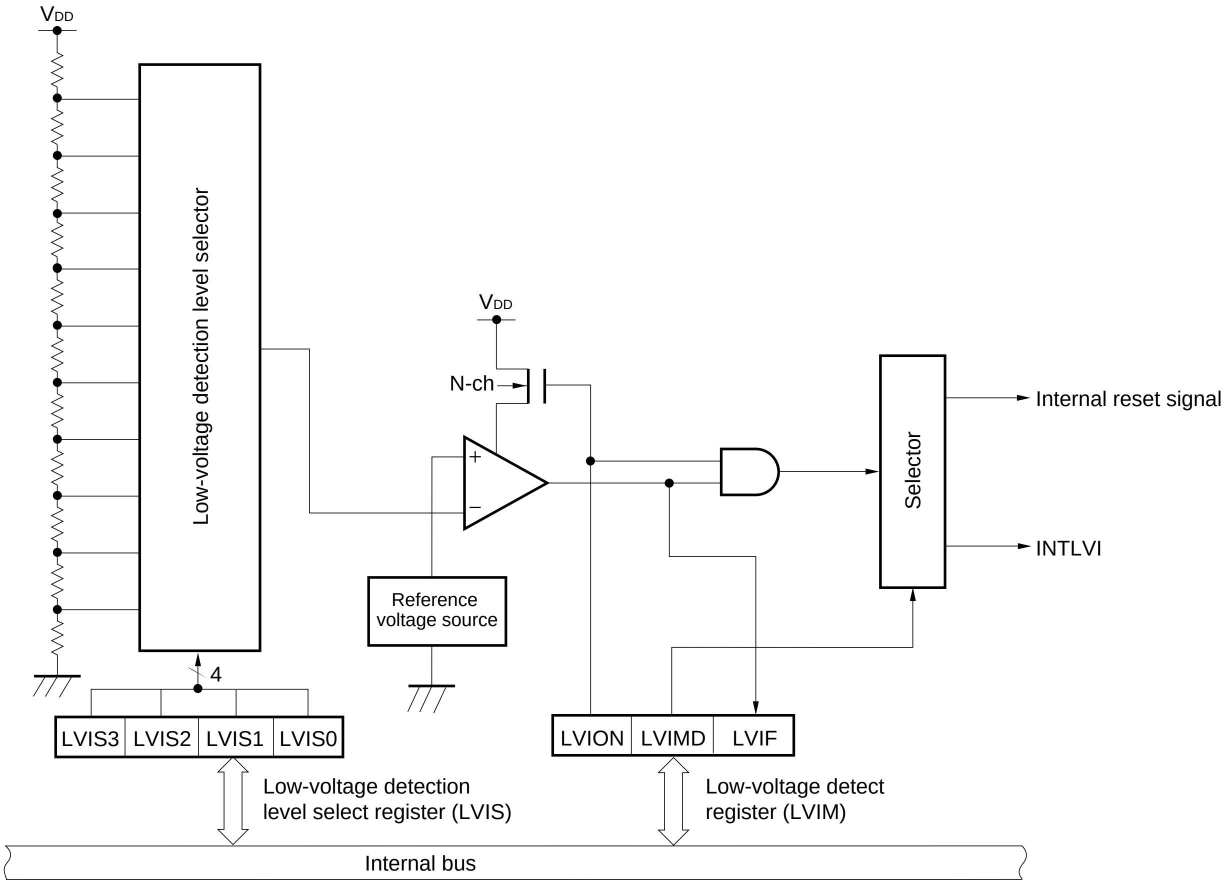
- Compares supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an internal interrupt signal or internal reset signal when $V_{DD} < V_{LVI}$.
- Detection levels (ten levels) of supply voltage can be changed by software.
- Interrupt or reset function can be selected by software.
- Operable in STOP mode.

When the low-voltage detector is used to reset, bit 0 (LVIRF) of the reset control flag register (RESF) is set to 1 if reset occurs. For details of RESF, refer to **CHAPTER 15 RESET FUNCTION**.

## 17.2 Configuration of Low-Voltage Detector

The block diagram of the low-voltage detector is shown in Figure 17-1.

**Figure 17-1. Block Diagram of Low-Voltage Detector**



## 17.3 Registers Controlling Low-Voltage Detector

The low-voltage detector is controlled by the following registers.

- Low-voltage detect register (LVIM)
- Low-voltage detection level select register (LVIS)

### (1) Low-voltage detect register (LVIM)

This register sets low-voltage detection and the operation mode.
This register can be set by a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation clears this register to 00H[Note 1].

**Figure 17-2. Format of Low-Voltage Detect Register (LVIM)**

Address: FF50H After reset: 00H[Note 1] R/W[Note 2]

| Symbol | <7> | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| LVIM | LVION | 0 | 0 | 0 | 0 | 0 | LVIMD | LVIF |

| LVION[Note 3] | Enabling low-voltage detection operation |
|---|---|
| 0 | Disable operation |
| 1 | Enable operation |

| LVIMD | Low-voltage detection operation mode selection |
|---|---|
| 0 | Generate interrupt signal when supply voltage ($V_{DD}$) < detection voltage ($V_{LVI}$) |
| 1 | Generate internal reset signal when supply voltage ($V_{DD}$) < detection voltage ($V_{LVI}$) |

| LVIF[Note 4] | 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.** Retained only after a reset by LVI.

**2.** Bit 0 is a read-only bit.

**3.** When LVION is set to 1, operation of the comparator in the LVI circuit is started. Use software to instigate a wait of at least 0.2 ms from when LVION is set to 1 until the voltage is confirmed at LVIF.

**4.** The value of LVIF is output as the interrupt request signal INTLVI when LVION = 1 and LVIMD = 0.

**Cautions 1. To stop LVI, follow either of the procedures below.**

- **When using 8-bit manipulation instruction: Write 00H to LVIM.**
- **When using 1-bit memory manipulation instruction: Clear LVION to 0.**

**2. Be sure to set bits 2 to 6 to 0.**

### (2) Low-voltage detection level select register (LVIS)

This register selects the low-voltage detection level.

This register can be set by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H[Note].

**Figure 17-3. Format of Low-Voltage Detection Level Select Register (LVIS)**

Address: FF51H, After reset: 00H[Note] R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| LVIS | 0 | 0 | 0 | 0 | LVIS3 | LVIS2 | LVIS1 | LVIS0 |

| LVIS3 | LVIS2 | LVIS1 | LVIS0 | Detection level |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $V_{LVI0}$ (4.3 V ±0.2 V) |
| 0 | 0 | 0 | 1 | $V_{LVI1}$ (4.1 V ±0.2 V) |
| 0 | 0 | 1 | 0 | $V_{LVI2}$ (3.9 V ±0.2 V) |
| 0 | 0 | 1 | 1 | $V_{LVI3}$ (3.7 V ±0.2 V) |
| 0 | 1 | 0 | 0 | $V_{LVI4}$ (3.5 V ±0.2 V) |
| 0 | 1 | 0 | 1 | $V_{LVI5}$ (3.3 V ±0.15 V) |
| 0 | 1 | 1 | 0 | $V_{LVI6}$ (3.1 V ±0.15 V) |
| 0 | 1 | 1 | 1 | $V_{LVI7}$ (2.85 V ±0.15 V) |
| 1 | 0 | 0 | 0 | $V_{LVI8}$ (2.6 V ±0.1 V) |
| 1 | 0 | 0 | 1 | $V_{LVI9}$ (2.35 V ±0.1 V) |
| Other than above | | | | Setting prohibited |

**Note** Retained only after a reset by LVI.

**Caution Bits 4 to 7 must be set to 0.**

## 17.4 Operation of Low-Voltage Detector

The low-voltage detector can be used in the following two modes.

- Used as reset
  Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an internal reset signal when $V_{DD} < V_{LVI}$, and releases internal reset when $V_{DD} \geq V_{LVI}$.
- Used as interrupt
  Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an interrupt signal (INTLVI) when $V_{DD} < V_{LVI}$.

The operation is set as follows.

**(1) When used as reset**

- When starting operation

<1> Mask the LVI interrupt (LVIMK = 1).

<2> Set the detection voltage using bits 3 to 0 (LVIS3 to LVIS0) of the low-voltage detection level select register (LVIS).

<3> Set bit 7 (LVION) of LVIM to 1 (enables LVI operation).

<4> Use software to instigate a wait of at least 0.2 ms.

<5> Wait until “supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$)” at bit 0 (LVIF) of LVIM is confirmed.

<6> Set bit 1 (LVIMD) of LVIM to 1 (generates internal reset signal when supply voltage ($V_{DD}$) < detection voltage ($V_{LVI}$)).

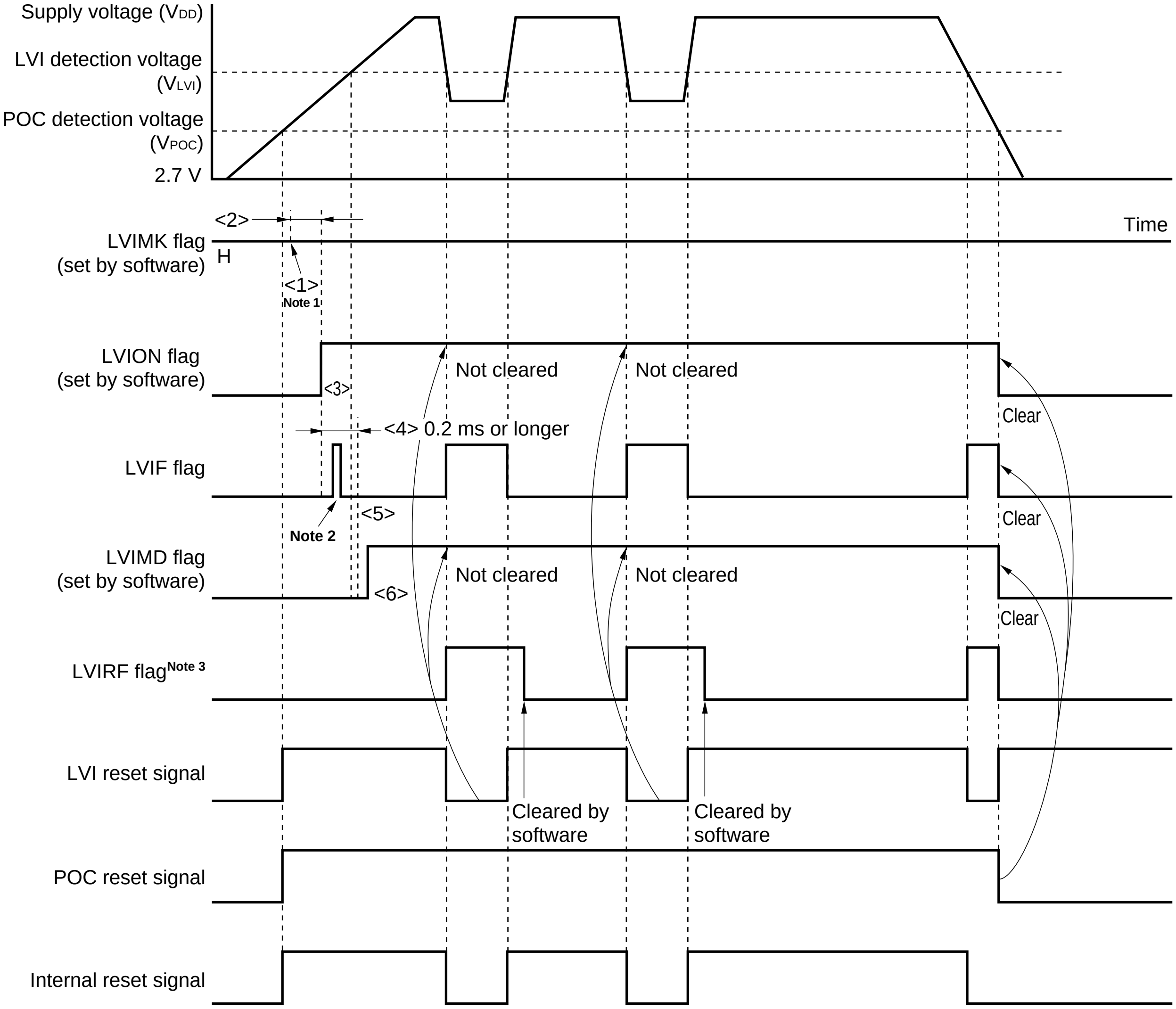
Figure 17-4 shows the timing of generating the internal reset signal of the low-voltage detector. Numbers <1> to <6> in this figure correspond to <1> to <6> above.

**Cautions 1. <1> must always be executed. When LVIMK = 0, an interrupt may occur immediately after the processing in <3>.**

**2. If supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$) when LVIM is set to 1, an internal reset signal is not generated.**

- When stopping operation
  Either of the following procedures must be executed.
  - When using 8-bit memory manipulation instruction: Write 00H to LVIM.
  - When using 1-bit memory manipulation instruction: Clear LVIMD to 0 and LVION to 0 in that order.

Figure 17-4. Timing of Low-Voltage Detector Internal Reset Signal Generation



**Notes 1.** The LVIMK flag is set to “1” by reset signal generation.

**2.** The LVIF flag may be set (1).

**3.** LVIRF is bit 0 of the reset control flag register (RESF). For details of RESF, refer to **CHAPTER 15 RESET FUNCTION**.

**Remark** <1> to <6> in Figure 17-4 above correspond to <1> to <6> in the description of “when starting operation” in **17.4 (1) When used as reset**.

**(2) When used as interrupt**

- When starting operation

<1> Mask the LVI interrupt (LVIMK = 1).

<2> Set the detection voltage using bits 3 to 0 (LVIS3 to LVIS0) of the low-voltage detection level select register (LVIS).

<3> Set bit 7 (LVION) of LVIM to 1 (enables LVI operation).

<4> Use software to instigate a wait of at least 0.2 ms.

<5> Wait until "supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$)" at bit 0 (LVIF) of LVIM is confirmed.

<6> Clear the interrupt request flag of LVI (LVIIF) to 0.

<7> Release the interrupt mask flag of LVI (LVIMK).

<8> Execute the EI instruction (when vector interrupts are used).

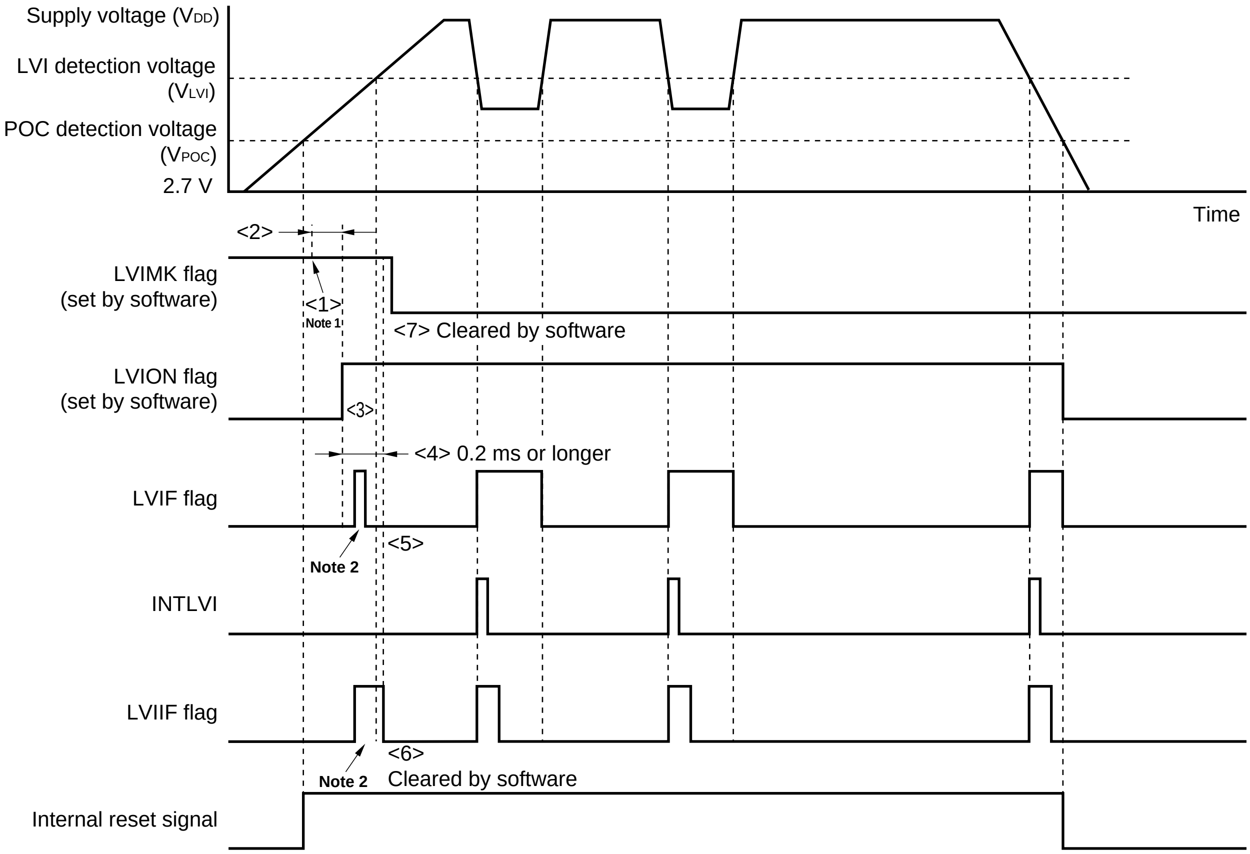
Figure 17-5 shows the timing of generating the interrupt signal of the low-voltage detector. Numbers <1> to <7> in this figure correspond to <1> to <7> above.

- When stopping operation

Either of the following procedures must be executed.

- When using 8-bit memory manipulation instruction: Write 00H to LVIM.
- When using 1-bit memory manipulation instruction: Clear LVION to 0.

**Figure 17-5. Timing of Low-Voltage Detector Interrupt Signal Generation**



**Notes 1.** The LVIMK flag is set to "1" by reset signal generation.

**2.** The LVIF and LVIIF flags may be set (1).

**Remark** <1> to <7> in Figure 17-5 above correspond to <1> to <7> in the description of "when starting operation" in **17.4 (2) When used as interrupt**.

## 17.5 Cautions for Low-Voltage Detector

In a system where the supply voltage ($V_{DD}$) fluctuates for a certain period in the vicinity of the LVI detection voltage ($V_{LVI}$), the operation is as follows depending on how the low-voltage detector is used.

**<1> When used as reset**

The system may be repeatedly reset and released from the reset status.

In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking action (1) below.

**<2> When used as interrupt**

Interrupt requests may be frequently generated. Take action (2) below.

In this system, take the following actions.

<Action>

**(1) When used as reset**

After releasing the reset signal, wait for the supply voltage fluctuation period of each system by means of a software counter that uses a timer, and then initialize the ports (see **Figure 17-6**).

**(2) When used as interrupt**

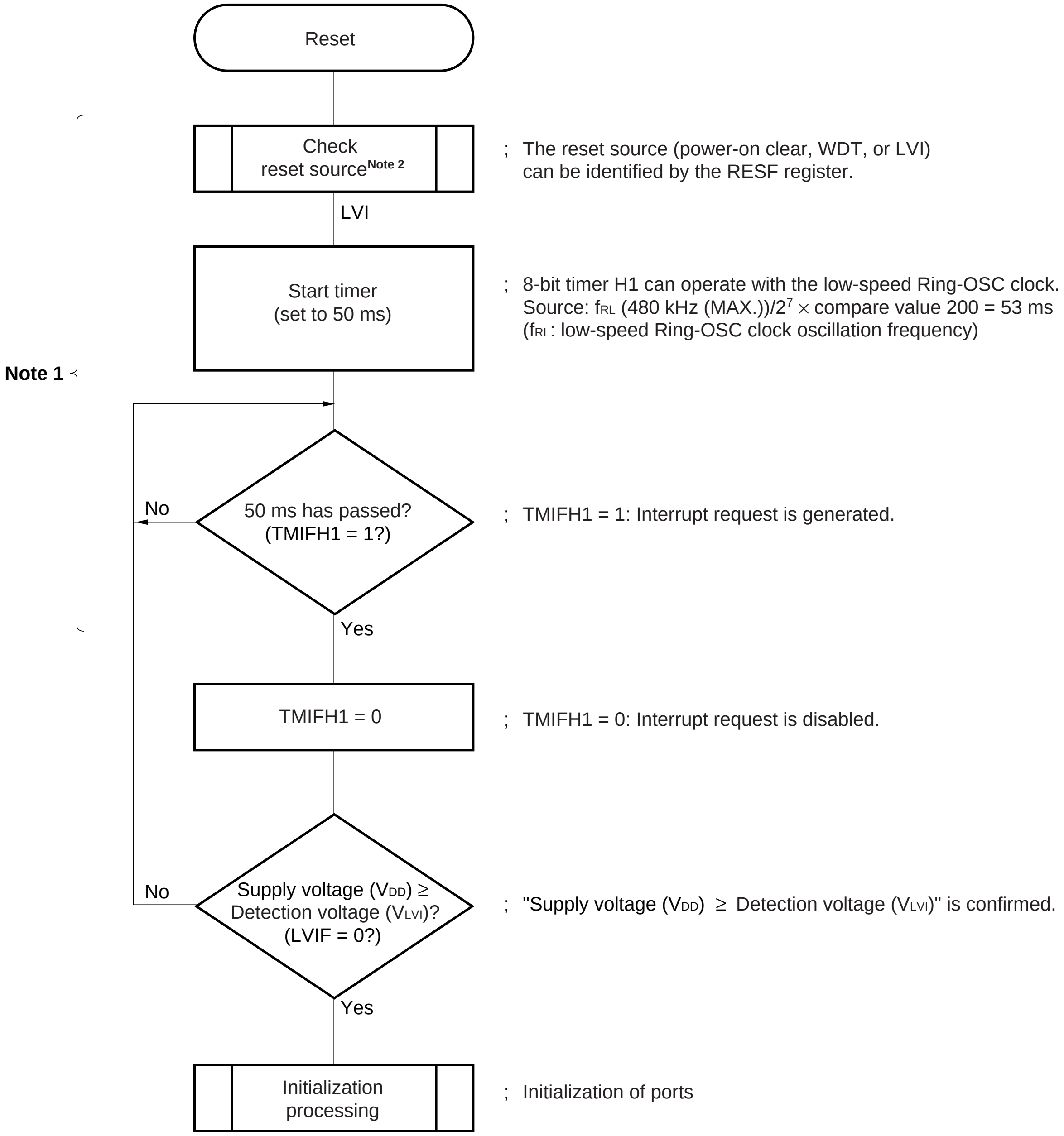
Perform the processing[Note] for low voltage detection. Check that “supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$)” in the servicing routine of the LVI interrupt by using bit 0 (LVIF) of the low-voltage detection register (LVIM). Clear bit 1 (LVIIF) of interrupt request flag register 0 (IF0) to 0 and enable interrupts (EI).

In a system where the supply voltage fluctuation period is long in the vicinity of the LVI detection voltage, wait for the supply voltage fluctuation period, check that “supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$)” using the LVIF flag. Clear the LVIIF flag to 0, and then enable interrupts (EI).

**Note** For low voltage detection processing, the CPU clock speed is switched to slow speed and the A/D converter is stopped, etc.

**Figure 17-6. Example of Software Processing After Release of Reset (1/2)**

- If supply voltage fluctuation is 50 ms or less in vicinity of LVI detection voltage

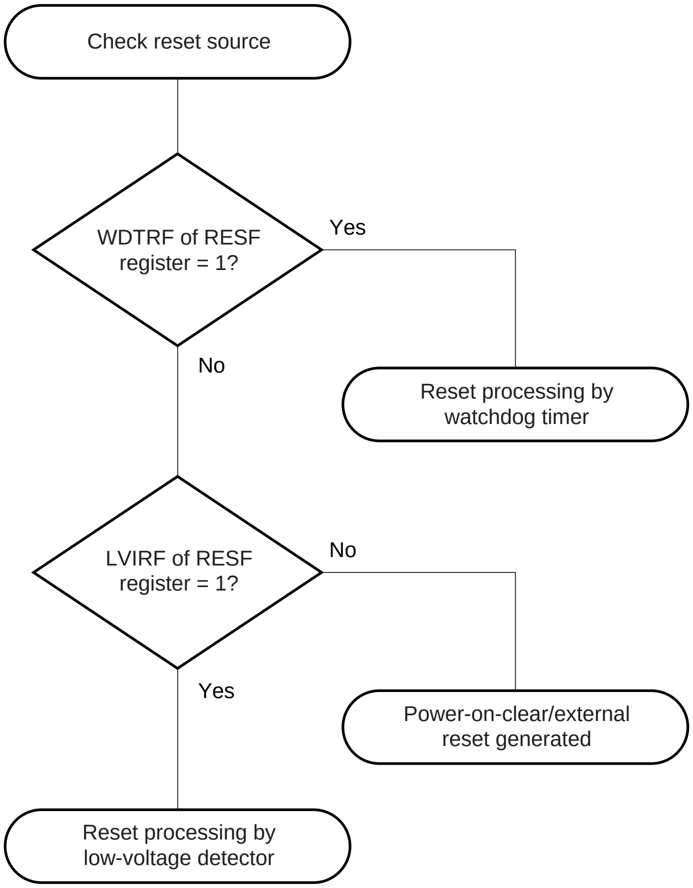

Reset
Check reset source Note 2
; The reset source (power-on clear, WDT, or LVI) can be identified by the RESF register.
LVI
Start timer (set to 50 ms)
; 8-bit timer H1 can operate with the low-speed Ring-OSC clock. Source: fRL (480 kHz (MAX.))/2^7 × compare value 200 = 53 ms (fRL: low-speed Ring-OSC clock oscillation frequency)
Note 1
No
50 ms has passed? (TMIFH1 = 1?)
; TMIFH1 = 1: Interrupt request is generated.
Yes
TMIFH1 = 0
; TMIFH1 = 0: Interrupt request is disabled.
No
Supply voltage (VDD) ≥ Detection voltage (VLVI)? (LVIF = 0?)
; "Supply voltage (VDD) ≥ Detection voltage (VLVI)" is confirmed.
Yes
Initialization processing
; Initialization of ports

**Notes 1.** If reset is generated again during this period, initialization processing is not started.

**2.** A flowchart is shown on the next page.

Figure 17-6. Example of Software Processing After Release of Reset (2/2)

- Checking reset source


Check reset source
WDTRF of RESF register = 1?
Yes
No
Reset processing by watchdog timer
LVIRF of RESF register = 1?
No
Yes
Power-on-clear/external reset generated
Reset processing by low-voltage detector

# CHAPTER 18 OPTION BYTE

The 78K0S/KB1+ has an area called an option byte at address 0080H of the flash memory. When using the product, be sure to set the following functions by using the option byte.

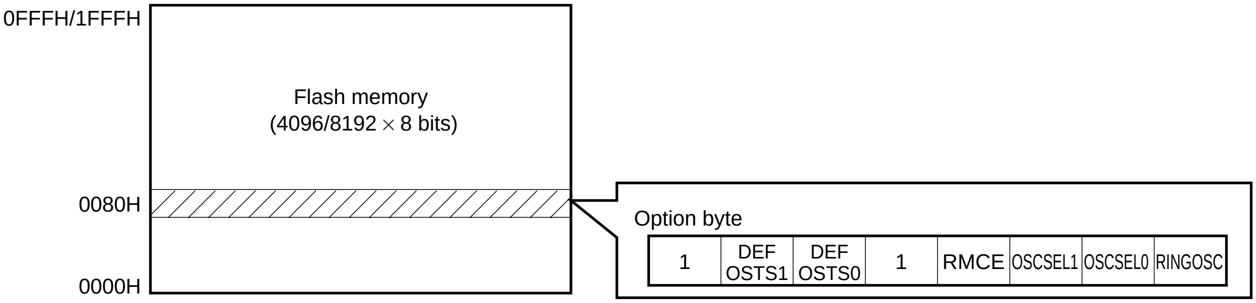
1. **Selection of system clock source**
   - High-speed Ring-OSC clock
   - Crystal/ceramic oscillation clock
   - External clock input

2. **Low-speed Ring-OSC clock oscillation**
   - Cannot be stopped.
   - Can be stopped by software.

3. **Control of $\overline{\text{RESET}}$ pin**
   - Used as $\overline{\text{RESET}}$ pin
   - $\overline{\text{RESET}}$ pin is used as an input port pin (P34).

4. **Oscillation stabilization time on power application or after reset release**
   - $2^{10}/f_X$
   - $2^{12}/f_X$
   - $2^{15}/f_X$
   - $2^{17}/f_X$

**Figure 18-1. Positioning of Option Byte**



### Figure 18-2. Format of Option Byte (1/2)

Address: 0080H

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | DEFOSTS1 | DEFOSTS0 | 1 | RMCE | OSCSEL1 | OSCSEL0 | RINGOSC |

| RINGOSC | Low-speed Ring-OSC clock oscillation |
|---|---|
| 1 | Cannot be stopped (oscillation does not stop even if 1 is written to the LSRSTOP bit) |
| 0 | Can be stopped by software (oscillation stops when 1 is written to the LSRSTOP bit) |

**Cautions 1. If it is selected that low-speed Ring-OSC clock oscillation cannot be stopped, the count clock to the watchdog timer (WDT) is fixed to low-speed Ring-OSC.**

**2. If it is selected that low-speed Ring-OSC can be stopped by software, supply of the count clock to WDT is stopped in the HALT/STOP mode, regardless of the setting of bit 0 (LSRSTOP) of the low-speed Ring-OSC mode register (LSRCM).** Similarly, clock supply is also stopped when a clock other than the low-speed Ring-OSC is selected as a count clock to WDT. **If low-speed Ring-OSC is selected as the count clock to 8-bit timer H1, however, the count clock is supplied in the HALT/STOP mode while low-speed Ring-OSC operates (LSRSTOP = 0).**

| OSCSEL1 | OSCSEL0 | Selection of system clock source |
|---|---|---|
| 0 | 0 | Crystal/ceramic oscillation clock |
| 0 | 1 | External clock input |
| 1 | × | High-speed Ring-OSC clock |

**Caution Because the X1 and X2 pins are also used as the P121 and P122 pins, the conditions under which the X1 and X2 pins can be used differ depending on the selected system clock source.**

**(1) High-speed Ring-OSC clock**
**P121 and P122 can be used as I/O port pins.**

**(2) Crystal/ceramic oscillation clock**
**The X1 and X2 pins cannot be used as I/O port pins because they are used as clock input pins.**

**(3) External clock input**
**Because the X1 pin is used as an external clock input pin, P121 cannot be used as an I/O port pin.**

**Remark** × : don’t care

| RMCE | Control of $\overline{\text{RESET}}$ pin |
|---|---|
| 1 | $\overline{\text{RESET}}$ pin is used as is. |
| 0[Note] | $\overline{\text{RESET}}$ pin is used as input port pin (P34). |

**Note** When clearing the RMCE to 0, connect pull-up resistor.

**Caution If a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the power-on clear function and before the option byte is referenced again, the 78K0S/KB1+ is reset, and the status is held until a high level is input to the $\overline{\text{RESET}}$ pin.**

**Figure 18-2. Format of Option Byte (2/2)**

| DEFOSTS1 | DEFOSTS0 | Oscillation stabilization time on power application or after reset release |
|---|---|---|
| 0 | 0 | $2^{10}$/fx (102.4 $\mu$s) |
| 0 | 1 | $2^{12}$/fx (409.6 $\mu$s) |
| 1 | 0 | $2^{15}$/fx (3.27 ms) |
| 1 | 1 | $2^{17}$/fx (13.1 ms) |

**Caution The setting of this option is valid only when the crystal/ceramic oscillation clock is selected as the system clock source. No wait time elapses if the high-speed Ring-OSC or external clock input is selected as the system clock source.**

**Remarks 1.** ( ): $f_X$ = 10 MHz

**2.** For the oscillation stabilization time of the resonator, refer to the characteristics of the resonator to be used.

An example of software coding for setting the option bytes is shown below.

```
OPT OSEG AT 0080H
DB 10010101B            ; Sets to option byte
                        ; Low-Speed Ring-OSC cannot be stopped
                        ; Selects the high-speed Ring-OSC as the system clock source
                        ; Uses RESET pin as bit input-only port (P34)
                        ; The oscillation stabilization time is minimum (2^10/fX)
```

# CHAPTER 19 FLASH MEMORY

## 19.1 Features

The internal flash memory of the 78K0S/KB1+ has the following features.

○ Erase/write with a single power supply

○ Capacity: 4 KB/8 KB
- Erase unit: 1 block (256 bytes)
- Write unit: 1 byte

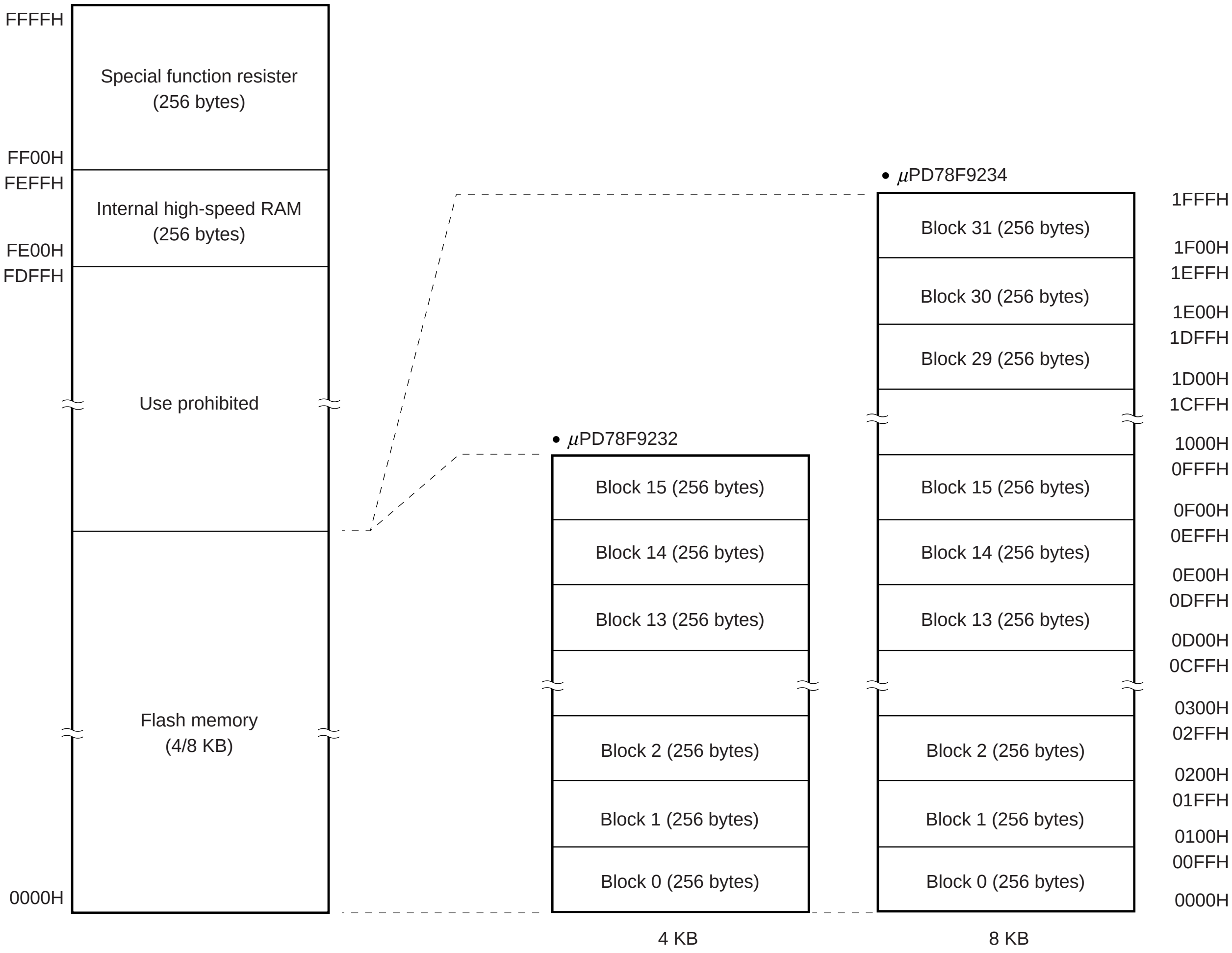
○ Rewriting method
- Rewriting by communication with dedicated flash programmer (on-board/off-board programming)
- Rewriting flash memory by user program (self programming)

○ Flash memory write prohibit function supported (security function)

## 19.2 Memory Configuration

The 4/8 KB internal flash memory area is divided into 16/32 blocks and can be programmed/erased in block units. All the blocks can also be erased at once.

**Figure 19-1. Flash Memory Mapping**



## 19.3 Functional Outline

The internal flash memory of the 78K0S/KB1+ can be rewritten by using the rewrite function of the dedicated flash programmer, regardless of whether the 78K0S/KB1+ has already been mounted on the target system or not (on-board/off-board programming).

The function for rewriting a program with the user program (self programming), which is ideal for an application when it is assumed that the program is changed after production/shipment of the target system, is provided.

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.

Refer to **19.7.4 Security settings** for details on the security function.

**Table 19-1. 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. | Self programming mode |

**Remarks 1.** The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

**2.** Refer to the following sections for details on the flash memory writing control function.

- **19.7 On-Board and Off-Board Flash Memory Programming**
- **19.8 Flash Memory Programming by Self Writing**

## 19.4 Writing with Flash Programmer

The following two types of dedicated flash programmers can be used for writing data to the internal flash memory of the 78K0S/KB1+.

- FlashPro4 (PG-FP4, FL-PR4)
- PG-FPL2

Data can be written to the flash memory on-board or off-board, by using a dedicated flash programmer.

**(1) On-board programming**

The contents of the flash memory can be rewritten after the 78K0S/KB1+ has been mounted on the target system. The connectors that connect the dedicated flash programmer and the test pad must be mounted on the target system. The test pad is required only when writing data with the crystal/ceramic resonator mounted (refer to Figure 19-5 for mounting of the test pad).

**(2) Off-board programming**

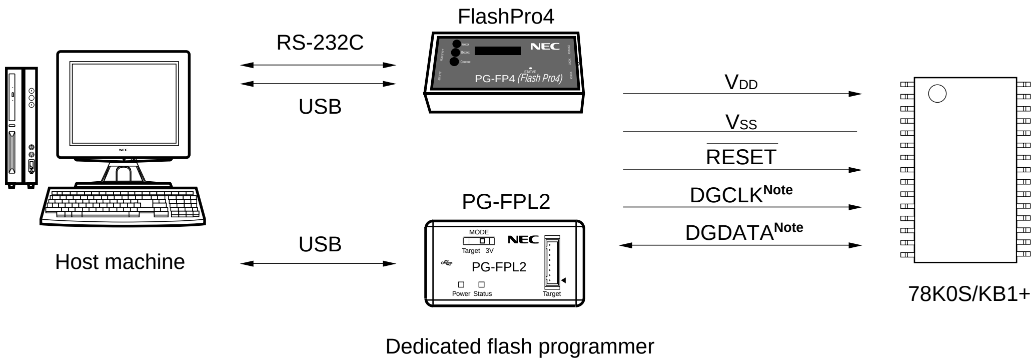
Data can be written to the flash memory with a dedicated program adapter (FA series) before the 78K0S/KB1+ is mounted on the target system.

**Remark** The FL-PR4 and FA series are products of Naito Densei Machida Mfg. Co., Ltd.

## 19.5 Programming Environment

The environment required for writing a program to the flash memory is illustrated below.

**Figure 19-2. Environment for Writing Program to Flash Memory**



**Note** DGCLK and DGDATA are single-wire bidirectional communication interfaces. They use UART as the communication mode.

A host machine that controls the dedicated flash programmer is necessary. When using the PG-FP4 or FL-PR4, data can be written with just the dedicated flash programmer after downloading the program from the host machine.

UART is used for manipulation such as writing and erasing when interfacing between the dedicated flash programmer and the 78K0S/KB1+. To write the flash memory off-board, a dedicated program adapter (FA series) is necessary.

Download the latest programmer firmware, GUI, and parameter file from the download site for development tools (http://www.necel.com/micro/ods/jpn/index.html).

**Table 19-2. Wiring Between 78K0S/KB1+ and FlashPro4**

| FlashPro4 Connection Pin | | | 78K0S/KB1+ Connection Pin | |
|---|---|---|---|---|
| Pin Name | I/O | Pin Function | Pin Name | Pin No. |
| CLK[Note] | Output | Clock to 78K0S/KB1+ | X1/P121 | 8 |
| FLMD0[Note] | Output | On-board mode signal | | |
| SI/RxD[Note] | Input | Receive signal | X2/P122 | 9 |
| SO/TxD[Note] | Output | Receive signal/on-board mode signal | | |
| /RESET | Output | Reset signal | $\overline{\text{RESET}}$/P34 | 10 |
| $V_{DD}$ | – | $V_{DD}$ voltage generation/voltage monitor | $V_{DD}$ | 7 |
| GND | – | Ground | $V_{SS}$ | 6 |

**Note** In the 78K0S/KB1+, the CLK and FLMD0 signals are connected to the X1 pin and the SI/RxD and SO/TxD signals to the X2 signal; therefore, these signals need to be directly connected.

**Figure 19-3. Communication with FlashPro4**

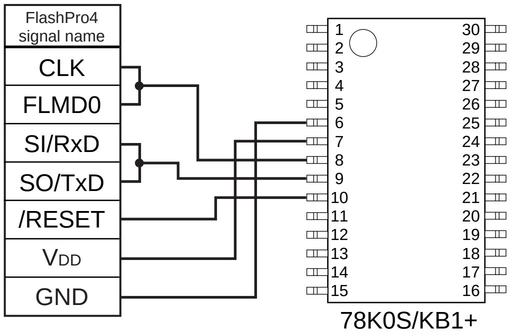
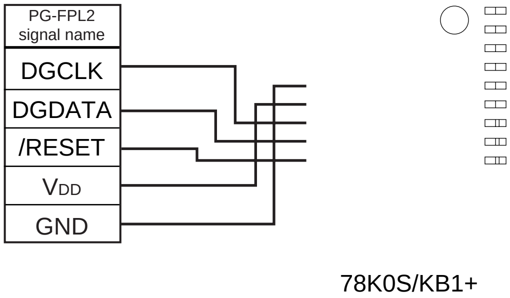


**Table 19-3. Wiring Between 78K0S/KB1+ and PG-FPL2**

| PG-FPL2 Connection Pin | | | 78K0S/KB1+ Connection Pin | |
|---|---|---|---|---|
| Pin Name | I/O | Pin Function | Pin Name | Pin No. |
| CLK | Output | Clock to 78K0S/KB1+ | X1/P121 | 8 |
| DGDATA | I/O | Transmit/receive signal, on-board mode signal | X2/P122 | 9 |
| /RESET | Output | Reset signal | RESET/P34 | 10 |
| $V_{DD}$ | I/O | $V_{DD}$ voltage generation | $V_{DD}$ | 7 |
| GND | – | Ground | $V_{SS}$ | 6 |

**Figure 19-4. Communication with PG-FPL2**



## 19.6 Pin Connection on Board

To write the flash memory on-board, connectors that connect the dedicated flash programmer must be provided on the target system. First provide a function that selects the normal operation mode or flash memory programming mode on the board.

When the flash memory programming mode is set, all the pins not used for programming the flash memory are in the same status as immediately after reset. Therefore, if the external device does not recognize the state immediately after reset, the pins must be processed as described below.

The state of the pins in the self programming mode is the same as that in the HALT mode.

### 19.6.1 X1 and X2 pins

The X1 and X2 pins are used as the serial interface of flash memory programming. Therefore, if the X1 and X2 pins are connected to an external device, a signal conflict occurs. To prevent the conflict of signals, isolate the connection with the external device.

Perform the following processing (1) and (2) when on-board writing is performed with the resonator mounted, when it is difficult to isolate the resonator, while a crystal or ceramic resonator is selected as the system clock.

(1) Mount the minimum-possible test pads between the device and the resonator, and connect the flash programmer via the test pad. Keep the wiring as short as possible (refer to **Figure 19-5** and **Table 19-4**).

(2) Set the oscillation frequency of the communication clock for writing using the GUI software of the dedicated flash programmer. Research the series/parallel resonant and antiresonant frequencies of the resonator used, and set the oscillation frequency so that it is outside the range of the resonant frequency ±10% (refer to **Figure 19-6** and **Table 19-5**).

**Figure 19-5. Example of Mounting Test Pads**

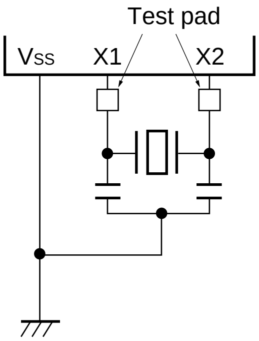
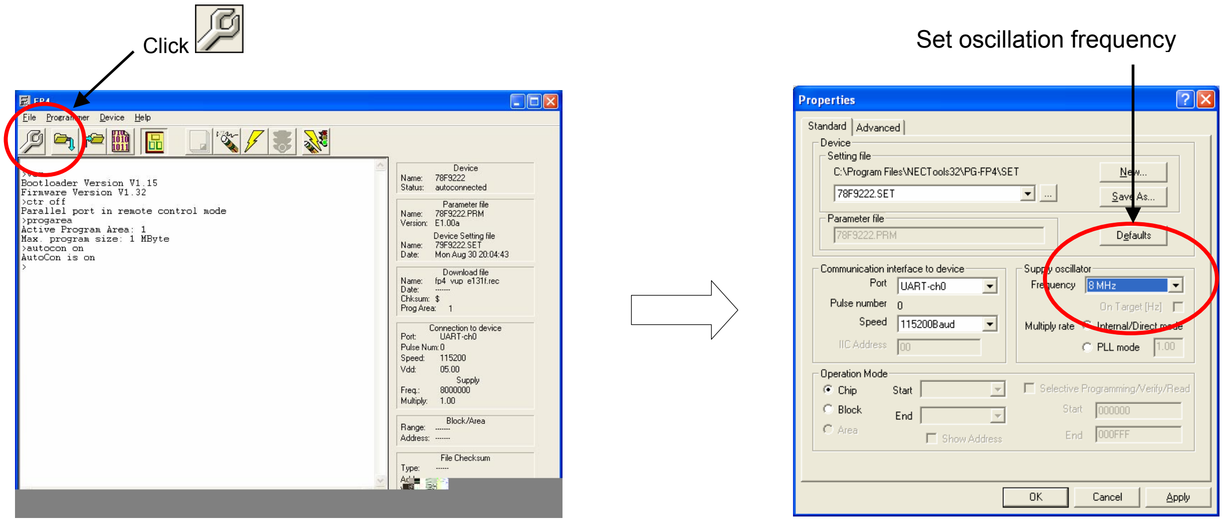


**Table 19-4. Clock to Be Used and Mounting of Test Pads**

| Clock to Be Used | | Mounting of Test Pads |
|---|---|---|
| High-speed Ring-OSC clock | | Not required |
| External clock | | |
| Crystal/ceramic oscillation clock | Before resonator is mounted | |
| | After resonator is mounted | Required |

**Figure 19-6. PG-FP4 GUI Software Setting Example**



Click

Set oscillation frequency

```
Bootloader Version V1.15
Firmware Version V1.32
>ctr off
Parallel port in remote control mode
>progarea
Active Program Area: 1
Max. program size: 1 MByte
>autocon on
AutoCon is on
>
```

Device
Name: 78F9222
Status: autoconnected

Parameter file
Name: 78F9222.PRM
Version: E1.00a

Device Setting file
Name: 79F9222.SET
Date: Mon Aug 30 20:04:43

Download file
Name: fp4 vup e131f.rec
Date: -------
Chksum: $
Prog Area: 1

Connection to device
Port: UART-ch0
Pulse Num: 0
Speed: 115200
Vdd: 05.00

Supply
Freq.: 8000000
Multiply: 1.00

Block/Area
Range: -------
Address: -------

File Checksum
Type: ------

(Main window)

Device

Setting file
C:\Program Files\NECTools32\PG-FP4\SET
78F9222.SET

Parameter file
78F9222.PRM

Communication interface to device
Port: UART-ch0
Pulse number: 0
Speed: 115200Baud
IIC Address: 00

Supply oscillator
Frequency: 8 MHz
On Target [Hz] □
Multiply rate: ○ Internal/Direct mode
○ PLL mode 1.00

Operation Mode
● Chip
○ Block
○ Area
Start
End
□ Show Address
□ Selective Programming/Verify/Read
Start 000000
End 000FFF

(Standard tab in Device setup window)

**Table 19-5. Oscillation Frequency and PG-FP4 GUI Software Setting Value Example**

| Oscillation Frequency | PG-FP4 GUI Software Setting Value Example<br>(Communication Frequency) |
|---|---|
| 1 MHz ≤ fx < 4 MHz | 8 MHz |
| 4 MHz ≤ fx < 8 MHz | 9 MHz |
| 8 MHz ≤ fx < 9 MHz | 10 MHz |
| 9 MHz ≤ fx ≤ 10 MHz | 8 MHz |

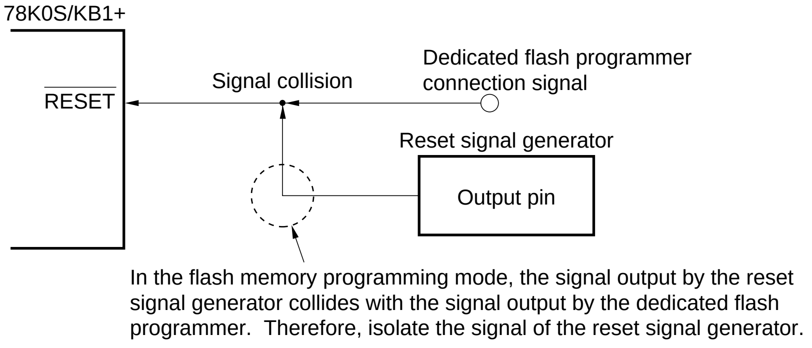
**Caution The above is a recommendation value. A value may change according to the environment to be used.**
**Set up after surely performing sufficient evaluation.**

### 19.6.2 RESET pin

If the reset signal of the dedicated flash programmer is connected to the RESET pin that is connected to the reset signal generator on the board, signal collision takes place. To prevent this collision, isolate the connection with the reset signal generator.

If the reset signal is input from the user system while the flash memory programming mode is set, the flash memory will not be correctly programmed. Do not input any signal other than the reset signal of the dedicated flash programmer.

**Figure 19-7. Signal Collision ($\overline{\text{RESET}}$ Pin)**


78K0S/KB1+
$\overline{\text{RESET}}$
Signal collision
Dedicated flash programmer connection signal
Reset signal generator
Output pin

In the flash memory programming mode, the signal output by the reset signal generator collides with the signal output by the dedicated flash programmer. Therefore, isolate the signal of the reset signal generator.

### 19.6.3 Port pins

When the flash memory programming mode is set, all the pins not used for flash memory programming enter the same status as that immediately after reset. If external devices connected to the ports do not recognize the port status immediately after reset, the port pin must be connected to $V_{DD}$ or $V_{SS}$ via a resistor.

The state of the pins in the self programming mode is the same as that in the HALT mode.

### 19.6.4 Power supply

Connect the $V_{DD}$ pin to $V_{DD}$ of the flash programmer, and the $V_{SS}$ pin to $V_{SS}$ of the flash programmer.

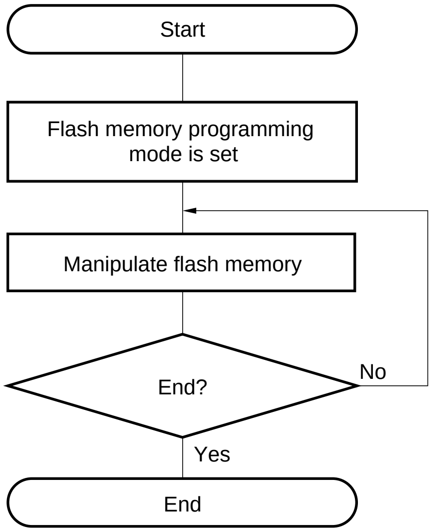
Supply the same other power supplies ($AV_{REF}$ and $AV_{SS}$) as those in the normal operation mode.

## 19.7 On-Board and Off-Board Flash Memory Programming

### 19.7.1 Controlling flash memory

The following figure illustrates the procedure to manipulate the flash memory.

**Figure 19-8. Flash Memory Manipulation Procedure**



### 19.7.2 Flash memory programming mode

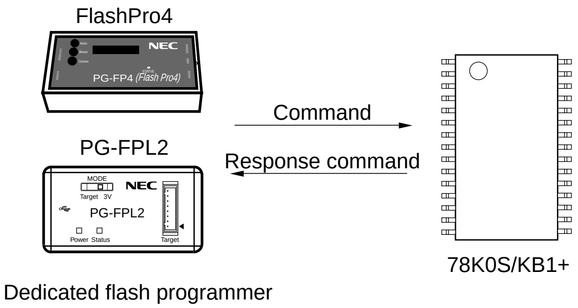
To rewrite the contents of the flash memory by using the dedicated flash programmer, set the 78K0S/KB1+ in the flash memory programming mode. When the 78K0S/KB1+ is connected to the flash programmer and a communication command is transmitted to the microcontroller, the microcontroller is set in the flash memory programming mode.

Change the mode by using a jumper when writing the flash memory on-board.

### 19.7.3 Communication commands

The 78K0S/KB1+ communicates with the dedicated flash programmer by using commands. The signals sent from the flash programmer to the 78K0S/KB1+ are called commands, and the commands sent from the 78K0S/KB1+ to the dedicated flash programmer are called response commands.

**Figure 19-9. Communication Commands**



The flash memory control commands of the 78K0S/KB1+ are listed in the table below. All these commands are issued from the programmer and the 78K0S/KB1+ perform processing corresponding to the respective commands.

**Table 19-6. Flash Memory Control Commands**

| Classification | Command Name | Function |
|---|---|---|
| Erase | Batch 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 to the specified address range and executes a verify check of the contents. |
| Checksum | Checksum command | Reads the checksum of the specified address range and compares with the written data. |
| Blank check | Blank check command | Confirms the erasure status of the entire memory. |
| Security | Security setting command | Prohibits batch erase (chip erase) command, block erase command, and write command to prevent operation by third parties. |

The 78K0S/KB1+ returns a response command for the command issued by the dedicated flash programmer. The response commands sent from the 78K0S/KB1+ are listed below.

**Table 19-7. Response Commands**

| Command Name | Function |
|---|---|
| ACK | Acknowledges command/data. |
| NAK | Acknowledges illegal command/data. |

### 19.7.4 Security settings

The operations shown below can be prohibited using the security setting command.

**Caution The security setting is valid when the programming mode is set next time. Therefore, when the security setting command is executed, exit from the programming mode, then set the programming mode again.**

- Batch erase (chip erase)
  Execution of the block erase and batch erase (chip erase) commands for entire blocks in the flash memory is prohibited. Once execution of the batch erase (chip erase) command is prohibited, all the prohibition settings can no longer be cancelled.

  **Caution After the security setting of the batch erase is set, erasure cannot be performed for the device. In addition, even if a write command is executed, data different from that which has already been written to the flash memory cannot be written because the erase command is disabled.**

- Block erase
  Execution of the block erase command for a specific block in the flash memory is prohibited. This prohibition setting can be cancelled using the batch erase (chip erase) command.

- Write
  Execution of the write and block erase commands for entire blocks in the flash memory is prohibited. This prohibition setting can be cancelled using the batch erase (chip erase) command.

The batch erase (chip erase), block erase, and write commands are enabled by the default setting when the flash memory is shipped. The above security settings are possible only for on-board/off-board programming. Each security setting can be used in combination.

Table 19-8 shows the relationship between the erase and write commands when the 78K0S/KB1+ security function is enabled.

**Table 19-8. Relationship Between Commands When Security Function Is Enabled**

| Command / Security | Batch Erase (Chip Erase) Command | Block Erase Command | Write Command |
|---|---|---|---|
| When batch erase (chip erase) security operation is enabled | Disabled | Disabled | Enabled[Note] |
| When block erase security operation is enabled | Enabled | | Enabled |
| When write security operation is enabled | | | Disabled |

**Note** Since the erase command is disabled, data different from that which has already been written to the flash memory cannot be written.

Table 19-9 shows the relationship between the security setting and the operation in each programming mode.

**Table 19-9. Relationship Between Security Setting and Operation In Each Programming Mode**

| Programming Mode<br>Security Setting | On-Board/Off-Board Programming | | Self Programming | |
|---|---|---|---|---|
| | Security Setting | Security Operation | Security Setting | Security Operation |
| Batch erase (chip erase) | Possible | Valid[Note 1] | Impossible | Invalid[Note 2] |
| Block erase | | | | |
| Write | | | | |

**Notes 1.** Execution of each command is prohibited by the security setting.

**2.** Execution of self programming command is possible regardless of the security setting.

## 19.8 Flash Memory Programming by Self Writing

The 78K0S/KB1+ supports a self programming function that can be used to rewrite the flash memory via a user program, making it possible to upgrade programs in the field.

**Caution Self programming processing must be included in the program before performing self writing.**

**Remark** To use the internal flash memory of the 78K0S/KB1+ as the external EEPROM for storing data, refer to “78K0S/Kx1+ EEPROM Emulation AN” (U17379E).

### 19.8.1 Outline of self programming

To execute self programming, shift the mode from the normal operation of the user program (normal mode) to the self programming mode. Write/erase processing for the flash memory, which has been set to the register in advance, is performed by executing the HALT instruction during self programming mode. The HALT state is automatically released when processing is completed.

To shift to the self programming mode, execute a specific sequence for a specific register. Refer to **19.8.4 Example of shifting normal mode to self programming** for details.

**Remark** Data written by self programming can be referenced with the MOV instruction.

**Table 19-10. Self Programming Mode**

| Mode | User Program Execution | Execution of Write/erase for Flash Memory with HALT Instruction |
|---|---|---|
| Normal mode | Enabled | – |
| Self programming mode | Enabled[Note] | Enabled |

**Note** Maskable interrupt servicing is disabled during self programming mode.

Figure 19-10 shows a block diagram for self programming, Figure 19-11 shows the self programming state transition diagram, Table 19-11 lists the commands for controlling self programming.

### Figure 19-10. Block Diagram of Self Programming

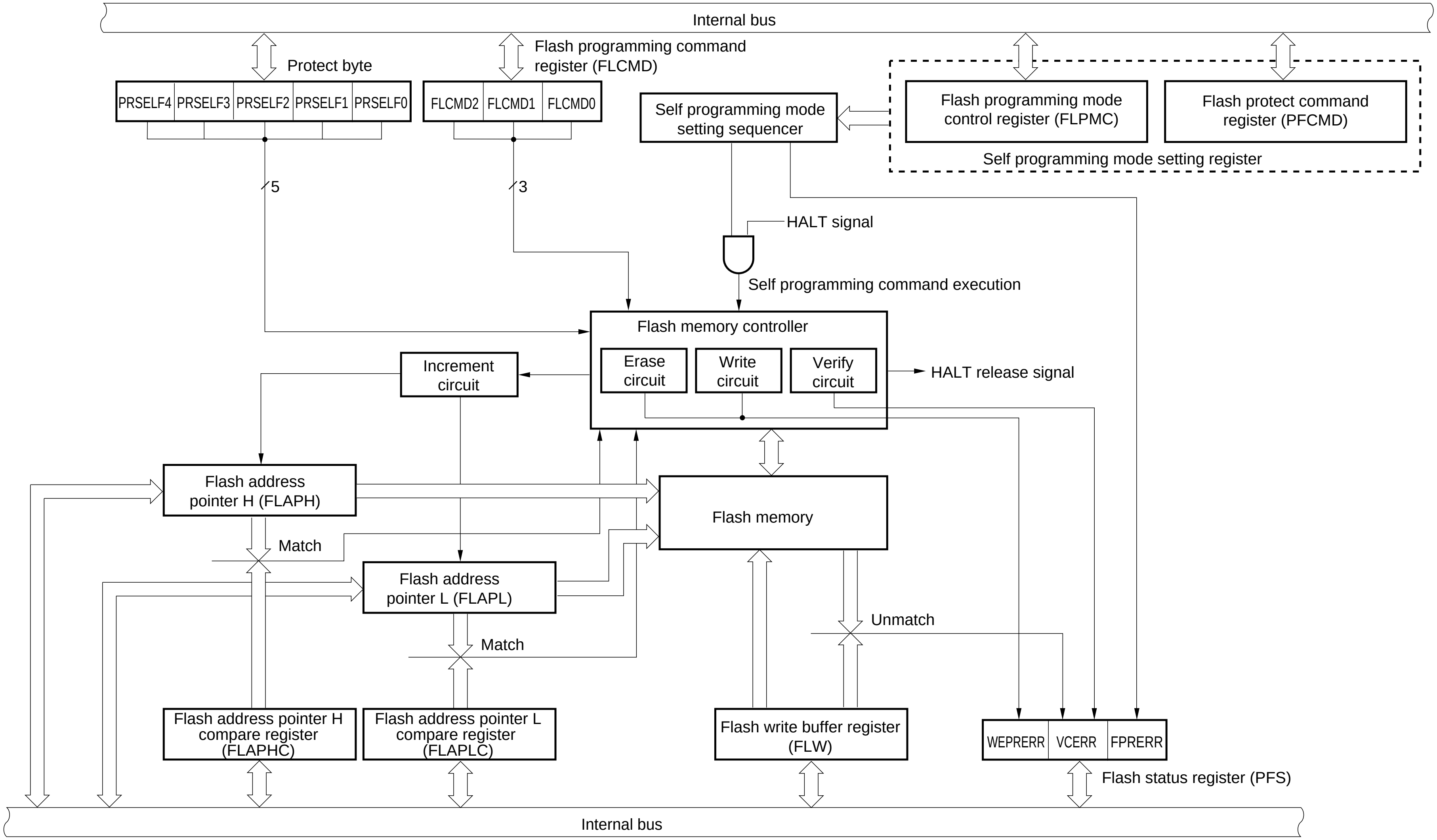

Internal bus
Protect byte
PRSELF4
PRSELF3
PRSELF2
PRSELF1
PRSELF0
Flash programming command register (FLCMD)
FLCMD2
FLCMD1
FLCMD0
Self programming mode setting sequencer
Flash programming mode control register (FLPMC)
Flash protect command register (PFCMD)
Self programming mode setting register
5
3
HALT signal
Self programming command execution
Flash memory controller
Erase circuit
Write circuit
Verify circuit
HALT release signal
Increment circuit
Flash address pointer H (FLAPH)
Flash memory
Match
Flash address pointer L (FLAPL)
Match
Unmatch
Flash address pointer H compare register (FLAPHC)
Flash address pointer L compare register (FLAPLC)
Flash write buffer register (FLW)
WEPRERR
VCERR
FPRERR
Flash status register (PFS)
Internal bus

Figure 19-11. Self Programming State Transition Diagram

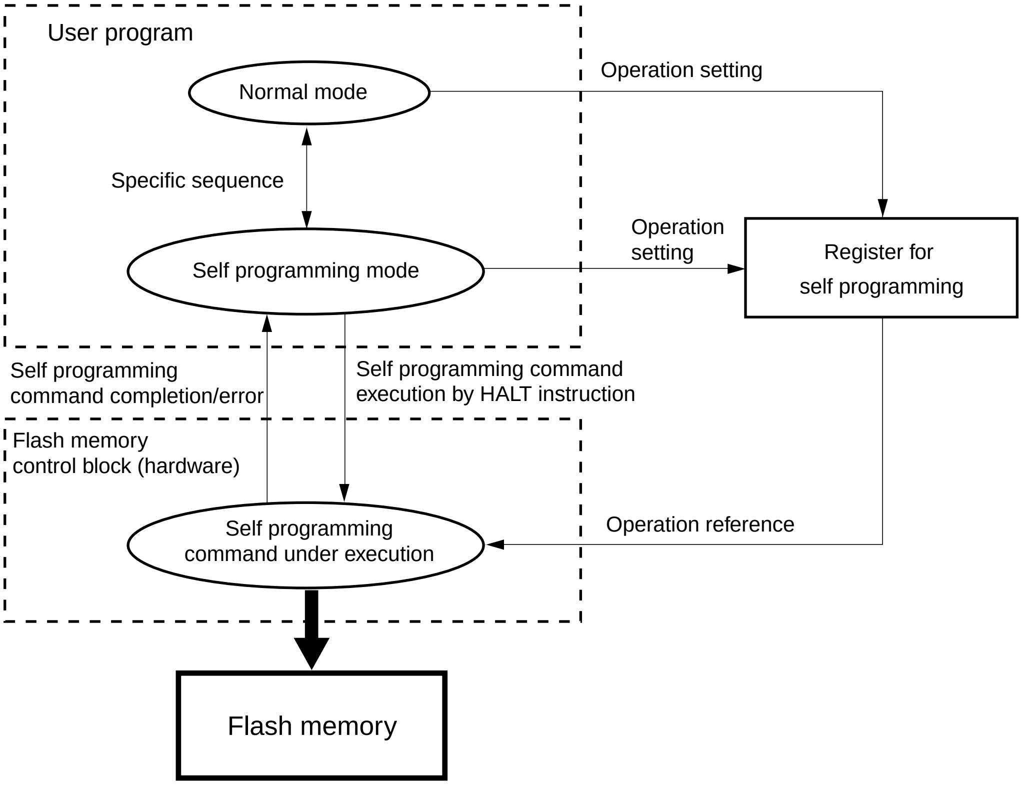


Table 19-11. Self Programming Controlling Commands

| Command Name | Function | Time Taken from HALT Instruction Execution to Command Execution End |
|---|---|---|
| Internal verify | This command is used to check if data has been correctly written to the flash memory. After data has been written to the memory, specify the block number, the start address, and the end address, then execute this command. | Internal verify for 1 block (internal verify command executed once): 6.8 ms<br>Internal verify for 1 byte: 27 μs |
| Block erasure | This command is used to erase a specified block. Specify the block number before execution. | 8.5 ms |
| Block blank check | This command is used to check if data in a specified block has been erased. Specify the block number, then execute this command. | 480 μs |
| Byte write | This command is used to write 1-byte data to the specified address in the flash memory. Specify the write address and write data, then execute this command. | 150 μs |

### 19.8.2 Cautions on self programming function

- If an interrupt occurs during self programming, the interrupt request flag is set (1), and interrupt servicing is performed after the self programming mode is released. To avoid this operation, disable interrupt servicing (by setting MK0 and MK1 to FFH, and executing the DI instruction) during self programming or before a mode is shifted from the normal mode to the self programming mode with a specific sequence.
- No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 19-11 for the time taken for the execution of self programming.
- RAM is not used while a self programming command is being executed.
- If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed.
- The value of the blank data set during block erasure is FFH.
- When the oscillator or the external clock is selected as the main clock, a wait time of 16 $\mu$s is required starting from the setting of the self programming mode to the execution of the HALT instruction.
- The state of the pins in self programming mode is the same as that in HALT mode.
- Since the security function set via on-board/off-board programming is disabled in self programming mode, the self programming command can be executed regardless of the security function setting. To disable write or erase processing during self programming, set the protect byte.
- Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.

### 19.8.3 Registers used for self programming function

The following registers are used for the self programming function.

- Flash programming mode control register (FLPMC)
- Flash protect command register (PFCMD)
- Flash status register (PFS)
- Flash programming command register (FLCMD)
- Flash address pointers H and L (FLAPH and FLAPL)
- Flash address pointer H compare register and flash address pointer L compare register (FLAPHC and FLAPLC)
- Flash write buffer register (FLW)

The 78K0S/KB1+ has an area called a protect byte at address 0081H of the flash memory.

#### (1) Flash programming mode control register (FLPMC)

This register is used to set the operation mode when data is written to the flash memory in the self programming mode, and to read the set value of the protect byte.

Data can be written to FLPMC only in a specific sequence (refer to **19.8.3 (2) Flash protect command register (PFCMD)**) so that the application system does not stop by accident because of malfunctions due to noise or program hang-ups.

This register is set with an 8-bit memory manipulation instruction.

Reset signal generation makes the contents of this register undefined.

**Figure 19-12. Format of Flash Programming Mode Control Register (FLPMC)**

Address: FFA2H  After reset: Undefined[Note 1]  R/W[Note 2]

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FLPMC | 0 | PRSELF4 | PRSELF3 | PRSELF2 | PRSELF1 | PRSELF0 | 0 | FLSPM |

| FLSPM | Selection of operation mode during self programming mode |
|---|---|
| 0 | Normal mode<br>Flash memory instructions can be fetched from all addresses. |
| 1 | Self programming mode<br>Before executing the HALT instruction, set the command, address offset, write data, and set FLSPM to 1. After setting these items, execute the HALT instruction; the flash memory mode is then shifted from the normal mode to the flash memory programming mode. |

| | | | | | |
|---|---|---|---|---|---|
| PRSELF4 | PRSELF3 | PRSELF2 | PRSELF1 | PRSELF0 | The set value of the protect byte is read to these bits. |

**Notes 1.** Bit 0 (FLSPM) is cleared to 0 when reset is released. The set value of the protect byte is read to bits 2 to 6 (PRSELF0 to PRSELF4) after reset is released.

**2.** Bits 2 to 6 (PRSELF0 to PRSELF4) are read-only.

**Cautions 1. Note the following when setting the self programming mode.**

- **If an interrupt occurs during self programming, the interrupt request flag is set (1), and interrupt servicing is performed after the self programming mode is released. To avoid this operation, disable interrupt servicing (by setting MK0 and MK1 to FFH, and executing the DI instruction) during self programming or before a mode is shifted from the normal mode to the self programming mode with a specific sequence.**
- **No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 19-11 for the time taken for the execution of self programming.**
- **If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed.**

**2. When the oscillator or the external clock is selected as the main clock, a wait time of 16 μs is required from setting FLSPM to 1 to execution of the HALT instruction.**

### (2) Flash protect command register (PFCMD)

If the application system stops inadvertently due to malfunction caused by noise or program hang-up, an operation to write the flash programming mode control register (FLPMC) may have a serious effect on the system. PFCMD is used to protect FLPMC from being written, so that the application system does not stop inadvertently.

Writing FLPMC is enabled only when a write operation is performed in the following specific sequence.

<1> Write a specific value to PFCMD (A5H)
<2> Write the value to be set to FLPMC (writing in this step is invalid)
<3> Write the inverted value of the value to be set to FLPMC (writing in this step is invalid)
<4> Write the value to be set to FLPMC (writing in this step is valid)

This rewrites the value of the register, so that the register cannot be written illegally.

Occurrence of an illegal store operation can be checked by bit 0 (FPRERR) of the flash status register (PFS).

A5H must be written to PFCMD each time the value of FLPMC is changed.

PFCMD can be set with an 8-bit memory manipulation instruction.

Reset signal generation makes PFCMD undefined.

**Figure 19-13. Format of Flash Protect Command Register (PFCMD)**

Address: FFA0H After reset: Undefined W

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

**Caution Disable interrupt servicing (by setting MK0 and MK1 to FFH and executing the DI instruction) while the specific sequence is under execution.**

### (3) Flash status register (PFS)

If data is not written to the flash programming mode control register (FLPMC), which is protected, in the correct sequence (writing the flash protect command register (PFCMD)), FLPMC is not written and a protection error occurs. If this happens, bit 0 of PFS (FPRERR) is set to 1.

When FPRERR is 1, it can be cleared to 0 by writing 0 to it.

Errors that may occur during self programming are reflected in bit 1 (VCERR) and bit 2 (WEPRERR) of PFS. VCERR or WEPRERR can be cleared by writing 0 to them.

All the flags of the PFS register must be pre-cleared to 0 to check if the operation is performed correctly.

PFS can be set with a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears PFS to 00H.

**Figure 19-14. Format of Flash Status Register (PFS)**

Address: FFA1H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFS | 0 | 0 | 0 | 0 | 0 | WEPRERR | VCERR | FPRERR |

1. Operating conditions of FPRERR flag

<Setting conditions>

- If PFCMD is written when the store instruction operation recently performed on a peripheral register is not to write a specific value (A5H) to PFCMD
- If the first store instruction operation after <1> is on a peripheral register other than FLPMC
- If the first store instruction operation after <2> is on a peripheral register other than FLPMC
- If a value other than the inverted value of the value to be set to FLPMC is written by the first store instruction after <2>
- If the first store instruction operation after <3> is on a peripheral register other than FLPMC
- If a value other than the value to be set to FLPMC (value written in <2>) is written by the first store instruction after <3>

**Remark** The numbers in angle brackets above correspond to the those in **(2) Flash protect command register (PFCMD)**.

<Reset conditions>

- If 0 is written to the FPRERR flag
- If the reset signal is generated

2. Operating conditions of VCERR flag

<Setting conditions>

- Erasure verification error
- Internal writing verification error

If VCERR is set, it means that the flash memory has not been erased or written correctly. Erase or write the memory again in the specified procedure.

**Remark** The VCERR flag may also be set if an erase or write protect error occurs.

<Reset conditions>

- When 0 is written to the VCERR flag
- When the reset signal is generated

3. Operating conditions of WEPRERR flag

<Setting conditions>

- If the area specified by the protect byte to be protected from erasing or writing is specified by the flash address pointer H (FLAPH) and a command is executed to this area

<Reset conditions>

- When 0 is written to the WEPRERR flag
- When the reset signal is generated

### (4) Flash programming command register (FLCMD)

This register is used to specify whether the flash memory is erased, written, or verified in the self programming mode.

This register is set with a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Figure 19-15. Format of Flash Programming Command Register (FLCMD)**

Address: FFA3H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FLCMD | 0 | 0 | 0 | 0 | 0 | FLCMD2 | FLCMD1 | FLCMD0 |

| FLCMD2 | FLCMD1 | FLCMD0 | Command Name | Function |
|---|---|---|---|---|
| 0 | 0 | 1 | Internal verify | This command is used to check if data has been correctly written to the flash memory. After data has been written to the memory, execute this command by specifying a block number, start address, and end address. If an error occurs, bit 1 (VCERR) or bit 2 (WEPRERR) of the flash status register (PFS) is set to 1. |
| 0 | 1 | 1 | Block erase | This command is used to erase specified block. It is used both in the on-board mode and self programming mode. |
| 1 | 0 | 0 | Block blank check | This command is used to check if the specified block has been erased. |
| 1 | 0 | 1 | Byte write | This command is used to write 1-byte data to the specified address in the flash memory. Specify the write address and write data, then execute this command. |
| Other than above[Note] | | | Setting prohibited | |

**Note** If a value other than the above is set and the self programming mode is set, the self programming mode is canceled immediately and no execution occurs. At this time, the flag of the PFS register is not set.

### (5) Flash address pointers H and L (FLAPH and FLAPL)

These registers are used to specify the start address of the flash memory when the memory is erased, written, or verified in the self programming mode.

FLAPH and FLAPL consist of counters, and they are incremented until the values match with those of FLAPHC and FLAPLC when the programming command is not executed. When the programming command is executed, therefore, set the value again.

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

Reset signal generation makes these registers undefined.

**Figure 19-16. Format of Flash Address Pointer H/L (FLAPH/FLAPL)**

Address: FFA4H, FFA5H After reset: Undefined R/W

| FLAPH (FFA5H) | | | | | | | | FLAPL (FFA4H) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | FLAP11 | FLAP10 | FLAP9 | FLAP8 | FLAP7 | FLAP6 | FLAP5 | FLAP4 | FLAP3 | FLAP2 | FLAP1 | FLAP0 |

**Caution Be sure to clear bits 4 to 7 of FLAPH and FLAPHC to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.**

### (6) Flash address pointer H compare register and flash address pointer L compare register (FLAPHC and FLAPLC)

These registers are used to specify the address range in which the internal sequencer operates when the flash memory is verified in the self programming mode.

Set FLAPHC to the same value as that of FLAPH. Set the last address of the range in which verification is to be executed to FLAPLC.

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

Reset signal generation clears these registers to 00H.

**Figure 19-17. Format of Flash Address Pointer H/L Compare Registers (FLAPHC/FLAPLC)**

Address: FFA6H, FFA7H After reset: 00H R/W

| FLAPHC (FFA7H) | | | | | | | | FLAPLC (FFA6H) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | FLAPC11 | FLAPC10 | FLAPC9 | FLAPC8 | FLAPC7 | FLAPC6 | FLAPC5 | FLAPC4 | FLAPC3 | FLAPC2 | FLAPC1 | FLAPC0 |

**Cautions 1. Be sure to clear bits 4 to 7 of FLAPH and FLAPHC to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.**

**2. Set the number of the block subject to a block erase, write, verify, or blank check (same value as FLAPH) to FLAPHC.**

**3. Clear FLAPLC to 00H when a block erase is performed, and FFH when a blank check is performed.**

### (7) Flash write buffer register (FLW)

This register is used to store the data to be written to the flash memory.

This register is set with a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears these registers to 00H.

**Figure 19-18. Format of Flash Write Buffer Register (FLW)**

Address: FFA8H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FLW | FLW7 | FLW6 | FLW5 | FLW4 | FLW3 | FLW2 | FLW1 | FLW0 |

### (8) Protect byte

This protect byte is used to specify the area that is to be protected from writing or erasing. The specified area is valid only in the self programming mode. Because self programming of the protected area is invalid, the data written to the protected area is guaranteed.

**Figure 19-19. Format of Protect Byte (1/2)**

Address: 0081H

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | PRSELF4 | PRSELF3 | PRSELF2 | PRSELF1 | PRSELF0 | 1 | 1 |

- μPD78F9232

| PRSELF4 | PRSELF3 | PRSELF2 | PRSELF1 | PRSELF0 | Status |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | Blocks 15 to 0 are protected. |
| 0 | 1 | 0 | 0 | 1 | Blocks 13 to 0 are protected.<br>Blocks 14 and 15 can be written or erased. |
| 0 | 1 | 0 | 1 | 0 | Blocks 11 to 0 are protected.<br>Blocks 12 to 15 can be written or erased. |
| 0 | 1 | 0 | 1 | 1 | Blocks 9 to 0 are protected.<br>Blocks 10 to 15 can be written or erased. |
| 0 | 1 | 1 | 0 | 0 | Blocks 7 to 0 are protected.<br>Blocks 8 to 15 can be written or erased. |
| 0 | 1 | 1 | 0 | 1 | Blocks 5 to 0 are protected.<br>Blocks 6 to 15 can be written or erased. |
| 0 | 1 | 1 | 1 | 0 | Blocks 3 to 0 are protected.<br>Blocks 4 to 15 can be written or erased. |
| 0 | 1 | 1 | 1 | 1 | Blocks 1 and 0 are protected.<br>Blocks 2 to 15 can be written or erased. |
| 1 | 1 | 1 | 1 | 1 | All blocks can be written or erased. |
| Other than above | | | | | Setting prohibited |

Figure 19-19. Format of Protect Byte (2/2)

- μPD78F9234

| PRSELF4 | PRSELF3 | PRSELF2 | PRSELF1 | PRSELF0 | Status |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Blocks 31 to 0 are protected. |
| 0 | 0 | 0 | 0 | 1 | Blocks 29 to 0 are protected.<br>Blocks 30 and 31 can be written or erased. |
| 0 | 0 | 0 | 1 | 0 | Blocks 27 to 0 are protected.<br>Blocks 28 to 31 can be written or erased. |
| 0 | 0 | 0 | 1 | 1 | Blocks 25 to 0 are protected.<br>Blocks 26 and 31 can be written or erased. |
| 0 | 0 | 1 | 0 | 0 | Blocks 23 to 0 are protected.<br>Blocks 24 to 31 can be written or erased. |
| 0 | 0 | 1 | 0 | 1 | Blocks 21 to 0 are protected.<br>Blocks 22 to 31 can be written or erased. |
| 0 | 0 | 1 | 1 | 0 | Blocks 19 to 0 are protected.<br>Blocks 20 to 31 can be written or erased. |
| 0 | 0 | 1 | 1 | 1 | Blocks 17 to 0 are protected.<br>Blocks 18 to 31 can be written or erased. |
| 0 | 1 | 0 | 0 | 0 | Blocks 15 to 0 are protected.<br>Blocks 16 to 31 can be written or erased. |
| 0 | 1 | 0 | 0 | 1 | Blocks 13 to 0 are protected.<br>Blocks 14 to 31 can be written or erased. |
| 0 | 1 | 0 | 1 | 0 | Blocks 11 to 0 are protected.<br>Blocks 12 to 31 can be written or erased. |
| 0 | 1 | 0 | 1 | 1 | Blocks 9 to 0 are protected.<br>Blocks 10 to 31 can be written or erased. |
| 0 | 1 | 1 | 0 | 0 | Blocks 7 to 0 are protected.<br>Blocks 8 to 31 can be written or erased. |
| 0 | 1 | 1 | 0 | 1 | Blocks 5 to 0 are protected.<br>Blocks 6 to 31 can be written or erased. |
| 0 | 1 | 1 | 1 | 0 | Blocks 3 to 0 are protected.<br>Blocks 4 to 31 can be written or erased. |
| 0 | 1 | 1 | 1 | 1 | Blocks 1 and 0 are protected.<br>Blocks 2 to 31 can be written or erased. |
| 1 | 1 | 1 | 1 | 1 | All blocks can be written or erased. |
| Other than above | | | | | Setting prohibited |

### 19.8.4 Example of shifting normal mode to self programming mode

The operating mode must be shifted from normal mode to self programming mode before performing self programming.

An example of shifting to self programming mode is explained below.

<1> Disable interrupts if the interrupt function is used (by setting the interrupt mask flag registers (MK0, MK1) to FFH and executing the DI instruction).

<2> Clear the flash status register (PFS).

<3> Set self programming mode using a specific sequence.

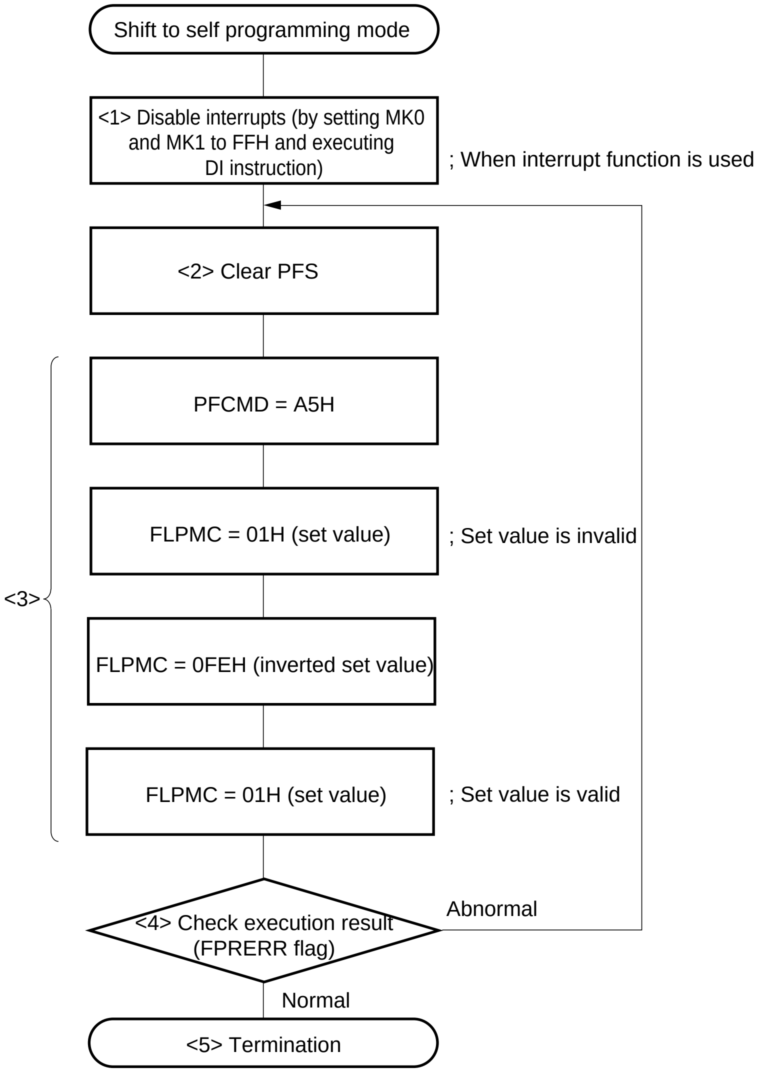
- Write a specific value (A5H) to PFCMD.
- Write 01H to FLPMC (writing in this step is invalid).
- Write 0FEH (inverted value of 01H) to FLPMC (writing in this step is invalid).
- Write 01H to FLPMC (writing in this step is valid).

<4> Check the execution result of the specific sequence using bit 0 (FPRERR) of PFS.
Abnormal → <2>, normal → <5>

<5> Mode shift is completed.

**Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

**Figure 19-20. Example of Shifting to Self Programming Mode**


Shift to self programming mode
<1> Disable interrupts (by setting MK0 and MK1 to FFH and executing DI instruction)
; When interrupt function is used
<2> Clear PFS
<3>
PFCMD = A5H
FLPMC = 01H (set value)
; Set value is invalid
FLPMC = 0FEH (inverted set value)
FLPMC = 01H (set value)
; Set value is valid
<4> Check execution result (FPRERR flag)
Abnormal
Normal
<5> Termination

**Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

**Remark** <1> to <5> in Figure 19-20 correspond to <1> to <5> in **19.8.4** (previous page).

An example of the program list that shifts the mode to self programming mode is shown below.

```
;----------------------------
;START
;----------------------------
       MOV      MK0,#11111111B     ; Masks all interrupts
       MOV      MK1,#11111111B

       DI

ModeOnLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H        ; PFCMD register control
       MOV      FLPMC,#01H         ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH        ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H         ; Sets self programming mode with FLPMC register
                                   ; control (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOnLoop        ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs.

;----------------------------
;END
;----------------------------
```

### 19.8.5 Example of shifting self programming mode to normal mode

The operating mode must be returned from self programming mode to normal mode after performing self programming.

An example of shifting to normal mode is explained below.

<1> Clear the flash status register (PFS).

<2> Set normal mode using a specific sequence.
- Write the specific value (A5H) to PFCMD.
- Write 00H to FLPMC (writing in this step is invalid)
- Write 0FFH (inverted value of 00H) to FLPMC (writing in this step is invalid)
- Write 00H to FLPMC (writing in this step is valid)

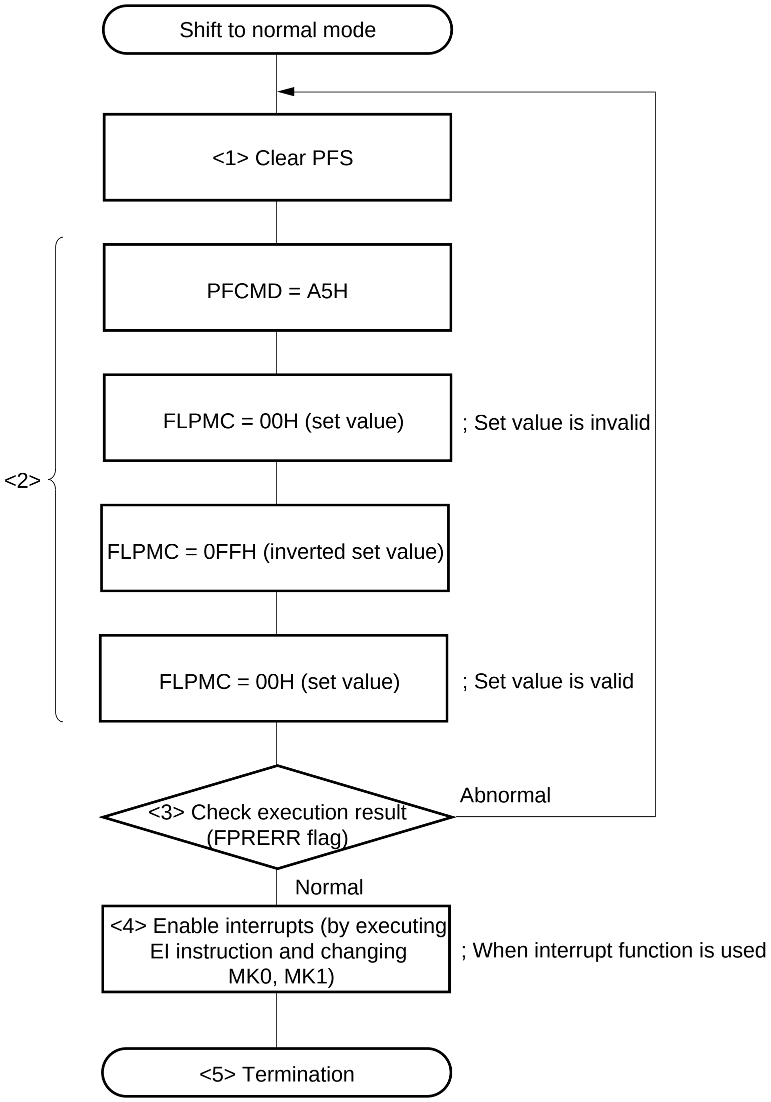
<3> Check the execution result of the specific sequence using bit 0 (FPRERR) of PFS.
Abnormal → <1>, normal → <4>

<4> Enable interrupt servicing (by executing the EI instruction and changing MK0 and MK1) to restore the original state.

<5> Mode shift is completed

**Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

Figure 19-21. Example of Shifting to Normal Mode


Shift to normal mode
<1> Clear PFS
PFCMD = A5H
FLPMC = 00H (set value)
; Set value is invalid
<2>
FLPMC = 0FFH (inverted set value)
FLPMC = 00H (set value)
; Set value is valid
<3> Check execution result (FPRERR flag)
Abnormal
Normal
<4> Enable interrupts (by executing EI instruction and changing MK0, MK1)
; When interrupt function is used
<5> Termination

**Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

**Remark** <1> to <5> in Figure 19-21 correspond to <1> to <5> in **19.8.5** (previous page).

An example of a program list that shifts the mode to normal mode is shown below.

```
;----------------------------
;START
;----------------------------

ModeOffLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H       ; PFCMD register control
       MOV      FLPMC,#00H        ; FLPMC register control (sets value)
       MOV      FLPMC,#0FFH       ; FLPMC register control (inverts set value)
       MOV      FLPMC,#00H        ; Sets normal mode via FLPMC register control (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOffLoop      ; Checks completion of write to specific registers
                                  ; Repeats the same processing when an error occurs

       MOV      MK0,#INT_MK0      ; Restores interrupt mask flag
       MOV      MK1,#INT_MK1

       EI

;----------------------------
;END
;----------------------------
```

### 19.8.6 Example of block erase operation in self programming mode

An example of the block erase operation in self programming mode is explained below.

<1> Set 03H (block erase) to the flash program command register (FLCMD).

<2> Set the block number to be erased, to flash address pointer H (FLAPH).

<3> Set flash address pointer L (FLAPL) to 00H.

<4> Write the same value as FLAPH to the flash address pointer H compare register (FLAPHC).

<5> Set the flash address pointer L compare register (FLAPLC) to 00H.

<6> Clear the flash status register (PFS).

<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].

<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)

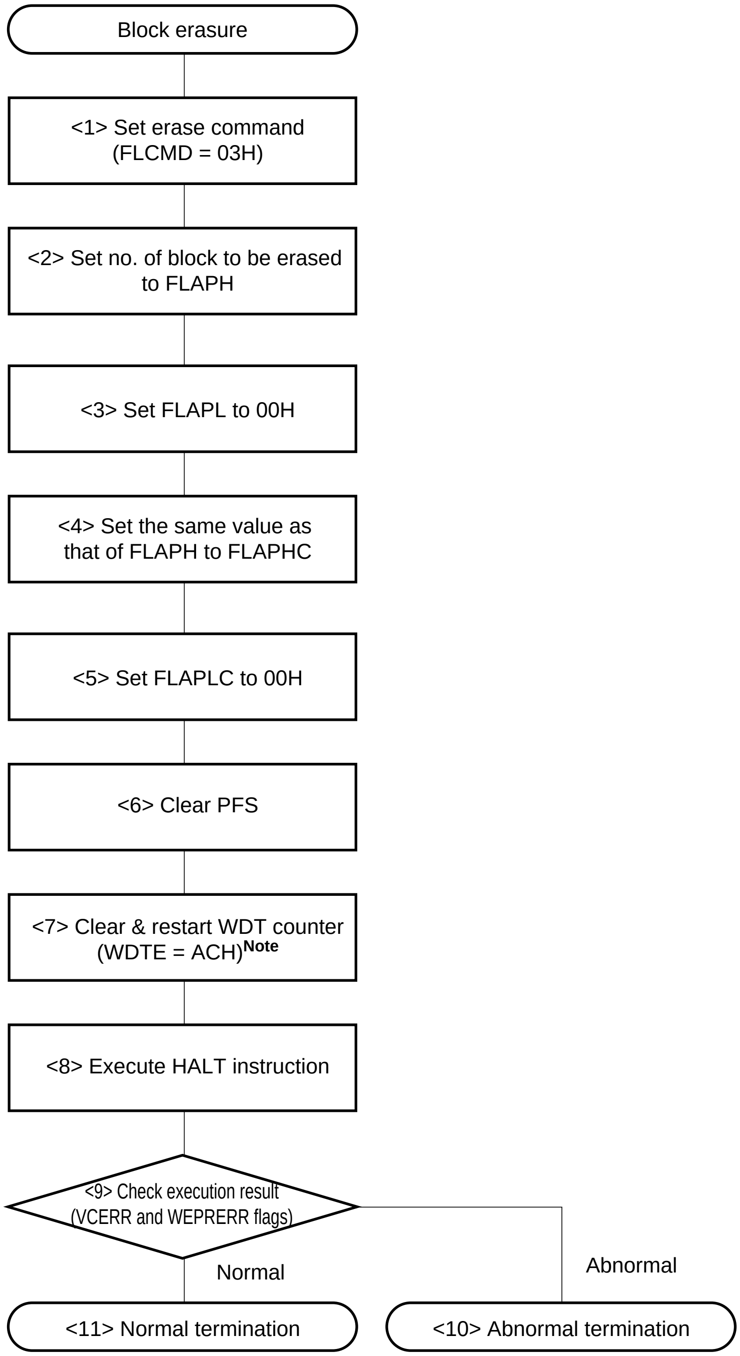
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>

<10> Block erase processing is abnormally terminated.

<11> Block erase processing is normally terminated.

**Note** This setting is not required when the watchdog timer is not used.

**Figure 19-22. Example of Block Erase Operation in Self Programming Mode**


Block erasure
<1> Set erase command (FLCMD = 03H)
<2> Set no. of block to be erased to FLAPH
<3> Set FLAPL to 00H
<4> Set the same value as that of FLAPH to FLAPHC
<5> Set FLAPLC to 00H
<6> Clear PFS
<7> Clear & restart WDT counter (WDTE = ACH)Note
<8> Execute HALT instruction
<9> Check execution result (VCERR and WEPRERR flags)
Normal
Abnormal
<11> Normal termination
<10> Abnormal termination

**Note** This setting is not required when the watchdog timer is not used.

**Remark** <1> to <11> in Figure 19-22 correspond to <1> to <11> in **19.8.6** (previous page).

An example of a program list that performs a block erase in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------

FlashBlockErase:
       MOV      FLCMD,#03H     ; Sets flash control command (block erase)
       MOV      FLAPH,#07H     ; Sets number of block to be erased (block 7 is specified here)
       MOV      FLAPL,#00H     ; Fixes FLAPL to “00H”
       MOV      FLAPHC,#07H    ; Sets erase block compare number (same value as that of FLAPH)
       MOV      FLAPLC,#00H    ; Fixes FLAPLC to “00H”

       MOV      PFS,#00H       ; Clears flash status register
       MOV      WDTE,#0ACH     ; Clears & restarts WDT
       HALT                    ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A    ; Execution result is stored in variable
                               ; (CmdStatus = 0: normal termination, other than 0: abnormal
                               ; termination)

;----------------------------
;END
;----------------------------
```

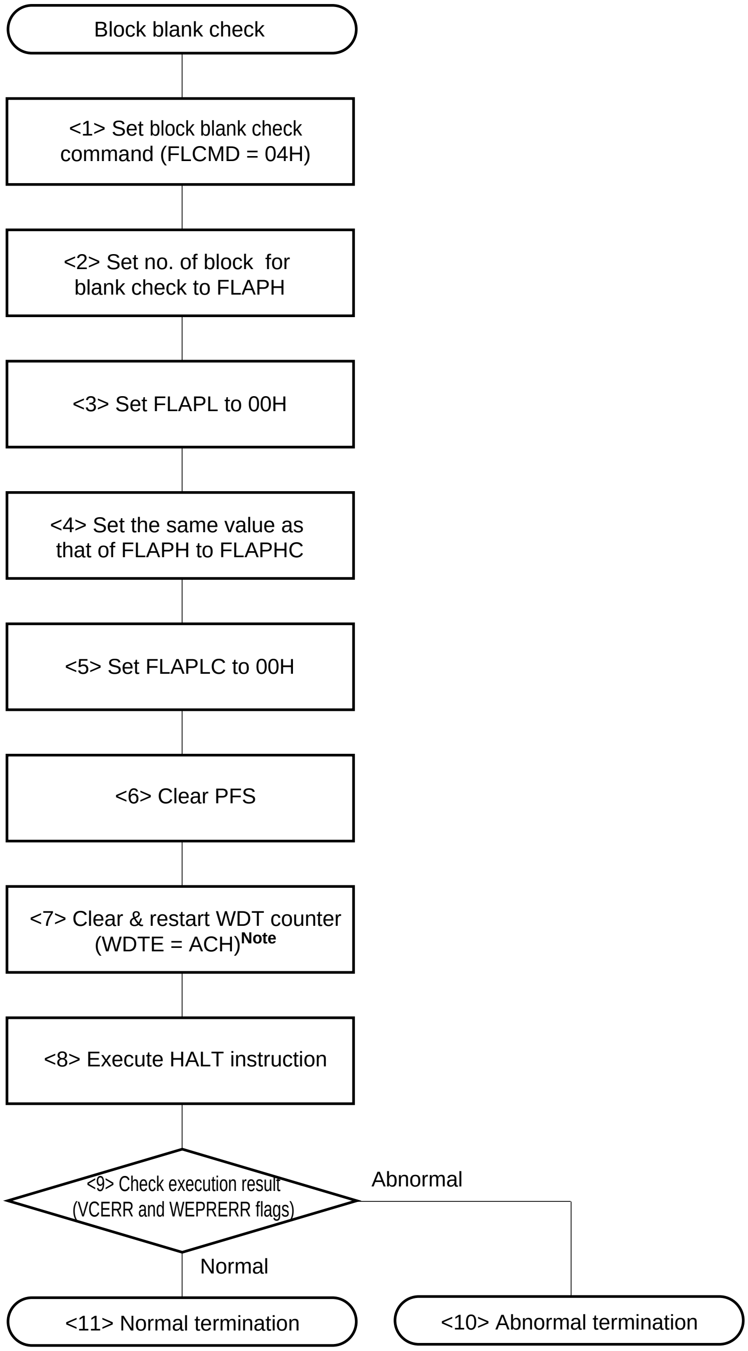
### 19.8.7 Example of block blank check operation in self programming mode

An example of the block blank check operation in self programming mode is explained below.

<1> Set 04H (block blank check) to the flash program command register (FLCMD).
<2> Set the number of block for which a blank check is performed, to flash address pointer H (FLAPH).
<3> Set flash address pointer L (FLAPL) to 00H.
<4> Write the same value as FLAPH to the flash address pointer H compare register (FLAPHC).
<5> Set the flash address pointer L compare register (FLAPLC) to FFH.
<6> Clear the flash status register (PFS).
<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>
<10> Block blank check is abnormally terminated.
<11> Block blank check is normally terminated.

**Note** This setting is not required when the watchdog timer is not used.

**Figure 19-23. Example of Block Blank Check Operation in Self Programming Mode**


Block blank check
<1> Set block blank check command (FLCMD = 04H)
<2> Set no. of block for blank check to FLAPH
<3> Set FLAPL to 00H
<4> Set the same value as that of FLAPH to FLAPHC
<5> Set FLAPLC to 00H
<6> Clear PFS
<7> Clear & restart WDT counter (WDTE = ACH)Note
<8> Execute HALT instruction
<9> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
<11> Normal termination
<10> Abnormal termination

**Note** This setting is not required when the watchdog timer is not used.

**Remark** <1> to <11>in Figure 19-23 correspond to <1> to <11> in **19.8.7** (previous page).

An example of a program list that performs a block blank check in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------

FlashBlockBlankCheck:
       MOV      FLCMD,#04H      ; Sets flash control command (block blank check)
       MOV      FLAPH,#07H      ; Sets number of block for blank check (block 7 is specified
                                ; here)
       MOV      FLAPL,#00H      ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H     ; Sets blank check block compare number (same value as that of
                                ; FLAPH)
       MOV      FLAPLC,#0FFH    ; Fixes FLAPLC to "FFH"

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

### 19.8.8 Example of byte write operation in self programming mode

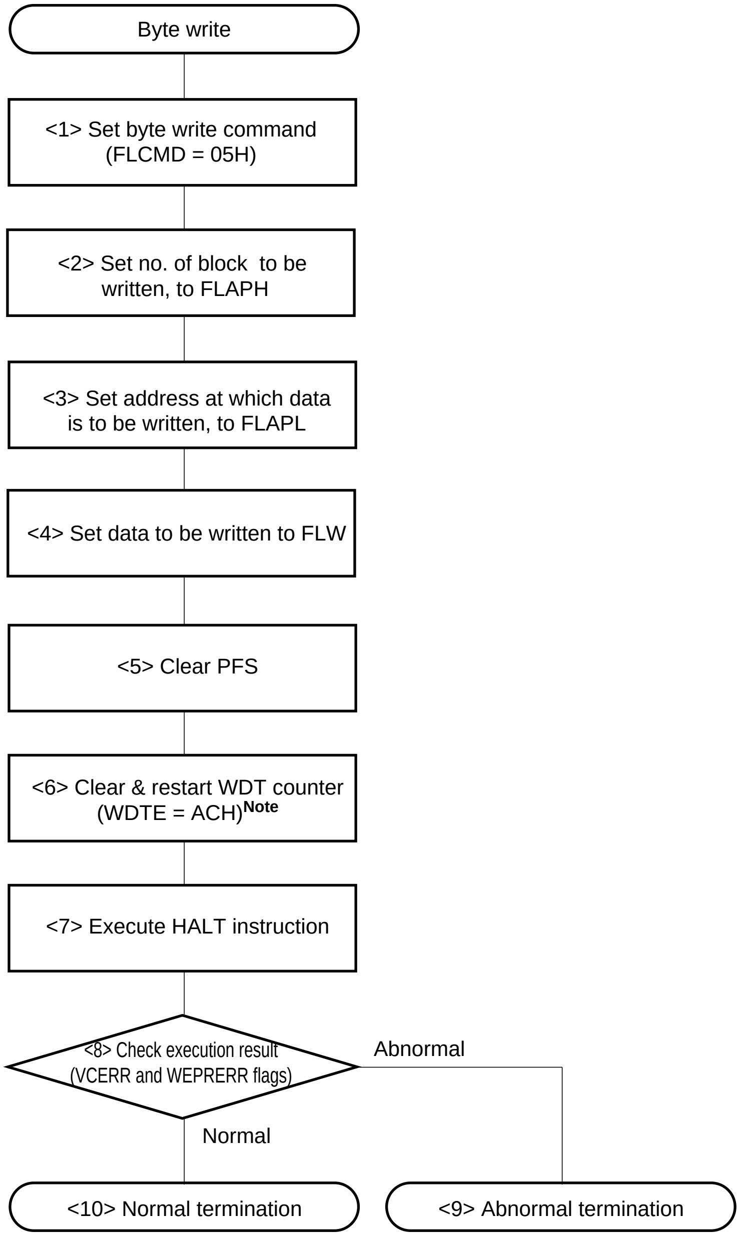
An example of the byte write operation in self programming mode is explained below.

<1> Set 05H (byte write) to the flash program command register (FLCMD).
<2> Set the number of block to which data is to be written, to flash address pointer H (FLAPH).
<3> Set the address at which data is to be written, to flash address pointer L (FLAPL).
<4> Set the data to be written, to the flash write buffer register (FLW).
<5> Clear the flash status register (PFS).
<6> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<7> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<8> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <9>
Normal → <10>
<9> Byte write processing is abnormally terminated.
<10> Byte write processing is normally terminated.

**Note** This setting is not required when the watchdog timer is not used.

**Caution If a write results in failure, erase the block once and write to it again.**

**Figure 19-24. Example of Byte Write Operation in Self Programming Mode**


Byte write
<1> Set byte write command (FLCMD = 05H)
<2> Set no. of block to be written, to FLAPH
<3> Set address at which data is to be written, to FLAPL
<4> Set data to be written to FLW
<5> Clear PFS
<6> Clear & restart WDT counter (WDTE = ACH)Note
<7> Execute HALT instruction
<8> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
<10> Normal termination
<9> Abnormal termination

**Note** This setting is not required when the watchdog timer is not used.

**Remark** <1> to <10> in Figure 19-24 correspond to <1> to <10> in **19.8.8** (previous page).

An example of a program list that performs a byte write in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------
FlashWrite:
       MOV      FLCMD,#05H      ; Sets flash control command (byte write)
       MOV      FLAPH,#07H      ; Sets address to which data is to be written, with
                                ; FLAPH (block 7 is specified here)
       MOV      FLAPL,#20H      ; Sets address to which data is to be written, with
                                ; FLAPL (address 20H is specified here)
       MOV      FLW,#10H        ; Sets data to be written (10H is specified here)

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

### 19.8.9 Example of internal verify operation in self programming mode

An example of the internal verify operation in self programming mode is explained below.

<1> Set 01H (internal verify) to the flash program command register (FLCMD).

<2> Set the number of block for which internal verify is performed, to flash address pointer H (FLAPH).

<3> Sets the verify start address to flash address pointer L (FLAPL).

<4> Write the same value as that of FLAPH to the flash address pointer H compare register (FLAPHC).

<5> Sets the verify end address to the flash address pointer L compare register (FLAPLC).

<6> Clear the flash status register (PFS).

<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].

<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)

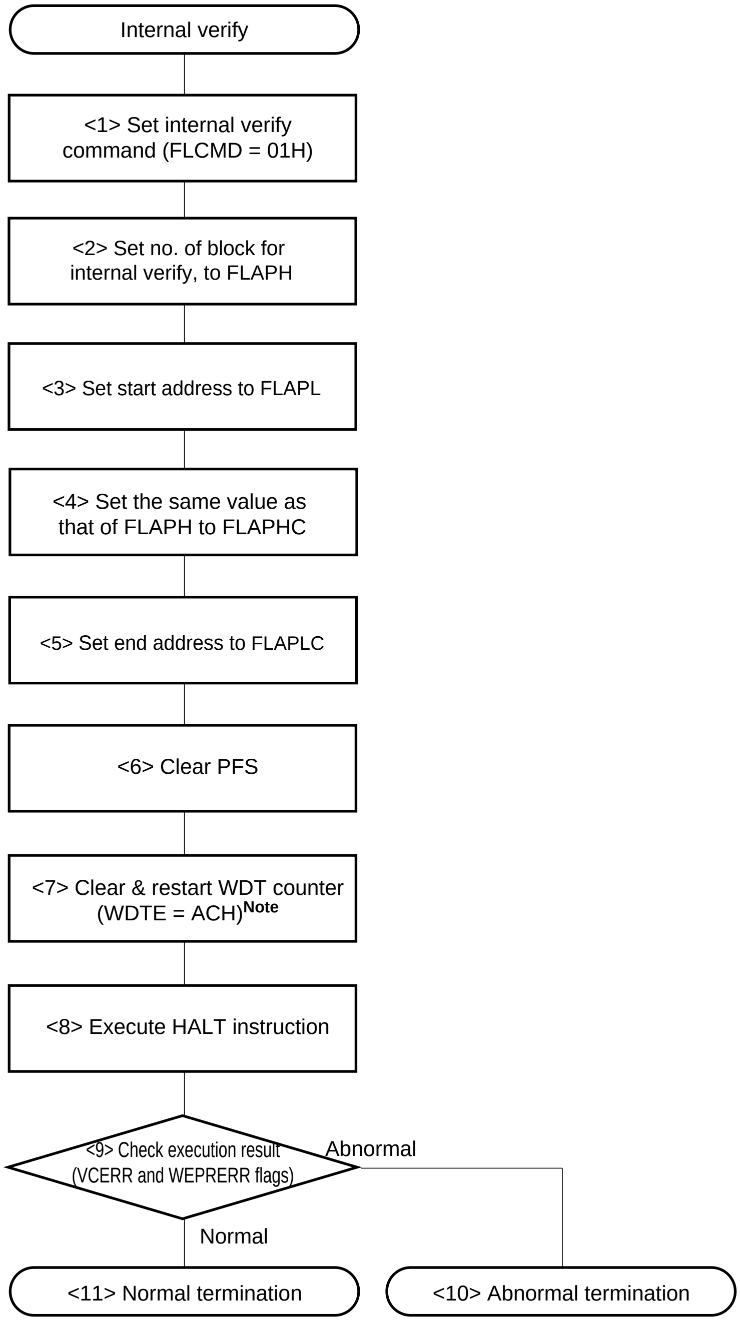
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>

<10> Internal verify processing is abnormally terminated.

<11> Internal verify processing is normally terminated.

**Note** This setting is not required when the watchdog timer is not used.

**Figure 19-25. Example of Internal Verify Operation in Self Programming Mode**


Internal verify
<1> Set internal verify command (FLCMD = 01H)
<2> Set no. of block for internal verify, to FLAPH
<3> Set start address to FLAPL
<4> Set the same value as that of FLAPH to FLAPHC
<5> Set end address to FLAPLC
<6> Clear PFS
<7> Clear & restart WDT counter (WDTE = ACH)Note
<8> Execute HALT instruction
<9> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
<11> Normal termination
<10> Abnormal termination

**Note** This setting is not required when the watchdog timer is not used.

**Remark** <1> to <11> in Figure 19-25 correspond to <1> to <11> in **19.8.9** (previous page).

An example of a program list that performs an internal verify in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------
FlashVerify:
       MOV      FLCMD,#01H      ; Sets flash control command (internal verify)
       MOV      FLAPH,#07H      ; Sets verify start address with FLAPH (block 7 is specified
                                ; here)
       MOV      FLAPL,#00H      ; Sets verify start address with FLAPL (Address 00H is
                                ; specified here)
       MOV      FLAPHC,#07H
       MOV      FLAPLC,#20H     ; Sets verify end address

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

### 19.8.10 Examples of operation when command execution time should be minimized in self programming mode

Examples of operation when the command execution time should be minimized in self programming mode are explained below.

**(1) Erasure to blank check**

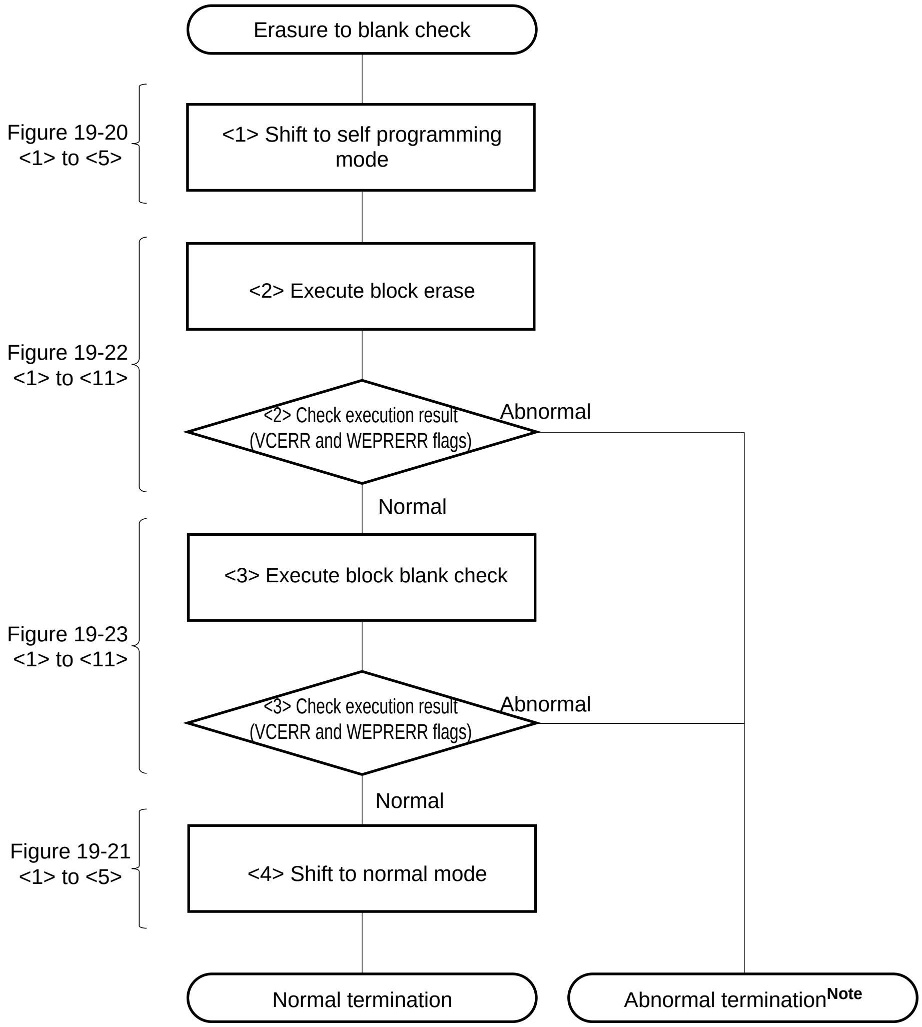
<1> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)

<2> Execution of block erase → Error check (<1> to <11> in **19.8.6**)

<3> Execution of block blank check → Error check (<1> to <11> in **19.8.7**)

<4> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)

**Figure 19-26. Example of Operation When Command Execution Time Should Be Minimized (from Erasure to Blank Check)**



**Note** Perform processing to shift to normal mode in order to return to normal processing.

**Remark** <1> to <4> in Figure 19-26 correspond to <1> to <4> in **19.8.10 (1)** above.

An example of a program list when the command execution time (from erasure to black check) should be minimized in self programming mode is shown below.

```
;----------------------------------------------------------------------
;START
;----------------------------------------------------------------------
        MOV     MK0,#11111111B   ; Masks all interrupts
        MOV     MK1,#11111111B

        DI

ModeOnLoop:
        MOV     PFS,#00H
        MOV     PFCMD,#0A5H      ; PFCMD register control
        MOV     FLPMC,#01H       ; FLPMC register control (sets value)
        MOV     FLPMC,#0FEH      ; FLPMC register control (inverts set value)
        MOV     FLPMC,#01H       ; Sets self programming mode with FLPMC register control (sets
                                 ; value)

        MOV     A,PFS
        CMP     A,#00H
        BNZ     $ModeOnLoop      ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs

FlashBlockErase:
        MOV     FLCMD,#03H       ; Sets flash control command (block erase)
        MOV     FLAPH,#07H       ; Sets number of block to be erased (block 7 is specified
                                 ; here)
        MOV     FLAPL,#00H       ; Fixes FLAPL to “00H”
        MOV     FLAPHC,#07H      ; Sets erase block compare number (same value as that of
                                 ; FLAPH)
        MOV     FLAPLC,#00H      ; Fixes FLAPLC to “00H”

        MOV     PFS,#00H         ; Clears flash status register
        MOV     WDTE,#0ACH       ; Clears & restarts WDT
        HALT                     ; Self programming is started
        MOV     A,PFS
        CMP     A,#00H
        BNZ     $StatusError     ; Checks erase error
                                 ; Performs abnormal termination processing when an error
                                 ; occurs.

FlashBlockBlankCheck:
        MOV     FLCMD,#04H       ; Sets flash control command (block blank check)
        MOV     FLAPH,#07H       ; Sets number of block for blank check (block 7 is specified
                                 ; here)
        MOV     FLAPL,#00H       ; Fixes FLAPL to “00H”
```

```
       MOV      FLAPHC,#07H      ; Sets blank check block compare number (same value as of
                                 ; FLAPH)
       MOV      FLAPLC,#0FFH     ; Fixes FLAPLC to “FFH”
       MOV      PFS,#00H         ; Clears flash status register
       MOV      WDTE,#0ACH       ; Clears & restarts WDT
       HALT                      ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError     ; Checks blank check error
                                 ; Performs abnormal termination processing when an error
                                 ; occurs.

ModeOffLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H      ; PFCMD register control
       MOV      FLPMC,#00H       ; FLPMC register control (sets value)
       MOV      FLPMC,#0FFH      ; FLPMC register control (inverts set value)
       MOV      FLPMC,#00H       ; Sets normal mode via FLPMC register control (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOffLoop     ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs

       MOV      MK0,#INT_MK0     ; Restores interrupt mask flag
       MOV      MK1,#INT_MK1

       EI

       BR       StatusNormal

;----------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;----------------------------------------------------------------------
StatusError:

;----------------------------------------------------------------------
;END (normal termination processing)
;----------------------------------------------------------------------
StatusNormal:
```

### (2) Write to internal verify

<1> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)

<2> Specification of source data for write

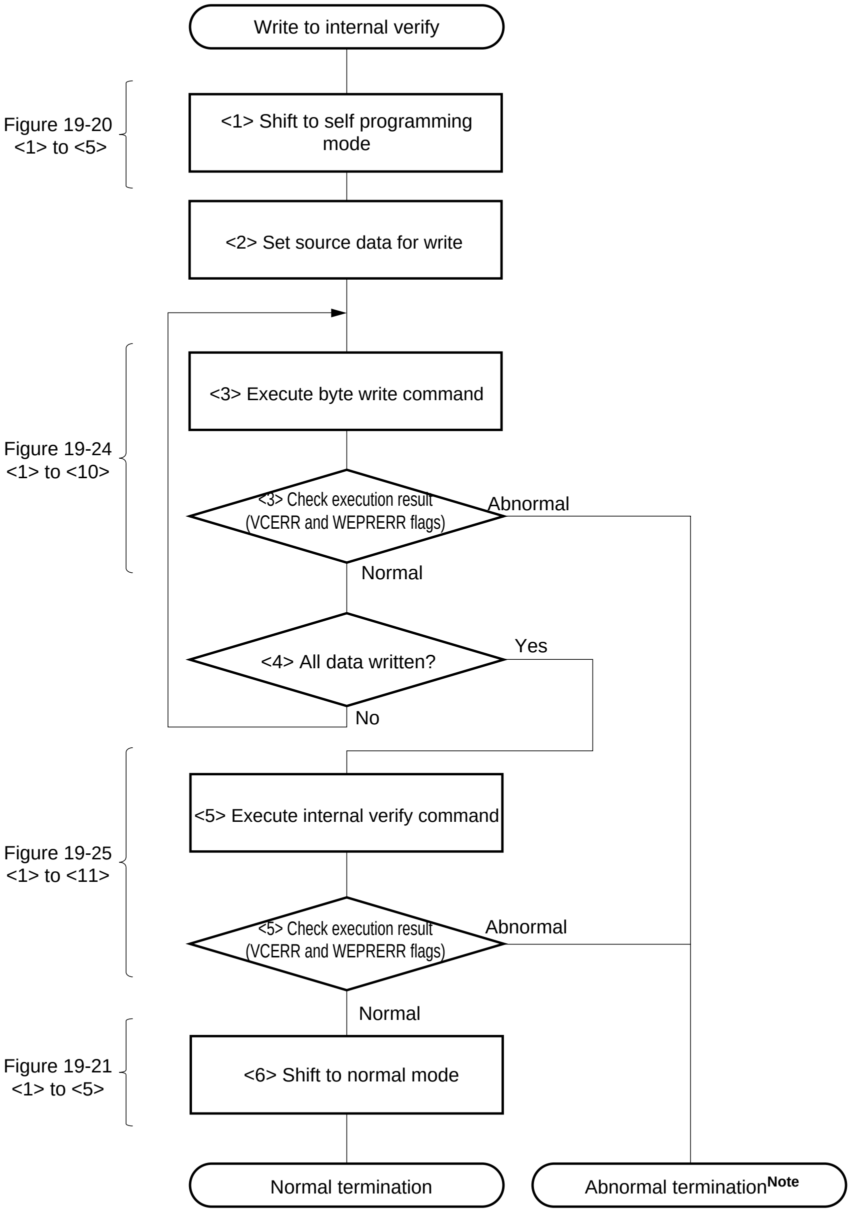
<3> Execution of byte write → Error check (<1> to <10> in **19.8.8**)

<4> <3> is repeated until all data are written.

<5> Execution of internal verify → Error check (<1> to <11> in **19.8.9**)

<6> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)

**Figure 19-27. Example of Operation When Command Execution Time Should Be Minimized (from Write to Internal Verify)**


Write to internal verify
Figure 19-20
<1> to <5>
<1> Shift to self programming mode
<2> Set source data for write
<3> Execute byte write command
Figure 19-24
<1> to <10>
<3> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
<4> All data written?
Yes
No
<5> Execute internal verify command
Figure 19-25
<1> to <11>
<5> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
Figure 19-21
<1> to <5>
<6> Shift to normal mode
Normal termination
Abnormal termination Note

**Note** Perform processing to shift to normal mode in order to return to normal processing.

**Remark** <1> to <6> in Figure 19-27 correspond to <1> to <6> in **19.8.10 (2)** above.

An example of a program list when the command execution time (from write to internal verify) should be minimized in self programming mode is shown below.

```
;----------------------------------------------------------------------
;START
;----------------------------------------------------------------------
       MOV      MK0,#11111111B   ; Masks all interrupts
       MOV      MK1,#11111111B

       DI

ModeOnLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H      ; PFCMD register control
       MOV      FLPMC,#01H       ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH      ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H       ; Sets self programming mode with FLPMC register control
                                 ; (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOnLoop      ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs

FlashWrite:
       MOVW     HL,#DataAdrTop   ; Sets address at which data to be written is located
       MOVW     DE,#WriteAdr     ; Sets address at which data is to be written
FlashWriteLoop:
       MOV      FLCMD,#05H       ; Sets flash control command (byte write)
       MOV      A,D
       MOV      FLAPH,A          ; Sets address at which data is to be written
       MOV      A,E
       MOV      FLAPL,A          ; Sets address at which data is to be written
       MOV      A,[HL]
       MOV      FLW,A            ; Sets data to be written

       MOV      PFS,#00H         ; Clears flash status register
       MOV      WDTE,#0ACH       ; Clears & restarts WDT
       HALT                      ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError     ; Checks write error
                                 ; Performs abnormal termination processing when an error
                                 ; occurs.

       INCW     HL               ; address at which data to be written is located + 1
       MOVW     AX,HL
       CMPW     AX,#DataAdrBtm   ; Performs internal verify processing
```

```
        BNC      $FlashVerify      ; if write of all data is completed

        INCW     DE                ; Address at which data is to be written + 1
        BR       FlashWriteLoop

FlashVerify:
        MOVW     HL,#WriteAdr      ; Sets verify address

        MOV      FLCMD,#01H        ; Sets flash control command (internal verify)
        MOV      A,H
        MOV      FLAPH,A           ; Sets verify start address
        MOV      A,L
        MOV      FLAPL,A           ; Sets verify start address
        MOV      A,D
        MOV      FLAPHC,A          ; Sets verify end address
        MOV      A,E
        MOV      FLAPLC,A          ; Sets verify end address

        MOV      PFS,#00H          ; Clears flash status register
        MOV      WDTE,#0ACH        ; Clears & restarts WDT
        HALT                       ; Self programming is started
        MOV      A,PFS
        CMP      A,#00H
        BNZ      $StatusError      ; Checks internal verify error
                                   ; Performs abnormal termination processing when an error
                                   ; occurs.

ModeOffLoop:
        MOV      PFS,#00H
        MOV      PFCMD,#0A5H       ; PFCMD register control
        MOV      FLPMC,#00H        ; FLPMC register control (sets value)
        MOV      FLPMC,#0FFH       ; FLPMC register control (inverts set value)
        MOV      FLPMC,#00H        ; Sets normal mode via FLPMC register control (sets value)

        MOV      A,PFS
        CMP      A,#00H
        BNZ      $ModeOffLoop      ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs

        MOV      MK0,#INT_MK0      ; Restores interrupt mask flag
        MOV      MK1,#INT_MK1

        EI

        BR       StatusNormal
```

```
;----------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;----------------------------------------------------------------------
StatusError:



;----------------------------------------------------------------------
;END (normal termination processing)
;----------------------------------------------------------------------
StatusNormal:



;----------------------------------------------------------------------
; Data to be written
;----------------------------------------------------------------------
DataAdrTop:
       DB       XXH
       DB       XXH
       DB       XXH
       DB       XXH

        :
        :

       DB       XXH
DataAdrBtm:
;----------------------------------------------------------------------
```

### 19.8.11 Examples of operation when interrupt-disabled time should be minimized in self programming mode

Examples of operation when the interrupt-disabled time should be minimized in self programming mode are explained below.

**(1) Erasure to blank check**

<1> Specification of block erase command (<1> to <5> in **19.8.6**)

<2> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)

<3> Execution of block erase command → Error check (<6> to <11> in **19.8.6**)

<4> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)

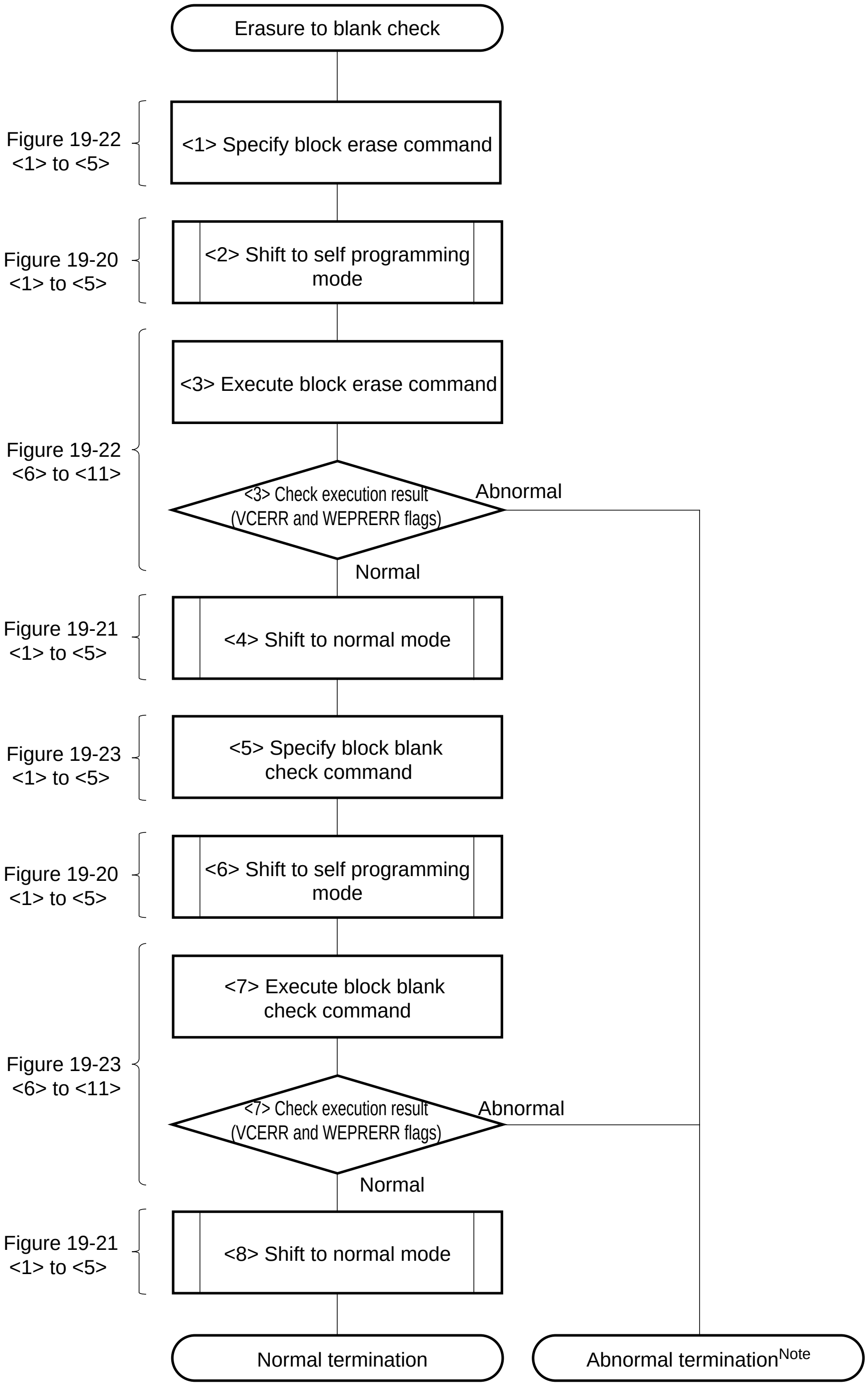
<5> Specification of block blank check command (<1> to <5> in **19.8.7**)

<6> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)

<7> Execution of block blank check command → Error check (<6> to <11> in **19.8.7**)

<8> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)

**Figure 19-28. Example of Operation When Interrupt-Disabled Time Should Be Minimized (from Erasure to Blank Check)**


Erasure to blank check
Figure 19-22 <1> to <5>
<1> Specify block erase command
Figure 19-20 <1> to <5>
<2> Shift to self programming mode
Figure 19-22 <6> to <11>
<3> Execute block erase command
<3> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
Figure 19-21 <1> to <5>
<4> Shift to normal mode
Figure 19-23 <1> to <5>
<5> Specify block blank check command
Figure 19-20 <1> to <5>
<6> Shift to self programming mode
Figure 19-23 <6> to <11>
<7> Execute block blank check command
<7> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
Figure 19-21 <1> to <5>
<8> Shift to normal mode
Normal termination
Abnormal termination Note

**Note** Perform processing to shift to normal mode in order to return to normal processing.

**Remark** <1> to <8> in Figure 19-28 correspond to <1> to <8> in **19.8.11 (1)** (previous page).

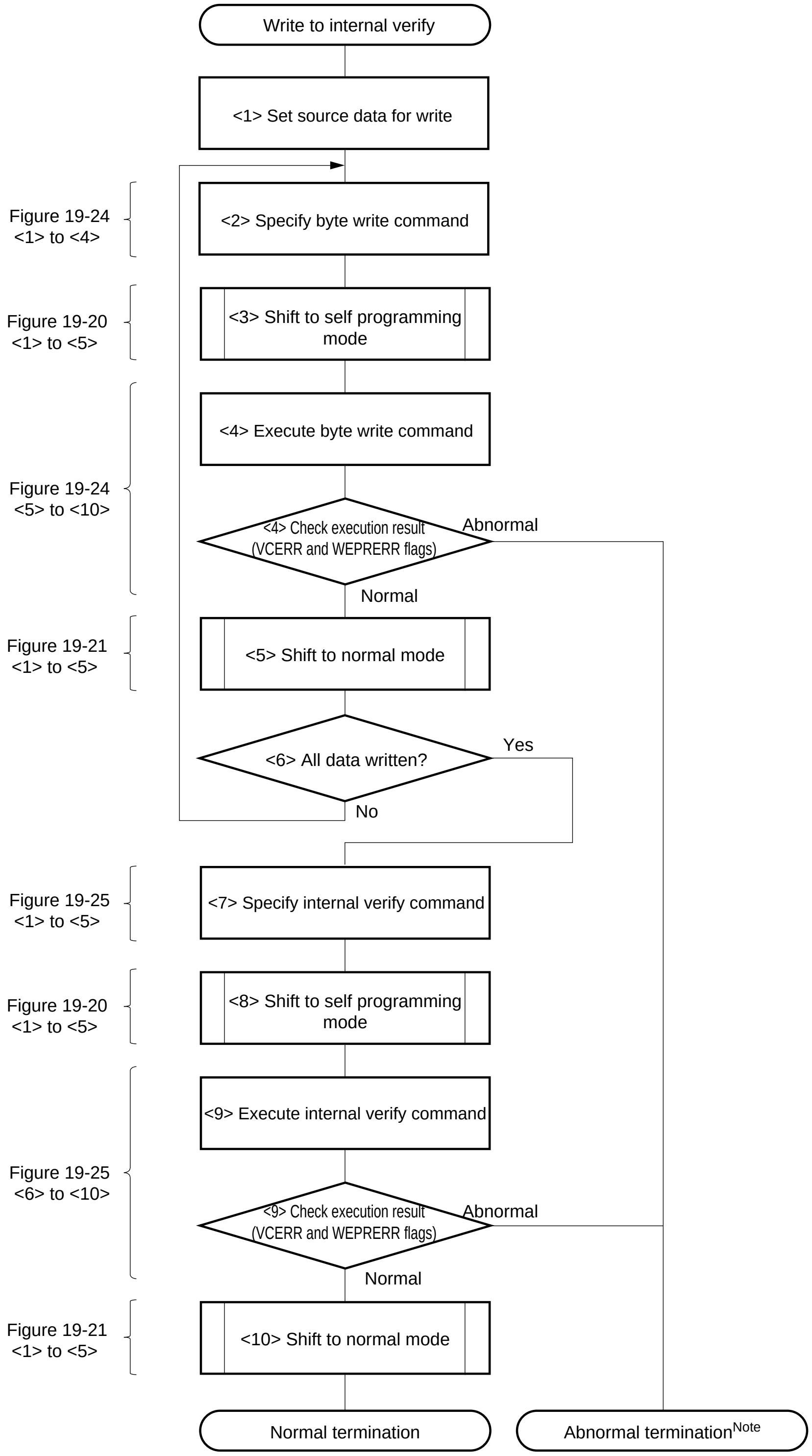
An example of a program list when the interrupt-disabled time (from erasure to blank check) should be minimized in self programming mode is shown below.

```
;----------------------------------------------------------------
;START
;----------------------------------------------------------------
FlashBlockErase:
       ; Sets erase command
       MOV      FLCMD,#03H      ; Sets flash control command (block erase)
       MOV      FLAPH,#07H      ; Sets number of block to be erased (block 7 is specified here)
       MOV      FLAPL,#00H      ; Fixes FLAPL to “00H”
       MOV      FLAPHC,#07H     ; Sets erase block compare number (same value as that of FLAPH)
       MOV      FLAPLC,#00H     ; Fixes FLAPLC to “00H”

       CALL     !ModeOn         ; Shift to self programming mode

       ; Execution of erase command
       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError    ; Checks erase error
       CALL     !ModeOff       ; Shift to normal mode

       BR       StatusNormal

;---------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;---------------------------------------------------------------------
StatusError:



;---------------------------------------------------------------------
;END (normal termination processing)
;---------------------------------------------------------------------
StatusNormal:



;---------------------------------------------------------------------
;Processing to shift to self programming mode
;---------------------------------------------------------------------
ModeOn:
       MOV      MK0,#11111111B   ; Masks all interrupts
       MOV      MK1,#11111111B

       DI

ModeOnLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H    ; PFCMD register control
       MOV      FLPMC,#01H     ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH    ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H     ; Sets self programming mode via FLPMC register control (sets
                               ; value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOnLoop    ; Checks completion of write to specific registers
                               ; Repeats the same processing when an error occurs

       RET

;---------------------------------------------------------------------
; Processing to shift to normal mode
;---------------------------------------------------------------------
ModeOff:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H    ; PFCMD register control
       MOV      FLPMC,#00H     ; FLPMC register control (sets value)
       MOV      FLPMC,#0FFH     ; FLPMC register control (inverts set value)
       MOV      FLPMC,#00H      ; Sets normal mode via FLPMC register control (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOff        ; Checks completion of write to specific registers
                                ; Repeats the same processing when an error occurs

       MOV      MK0,#INT_MK0    ; Restores interrupt mask flag
       MOV      MK1,#INT_MK1

       EI

       RET
```

### (2) Write to internal verify

<1> Specification of source data for write
<2> Specification of byte write command (<1> to <4> in **19.8.8**)
<3> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)
<4> Execution of byte write command → Error check (<5> to <10> in **19.8.8**)
<5> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)
<6> <2> to <5> is repeated until all data are written.
<7> The internal verify command is specified (<1> to <5> in **19.8.9**)
<8> Mode is shifted from normal mode to self programming mode (<1> to <5> in **19.8.4**)
<9> Execution of internal verify command → Error check (<6> to <11> in **19.8.9**)
<10> Mode is shifted from self programming mode to normal mode (<1> to <5> in **19.8.5**)

**Figure 19-29. Example of Operation When Interrupt-Disabled Time Should Be Minimized (from Write to Internal Verify)**


Write to internal verify
<1> Set source data for write
Figure 19-24 <1> to <4>
<2> Specify byte write command
Figure 19-20 <1> to <5>
<3> Shift to self programming mode
Figure 19-24 <5> to <10>
<4> Execute byte write command
<4> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
Figure 19-21 <1> to <5>
<5> Shift to normal mode
<6> All data written?
Yes
No
Figure 19-25 <1> to <5>
<7> Specify internal verify command
Figure 19-20 <1> to <5>
<8> Shift to self programming mode
Figure 19-25 <6> to <10>
<9> Execute internal verify command
<9> Check execution result (VCERR and WEPRERR flags)
Abnormal
Normal
Figure 19-21 <1> to <5>
<10> Shift to normal mode
Normal termination
Abnormal termination Note

**Note** Perform processing to shift to normal mode in order to return to normal processing.

**Remark** <1> to <10> in Figure 19-29 correspond to <1> to <10> in **19.8.11 (2)** (previous page).

An example of a program list when the interrupt-disabled time (from write to internal verify) should be minimized in self programming mode is shown below.

```
;--------------------------------------------------------------------
;START
;--------------------------------------------------------------------
       ; Sets write command
FlashWrite:
       MOVW     HL,#DataAdrTop ; Sets address at which data to be written is located
       MOVW     DE,#WriteAdr   ; Sets address at which data is to be written

FlashWriteLoop:
       MOV      FLCMD,#05H     ; Sets flash control command (byte write)
       MOV      A,D
       MOV      FLAPH,A        ; Sets address at which data is to be written
       MOV      A,E
       MOV      FLAPL,A        ; Sets address at which data is to be written
       MOV      A,[HL]
       MOV      FLW,A          ; Sets data to be written

       CALL     !ModeOn        ; Shift to self programming mode

       ; Execution of write command
       MOV      PFS,#00H       ; Clears flash status register
       MOV      WDTE,#0ACH     ; Clears & restarts WDT
       HALT                    ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError   ; Checks write error
                               ; Performs abnormal termination processing when an error
                               ; occurs.

       CALL     !ModeOff       ; Shift to normal mode

       MOV      MK0,#INT_MK0   ; Restores interrupt mask flag
       MOV      MK1,#INT_MK1

       EI

       ; Judgment of writing all data
       INCW     HL             ; Address at which data to be written is located + 1
       MOVW     AX,HL
       CMPW     AX,#DataAdrBtm ; Performs internal verify processing
       BNC      $FlashVerify   ; if write of all data is completed

       INCW     DE             ; Address at which data is to be written + 1
       BR       FlashWriteLoop
       ; Setting internal verify command
FlashVerify:
       MOVW     HL,#WriteAdr   ; Sets verify address

       MOV      FLCMD,#01H     ; Sets flash control command (internal verify)
       MOV      A,H
       MOV      FLAPH,A        ; Sets verify start address
       MOV      A,L
       MOV      FLAPL,A        ; Sets verify start address
       MOV      A,D
       MOV      FLAPHC,A       ; Sets verify end address
       MOV      A,E
       MOV      FLAPLC,A       ; Sets verify end address

       CALL     !ModeOn        ; Shift to self programming mode

       ; Execution of internal verify command
       MOV      PFS,#00H       ; Clears flash status register
       MOV      WDTE,#0ACH     ; Clears & restarts WDT
       HALT                    ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError   ; Checks internal verify error
                               ; Performs abnormal termination processing when an error occurs

       CALL     !ModeOff       ; Shift to normal mode

       BR       StatusNormal

;---------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;---------------------------------------------------------------------
StatusError:



;---------------------------------------------------------------------
;END (normal termination processing)
;---------------------------------------------------------------------
StatusNormal:



;---------------------------------------------------------------------
;Processing to shift to self programming mode
;---------------------------------------------------------------------
ModeOn:
       MOV      MK0,#11111111B ; Masks all interrupts
       MOV      MK1,#11111111B
       DI

ModeOnLoop:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H     ; PFCMD register control
       MOV      FLPMC,#01H      ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH     ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H      ; Sets self programming mode via FLPMC register control (sets
                                ; value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOnLoop     ; Checks completion of write to specific registers
                                ; Repeats the same processing when an error occurs

       RET


;----------------------------------------------------------------------
; Processing to shift to normal mode
;----------------------------------------------------------------------
ModeOff:
       MOV      PFS,#00H
       MOV      PFCMD,#0A5H     ; PFCMD register control
       MOV      FLPMC,#00H      ; FLPMC register control (sets value)
       MOV      FLPMC,#0FFH     ; FLPMC register control (inverts set value)
       MOV      FLPMC,#00H      ; Sets normal mode via FLPMC register control (sets value)

       MOV      A,PFS
       CMP      A,#00H
       BNZ      $ModeOff        ; Checks completion of write to specific registers
                                ; Repeats the same processing when an error occurs□

       MOV      MK0,#INT_MK0    ; Restores interrupt mask flag
       MOV      MK1,#INT_MK1

       EI

       RET


;----------------------------------------------------------------------
;Data to be written
;----------------------------------------------------------------------
DataAdrTop:
       DB       XXH
       DB       XXH
       DB       XXH
       DB       XXH

       :
       :

       DB       XXH
DataAdrBtm:
;-------------------------------------------------------------------
```

# CHAPTER 20 INSTRUCTION SET OVERVIEW

This chapter lists the instruction set of the 78K0S/KB1+. For details of the operation and machine language (instruction code) of each instruction, refer to **78K/0S Series Instructions User's Manual (U11047E)**.

## 20.1 Operation

### 20.1.1 Operand identifiers and description methods

Operands are described in "Operand" column of each instruction in accordance with the description method of the instruction operand identifier (refer to the assembler specifications for details). When there are two or more description methods, select one of them. Uppercase letters and the symbols #, !, $, and [ ] are key words and are described as they are. Each symbol has the following meaning.

- #: Immediate data specification
- !: Absolute address specification
- $: Relative address specification
- [ ]: Indirect address specification

In the case of immediate data, describe an appropriate numeric value or a label. When using a label, be sure to describe the #, !, $ and [ ] symbols.

For operand register identifiers, r and rp, either function names (X, A, C, etc.) or absolute names (names in parentheses in the table below, R0, R1, R2, etc.) can be used for description.

**Table 20-1. Operand Identifiers and Description Methods**

| Identifier | Description Method |
|---|---|
| r<br>rp<br>sfr | X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7)<br>AX (RP0), BC (RP1), DE (RP2), HL (RP3)<br>Special function register symbol |
| saddr<br>saddrp | FE20H to FF1FH Immediate data or labels<br>FE20H to FF1FH Immediate data or labels (even addresses only) |
| addr16<br>addr5 | 0000H to FFFFH Immediate data or labels (only even addresses for 16-bit data transfer instructions)<br>0040H to 007FH Immediate data or labels (even addresses only) |
| word<br>byte<br>bit | 16-bit immediate data or label<br>8-bit immediate data or label<br>3-bit immediate data or label |

**Remark** For symbols of special function registers, see **Table 3-3 Special Function Registers**.

### 20.1.2 Description of “Operation” column

| | |
|---|---|
| A: | A register; 8-bit accumulator |
| X: | X register |
| B: | B register |
| C: | C register |
| D: | D register |
| E: | E register |
| H: | H register |
| L: | L register |
| AX: | AX register pair; 16-bit accumulator |
| BC: | BC register pair |
| DE: | DE register pair |
| HL: | HL register pair |
| PC: | Program counter |
| SP: | Stack pointer |
| PSW: | Program status word |
| CY: | Carry flag |
| AC: | Auxiliary carry flag |
| Z: | Zero flag |
| IE: | Interrupt request enable flag |
| ( ): | Memory contents indicated by address or register contents in parentheses |
| ×H, ×L: | Higher 8 bits and lower 8 bits of 16-bit register |
| ∧: | Logical product (AND) |
| ∨: | Logical sum (OR) |
| ⊻: | Exclusive logical sum (exclusive OR) |
| ‾: | Inverted data |
| addr16: | 16-bit immediate data or label |
| jdisp8: | Signed 8-bit data (displacement value) |

### 20.1.3 Description of “Flag” column

| | |
|---|---|
| (Blank): | Unchanged |
| 0: | Cleared to 0 |
| 1: | Set to 1 |
| ×: | Set/cleared according to the result |
| R: | Previously saved value is stored |

## 20.2 Operation List

| Mnemonic | Operand | Bytes | Clocks | Operation | Flag | | |
|---|---|---|---|---|---|---|---|
| | | | | | Z | AC | CY |
| MOV | r, #byte | 3 | 6 | r ← byte | | | |
| | saddr, #byte | 3 | 6 | (saddr) ← byte | | | |
| | sfr, #byte | 3 | 6 | sfr ← byte | | | |
| | A, r Note 1 | 2 | 4 | A ← r | | | |
| | r, A Note 1 | 2 | 4 | r ← A | | | |
| | A, saddr | 2 | 4 | A ← (saddr) | | | |
| | saddr, A | 2 | 4 | (saddr) ← A | | | |
| | A, sfr | 2 | 4 | A ← sfr | | | |
| | sfr, A | 2 | 4 | sfr ← A | | | |
| | A, !addr16 | 3 | 8 | A ← (addr16) | | | |
| | !addr16, A | 3 | 8 | (addr16) ← A | | | |
| | PSW, #byte | 3 | 6 | PSW ← byte | × | × | × |
| | A, PSW | 2 | 4 | A ← PSW | | | |
| | PSW, A | 2 | 4 | PSW ← A | × | × | × |
| | A, [DE] | 1 | 6 | A ← (DE) | | | |
| | [DE], A | 1 | 6 | (DE) ← A | | | |
| | A, [HL] | 1 | 6 | A ← (HL) | | | |
| | [HL], A | 1 | 6 | (HL) ← A | | | |
| | A, [HL + byte] | 2 | 6 | A ← (HL + byte) | | | |
| | [HL + byte], A | 2 | 6 | (HL + byte) ← A | | | |
| XCH | A, X | 1 | 4 | A ↔ X | | | |
| | A, r Note 2 | 2 | 6 | A ↔ r | | | |
| | A, saddr | 2 | 6 | A ↔ (saddr) | | | |
| | A, sfr | 2 | 6 | A ↔ sfr | | | |
| | A, [DE] | 1 | 8 | A ↔ (DE) | | | |
| | A, [HL] | 1 | 8 | A ↔ (HL) | | | |
| | A, [HL, byte] | 2 | 8 | A ↔ (HL + byte) | | | |

**Notes 1.** Except r = A.

**2.** Except r = A, X.

**Remark** One instruction clock cycle is one CPU clock cycle ($f_{CPU}$) selected by the processor clock control register (PCC).

| Mnemonic | Operand | Bytes | Clocks | Operation | Flag | | |
|---|---|---|---|---|---|---|---|
| | | | | | Z | AC | CY |
| MOVW | rp, #word | 3 | 6 | rp ← word | | | |
| | AX, saddrp | 2 | 6 | AX ← (saddrp) | | | |
| | saddrp, AX | 2 | 8 | (saddrp) ← AX | | | |
| | AX, rp[Note] | 1 | 4 | AX ← rp | | | |
| | rp, AX[Note] | 1 | 4 | rp ← AX | | | |
| XCHW | AX, rp[Note] | 1 | 8 | AX ↔ rp | | | |
| ADD | A, #byte | 2 | 4 | A, CY ← A + byte | × | × | × |
| | saddr, #byte | 3 | 6 | (saddr), CY ← (saddr) + byte | × | × | × |
| | A, r | 2 | 4 | A, CY ← A + r | × | × | × |
| | A, saddr | 2 | 4 | A, CY ← A + (saddr) | × | × | × |
| | A, !addr16 | 3 | 8 | A, CY ← A + (addr16) | × | × | × |
| | A, [HL] | 1 | 6 | A, CY ← A + (HL) | × | × | × |
| | A, [HL + byte] | 2 | 6 | A, CY ← A + (HL + byte) | × | × | × |
| ADDC | A, #byte | 2 | 4 | A, CY ← A + byte + CY | × | × | × |
| | saddr, #byte | 3 | 6 | (saddr), CY ← (saddr) + byte + CY | × | × | × |
| | A, r | 2 | 4 | A, CY ← A + r + CY | × | × | × |
| | A, saddr | 2 | 4 | A, CY ← A + (saddr) + CY | × | × | × |
| | A, !addr16 | 3 | 8 | A, CY ← A + (addr16) + CY | × | × | × |
| | A, [HL] | 1 | 6 | A, CY ← A + (HL) + CY | × | × | × |
| | A, [HL + byte] | 2 | 6 | A, CY ← A + (HL + byte) + CY | × | × | × |
| SUB | A, #byte | 2 | 4 | A, CY ← A – byte | × | × | × |
| | saddr, #byte | 3 | 6 | (saddr), CY ← (saddr) – byte | × | × | × |
| | A, r | 2 | 4 | A, CY ← A – r | × | × | × |
| | A, saddr | 2 | 4 | A, CY ← A – (saddr) | × | × | × |
| | A, !addr16 | 3 | 8 | A, CY ← A – (addr16) | × | × | × |
| | A, [HL] | 1 | 6 | A, CY ← A – (HL) | × | × | × |
| | A, [HL + byte] | 2 | 6 | A, CY ← A – (HL + byte) | × | × | × |

**Note** Only when rp = BC, DE, or HL.

**Remark** One instruction clock cycle is one CPU clock cycle ($f_{CPU}$) selected by the processor clock control register (PCC).

| Mnemonic | Operand | Bytes | Clocks | Operation | Flag | | |
|---|---|---|---|---|---|---|---|
| | | | | | Z | AC | CY |
| SUBC | A, #byte | 2 | 4 | A, CY ← A – byte – CY | × | × | × |
| | saddr, #byte | 3 | 6 | (saddr), CY ← (saddr) – byte – CY | × | × | × |
| | A, r | 2 | 4 | A, CY ← A – r – CY | × | × | × |
| | A, saddr | 2 | 4 | A, CY ← A – (saddr) – CY | × | × | × |
| | A, !addr16 | 3 | 8 | A, CY ← A – (addr16) – CY | × | × | × |
| | A, [HL] | 1 | 6 | A, CY ← A – (HL) – CY | × | × | × |
| | A, [HL + byte] | 2 | 6 | A, CY ← A – (HL + byte) – CY | × | × | × |
| AND | A, #byte | 2 | 4 | A ← A ∧ byte | × | | |
| | saddr, #byte | 3 | 6 | (saddr) ← (saddr) ∧ byte | × | | |
| | A, r | 2 | 4 | A ← A ∧ r | × | | |
| | A, saddr | 2 | 4 | A ← A ∧ (saddr) | × | | |
| | A, !addr16 | 3 | 8 | A ← A ∧ (addr16) | × | | |
| | A, [HL] | 1 | 6 | A ← A ∧ (HL) | × | | |
| | A, [HL + byte] | 2 | 6 | A ← A ∧ (HL + byte) | × | | |
| OR | A, #byte | 2 | 4 | A ← A ∨ byte | × | | |
| | saddr, #byte | 3 | 6 | (saddr) ← (saddr) ∨ byte | × | | |
| | A, r | 2 | 4 | A ← A ∨ r | × | | |
| | A, saddr | 2 | 4 | A ← A ∨ (saddr) | × | | |
| | A, !addr16 | 3 | 8 | A ← A ∨ (addr16) | × | | |
| | A, [HL] | 1 | 6 | A ← A ∨ (HL) | × | | |
| | A, [HL + byte] | 2 | 6 | A ← A ∨ (HL + byte) | × | | |
| XOR | A, #byte | 2 | 4 | A ← A ⊻ byte | × | | |
| | saddr, #byte | 3 | 6 | (saddr) ← (saddr) ⊻ byte | × | | |
| | A, r | 2 | 4 | A ← A ⊻ r | × | | |
| | A, saddr | 2 | 4 | A ← A ⊻ (saddr) | × | | |
| | A, !addr16 | 3 | 8 | A ← A ⊻ (addr16) | × | | |
| | A, [HL] | 1 | 6 | A ← A ⊻ (HL) | × | | |
| | A, [HL + byte] | 2 | 6 | A ← A ⊻ (HL + byte) | × | | |

**Remark** One instruction clock cycle is one CPU clock cycle ($f_{CPU}$) selected by the processor clock control register (PCC).

| Mnemonic | Operand | Bytes | Clocks | Operation | Flag | | |
|---|---|---|---|---|---|---|---|
| | | | | | Z | AC | CY |
| CMP | A, #byte | 2 | 4 | A – byte | × | × | × |
| | saddr, #byte | 3 | 6 | (saddr) – byte | × | × | × |
| | A, r | 2 | 4 | A – r | × | × | × |
| | A, saddr | 2 | 4 | A – (saddr) | × | × | × |
| | A, !addr16 | 3 | 8 | A – (addr16) | × | × | × |
| | A, [HL] | 1 | 6 | A – (HL) | × | × | × |
| | A, [HL + byte] | 2 | 6 | A – (HL + byte) | × | × | × |
| ADDW | AX, #word | 3 | 6 | AX, CY ← AX + word | × | × | × |
| SUBW | AX, #word | 3 | 6 | AX, CY ← AX – word | × | × | × |
| CMPW | AX, #word | 3 | 6 | AX – word | × | × | × |
| INC | r | 2 | 4 | r ← r + 1 | × | × | |
| | saddr | 2 | 4 | (saddr) ← (saddr) + 1 | × | × | |
| DEC | r | 2 | 4 | r ← r – 1 | × | × | |
| | saddr | 2 | 4 | (saddr) ← (saddr) – 1 | × | × | |
| INCW | rp | 1 | 4 | rp ← rp + 1 | | | |
| DECW | rp | 1 | 4 | rp ← rp – 1 | | | |
| ROR | A, 1 | 1 | 2 | (CY, $A_7$ ← $A_0$, $A_{m-1}$ ← $A_m$) × 1 | | | × |
| ROL | A, 1 | 1 | 2 | (CY, $A_0$ ← $A_7$, $A_{m+1}$ ← $A_m$) × 1 | | | × |
| RORC | A, 1 | 1 | 2 | (CY ← $A_0$, $A_7$ ← CY, $A_{m-1}$ ← $A_m$) × 1 | | | × |
| ROLC | A, 1 | 1 | 2 | (CY ← $A_7$, $A_0$ ← CY, $A_{m+1}$ ← $A_m$) × 1 | | | × |
| SET1 | saddr.bit | 3 | 6 | (saddr.bit) ← 1 | | | |
| | sfr.bit | 3 | 6 | sfr.bit ← 1 | | | |
| | A.bit | 2 | 4 | A.bit ← 1 | | | |
| | PSW.bit | 3 | 6 | PSW.bit ← 1 | × | × | × |
| | [HL].bit | 2 | 10 | (HL).bit ← 1 | | | |
| CLR1 | saddr.bit | 3 | 6 | (saddr.bit) ← 0 | | | |
| | sfr.bit | 3 | 6 | sfr.bit ← 0 | | | |
| | A.bit | 2 | 4 | A.bit ← 0 | | | |
| | PSW.bit | 3 | 6 | PSW.bit ← 0 | × | × | × |
| | [HL].bit | 2 | 10 | (HL).bit ← 0 | | | |
| SET1 | CY | 1 | 2 | CY ← 1 | | | 1 |
| CLR1 | CY | 1 | 2 | CY ← 0 | | | 0 |
| NOT1 | CY | 1 | 2 | CY ← $\overline{CY}$ | | | × |

**Remark** One instruction clock cycle is one CPU clock cycle ($f_{CPU}$) selected by the processor clock control register (PCC).

| Mnemonic | Operand | Bytes | Clocks | Operation | Flag | | |
|---|---|---|---|---|---|---|---|
| | | | | | Z | AC | CY |
| CALL | !addr16 | 3 | 6 | (SP – 1) ← $(PC + 3)_H$, (SP – 2) ← $(PC + 3)_L$,<br>PC ← addr16, SP ← SP – 2 | | | |
| CALLT | [addr5] | 1 | 8 | (SP – 1) ← $(PC + 1)_H$, (SP – 2) ← $(PC + 1)_L$,<br>$PC_H$ ← (00000000, addr5 + 1),<br>$PC_L$ ← (00000000, addr5), SP ← SP – 2 | | | |
| RET | | 1 | 6 | $PC_H$ ← (SP + 1), $PC_L$ ← (SP), SP ← SP + 2 | | | |
| RETI | | 1 | 8 | $PC_H$ ← (SP + 1), $PC_L$ ← (SP), PSW ← (SP + 2),<br>SP ← SP + 3 | R | R | R |
| PUSH | PSW | 1 | 2 | (SP – 1) ← PSW, SP ← SP – 1 | | | |
| | rp | 1 | 4 | (SP – 1) ← $rp_H$, (SP – 2) ← $rp_L$, SP ← SP – 2 | | | |
| POP | PSW | 1 | 4 | PSW ← (SP), SP ← SP + 1 | R | R | R |
| | rp | 1 | 6 | $rp_H$ ← (SP + 1), $rp_L$ ← (SP), SP ← SP + 2 | | | |
| MOVW | SP, AX | 2 | 8 | SP ← AX | | | |
| | AX, SP | 2 | 6 | AX ← SP | | | |
| BR | !addr16 | 3 | 6 | PC ← addr16 | | | |
| | $addr16 | 2 | 6 | PC ← PC + 2 + jdisp8 | | | |
| | AX | 1 | 6 | $PC_H$ ← A, $PC_L$ ← X | | | |
| BC | $saddr16 | 2 | 6 | PC ← PC + 2 + jdisp8 if CY = 1 | | | |
| BNC | $saddr16 | 2 | 6 | PC ← PC + 2 + jdisp8 if CY = 0 | | | |
| BZ | $saddr16 | 2 | 6 | PC ← PC + 2 + jdisp8 if Z = 1 | | | |
| BNZ | $saddr16 | 2 | 6 | PC ← PC + 2 + jdisp8 if Z = 0 | | | |
| BT | saddr.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if (saddr.bit) = 1 | | | |
| | sfr.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if sfr.bit = 1 | | | |
| | A.bit, $addr16 | 3 | 8 | PC ← PC + 3 + jdisp8 if A.bit = 1 | | | |
| | PSW.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if PSW.bit = 1 | | | |
| BF | saddr.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if (saddr.bit) = 0 | | | |
| | sfr.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if sfr.bit = 0 | | | |
| | A.bit, $addr16 | 3 | 8 | PC ← PC + 3 + jdisp8 if A.bit = 0 | | | |
| | PSW.bit, $addr16 | 4 | 10 | PC ← PC + 4 + jdisp8 if PSW.bit = 0 | | | |
| DBNZ | B, $addr16 | 2 | 6 | B ← B – 1, then PC ← PC + 2 + jdisp8 if B ≠ 0 | | | |
| | C, $addr16 | 2 | 6 | C ← C – 1, then PC ← PC + 2 + jdisp8 if C ≠ 0 | | | |
| | saddr, $addr16 | 3 | 8 | (saddr) ← (saddr) – 1, then<br>PC ← PC + 3 + jdisp8 if (saddr) ≠ 0 | | | |
| NOP | | 1 | 2 | No Operation | | | |
| EI | | 3 | 6 | IE ← 1 (Enable Interrupt) | | | |
| DI | | 3 | 6 | IE ← 0 (Disable Interrupt) | | | |
| HALT | | 1 | 2 | Set HALT Mode | | | |
| STOP | | 1 | 2 | Set STOP Mode | | | |

**Remark** One instruction clock cycle is one CPU clock cycle ($f_{CPU}$) selected by the processor clock control register (PCC).

## 20.3 Instructions Listed by Addressing Type

### (1) 8-bit instructions

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, INC, DEC, ROR, ROL, RORC, ROLC, PUSH, POP, DBNZ

| 2nd Operand<br>1st Operand | #byte | A | r | sfr | saddr | !addr16 | PSW | [DE] | [HL] | [HL + byte] | $addr16 | 1 | None |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | | MOV[Note]<br>XCH[Note]<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | MOV<br>XCH | MOV<br>XCH<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | MOV<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | MOV | MOV<br>XCH | MOV<br>XCH<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | MOV<br>XCH<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | | ROR<br>ROL<br>RORC<br>ROLC | |
| r | MOV | MOV | | | | | | | | | | | INC<br>DEC |
| B, C | | | | | | | | | | | DBNZ | | |
| sfr | MOV | MOV | | | | | | | | | | | |
| saddr | MOV<br>ADD<br>ADDC<br>SUB<br>SUBC<br>AND<br>OR<br>XOR<br>CMP | MOV | | | | | | | | | DBNZ | | INC<br>DEC |
| !addr16 | | MOV | | | | | | | | | | | |
| PSW | MOV | MOV | | | | | | | | | | | PUSH<br>POP |
| [DE] | | MOV | | | | | | | | | | | |
| [HL] | | MOV | | | | | | | | | | | |
| [HL + byte] | | MOV | | | | | | | | | | | |

**Note** Except r = A.

### (2) 16-bit instructions

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

| 2nd Operand<br>1st Operand | #word | AX | rp[Note] | saddrp | SP | None |
|---|---|---|---|---|---|---|
| AX | ADDW SUBW<br>CMPW | | MOVW<br>XCHW | MOVW | MOVW | |
| rp | MOVW | MOVW[Note] | | | | INCW<br>DECW<br>PUSH<br>POP |
| saddrp | | MOVW | | | | |
| sp | | MOVW | | | | |

**Note** Only when rp = BC, DE, or HL.

### (3) Bit manipulation instructions

SET1, CLR1, NOT1, BT, BF

| 2nd Operand<br>1st Operand | $addr16 | None |
|---|---|---|
| A.bit | BT<br>BF | SET1<br>CLR1 |
| sfr.bit | BT<br>BF | SET1<br>CLR1 |
| saddr.bit | BT<br>BF | SET1<br>CLR1 |
| PSW.bit | BT<br>BF | SET1<br>CLR1 |
| [HL].bit | | SET1<br>CLR1 |
| CY | | SET1<br>CLR1<br>NOT1 |

### (4) Call instructions/branch instructions

CALL, CALLT, BR, BC, BNC, BZ, BNZ, DBNZ

| 2nd Operand<br>1st Operand | AX | !addr16 | [addr5] | $addr16 |
|---|---|---|---|---|
| Basic instructions | BR | CALL<br>BR | CALLT | BR<br>BC<br>BNC<br>BZ<br>BNZ |
| Compound instructions | | | | DBNZ |

### (5) Other instructions

RET, RETI, NOP, EI, DI, HALT, STOP

# CHAPTER 21 ELECTRICAL SPECIFICATIONS (TARGET VALUES)

These specifications are only target values, and may not be satisfied by mass-produced products.

**Absolute Maximum Ratings ($T_A$ = 25°C)**

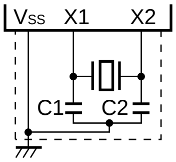
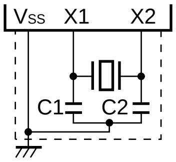
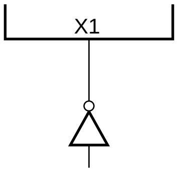
| Parameter | Symbol | Conditions | Ratings | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | –0.3 to +6.5 | V |
| | $V_{SS}$ | | –0.3 to +0.3 | V |
| | $AV_{REF}$ | | –0.3 to $V_{DD}$ + 0.3[Note] | V |
| | $AV_{SS}$ | | –0.3 to +0.3 | V |
| Input voltage | $V_{I1}$ | P00 to P03, P30 to P34, P40 to P47, P120 to P123 | –0.3 to $V_{DD}$ + 0.3[Note] | V |
| | $V_{I2}$ | P20 to 23 | – 0.3 to $AV_{REF}$ + 0.3[Note]<br>and –0.3 to $V_{DD}$ + 0.3[Note] | V |
| Output voltage | $V_O$ | | –0.3 to $V_{DD}$ + 0.3[Note] | V |
| Analog input voltage | $V_{AN}$ | | $AV_{SS}$ – 0.3 to $AV_{REF}$ + 0.3[Note]<br>and –0.3 to $V_{DD}$ + 0.3[Note] | V |
| Output current, high | $I_{OH}$ | Per pin | –10 | mA |
| | | Total of pins | –44 | mA |
| Output current, low | $I_{OL}$ | Per pin | 20 | mA |
| | | Total of all pins | 44 | mA |
| Operating ambient temperature | $T_A$ | In normal operation mode | –40 to +85 | °C |
| | | During flash memory programming | | |
| Storage temperature | $T_{stg}$ | | –40 to +125 | °C |

**Note** Must be 6.5 V or lower

**Caution** **Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.**

**Remark** Unless specified otherwise, the characteristics of alternate-function pins are the same as those of port pins.

### X1 Oscillator Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note 1], $V_{SS}$ = 0 V)

| Resonator | Recommended Circuit | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Oscillation frequency ($f_X$)[Note 2] | | 1 | | 10.0 | MHz |
| Crystal resonator |  | Oscillation frequency ($f_X$)[Note 2] | | 1 | | 10.0 | MHz |
| External clock |  | X1 input frequency ($f_X$)[Note 2] | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 1 | | 10.0 | MHz |
| | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1 | | 5.0 | |
| | | X1 input high-/low-level width ($t_{XH}$, $t_{XL}$) | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 0.045 | | 0.5 | $\mu$s |
| | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 0.09 | | 0.5 | |

**Notes 1.** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

**2.** Indicates only oscillator characteristics. Refer to **AC Characteristics** for instruction execution time.

**Caution When using the X1 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.**

**Remark** For the resonator selection and oscillator constant, users are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

**High-Speed Ring-OSC Oscillator Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note 1], $f_X$ = 8 MHz [Note 2])**

| Resonator | Parameter | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| On-chip high-speed Ring-OSC | Oscillation frequency ($f_X$) | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | $T_A$ = –10 to +80°C | | | ±3 | % |
| | | | $T_A$ = –40 to +85°C | | | ±5 | % |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | | | | T.B.D | % |

**Notes 1.** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

**2.** Indicates only oscillator characteristics. Refer to **AC Characteristics** for instruction execution time.

**Low-Speed Ring-OSC Oscillator Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note], $V_{SS}$ = 0 V)**

| Resonator | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| On-chip low-speed Ring-OSC | Oscillation frequency ($f_{RL}$) | | 120 | 240 | 480 | kHz |

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

### DC Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note], $V_{SS}$ = 0 V) (1/2)

| Parameter | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Output current, high | $I_{OH1}$ | Pins other than P20 to P23 | Per pin | 2.0 V ≤ $V_{DD}$ ≤ 5.5 V | | | –5 | mA |
| | | | Total | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V | | | –25 | mA |
| | | | | 2.0 V ≤ $V_{DD}$ < 4.0 V | | | –15 | mA |
| | $I_{OH2}$ | P20 to P23 | Per pin | 2.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | –5 | mA |
| | | | Total | 2.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | –15 | mA |
| Output current, low | $I_{OL}$ | Per pin | | 2.0 V ≤ $V_{DD}$ ≤ 5.5 V | | | 10 | mA |
| | | Total of all pins | | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V | | | 30 | mA |
| | | | | 2.0 V ≤ $V_{DD}$ < 4.0 V | | | 15 | mA |
| Input voltage, high | $V_{IH1}$ | P00 to P03, P30 to P34, P40 to P47, P120, P123 | | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | P20 to P23 | | | 0.7$AV_{REF}$ | | $AV_{REF}$ | V |
| | $V_{IH3}$ | P121, P122 | | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| Input voltage, low | $V_{IL1}$ | P00 to P03, P30 to P34, P40 to P47, P120, P123 | | | 0 | | 0.2$V_{DD}$ | V |
| | $V_{IL2}$ | P20 to P23 | | | 0 | | 0.3$AV_{REF}$ | V |
| | $V_{IL3}$ | P121, P122 | | | 0 | | 0.3$V_{DD}$ | V |
| Output voltage, high | $V_{OH1}$ | Total of pins other than P20 to P23<br>$I_{OH1}$ = –15 mA | | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V<br>$I_{OH1}$ = –5 mA | $V_{DD}$ – 1.0 | | | V |
| | | $I_{OH1}$ = –100 µA | | 2.0 V ≤ $V_{DD}$ < 4.0 V | $V_{DD}$ – 0.5 | | | V |
| | $V_{OH2}$ | Total of pins P20 to P23<br>$I_{OH2}$ = –10 mA | | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V<br>$I_{OH2}$ = –5 mA | $AV_{REF}$ – 1.0 | | | V |
| | | | | 2.0 V ≤ $AV_{REF}$ < 4.0 V<br>$I_{OH2}$ = –5 mA | $AV_{REF}$ – 0.5 | | | V |
| Output voltage, low | $V_{OL}$ | Total of pins<br>$I_{OL}$ = 30 mA | | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V<br>$I_{OL}$ = 10 mA | | | 1.3 | V |
| | | 2.0 V ≤ $V_{DD}$ < 4.0 V<br>$I_{OL}$ = 400 µA | | | | | 0.4 | V |
| Input leakage current, high | $I_{LIH1}$ | $V_I$ = $V_{DD}$ | | Pins other than X1 | | | 3 | µA |
| Input leakage current, low | $I_{LIL1}$ | $V_I$ = 0 V | | Pins other than X1 | | | –3 | µA |
| Output leakage current, high | $I_{LOH}$ | $V_O$ = $V_{DD}$ | | Pins other than X2 | | | 3 | µA |
| Output leakage current, low | $I_{LOL}$ | $V_O$ = 0 V | | Pins other than X2 | | | –3 | µA |

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

**Remark** Unless specified otherwise, the characteristics of alternate-function pins are the same as those of port pins.

### DC Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note 1], $V_{SS}$ = 0 V) (2/2)

| Parameter | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Pull-up resistance | $R_{PU}$ | $V_I$ = 0 V | | | 10 | 30 | 100 | kΩ |
| Pull-down resistance | $R_{PD}$ | P121, P122, reset status | | | 10 | 30 | 100 | kΩ |
| Supply current[Note 2] | $I_{DD1}$[Note 3] | Crystal/ceramic oscillation, external clock input oscillation operating mode[Note 6] | $f_X$ = 10 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When A/D converter is stopped | | 6.1 | 12.2 | mA |
| | | | | When A/D converter is operating[Note 8] | | 7.6 | 15.2 | |
| | | | $f_X$ = 6 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When A/D converter is stopped | | 5.5 | 11.0 | mA |
| | | | | When A/D converter is operating[Note 8] | | | 14.0 | |
| | | | $f_X$ = 5 MHz<br>$V_{DD}$ = 3.0 V ±10%[Note 5] | When A/D converter is stopped | | 3.0 | 6.0 | mA |
| | | | | When A/D converter is operating[Note 8] | | 4.5 | 9.0 | |
| | $I_{DD2}$ | Crystal/ceramic oscillation, external clock input HALT mode[Note 6] | $f_X$ = 10 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When peripheral functions are stopped | | 1.7 | 3.8 | mA |
| | | | | When peripheral functions are operating | | | 6.7 | |
| | | | $f_X$ = 6 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When peripheral functions are stopped | | 1.3 | 3.0 | mA |
| | | | | When peripheral functions are operating | | | 6.0 | |
| | | | $f_X$ = 5 MHz<br>$V_{DD}$ = 3.0 V ±10%[Note 5] | When peripheral functions are stopped | | 0.48 | 1 | mA |
| | | | | When peripheral functions are operating | | | 2.1 | |
| | $I_{DD3}$ [Note 3] | High-speed Ring-OSC operating mode[Note 7] | $f_X$ = 8 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When A/D converter is stopped | | 5.5 | 11.0 | mA |
| | | | | When A/D converter is operating[Note 8] | | 7.0 | 14.0 | |
| | $I_{DD4}$ | High-speed Ring-OSC HALT mode[Note 7] | $f_X$ = 8 MHz<br>$V_{DD}$ = 5.0 V ±10%[Note 4] | When peripheral functions are stopped | | 1.4 | 3.2 | mA |
| | | | | When peripheral functions are operating | | | 5.9 | |
| | $I_{DD5}$ | STOP mode | $V_{DD}$ = 5.0 V ±10% | When low-speed Ring-OSC is stopped | | 3.5 | 35.5 | μA |
| | | | | When low-speed Ring-OSC is operating | | 17.5 | 63.5 | |
| | | | $V_{DD}$ = 3.0 V ±10% | When low-speed Ring-OSC is stopped | | 3.5 | 15.5 | μA |
| | | | | When low-speed Ring-OSC is operating | | 11.0 | 30.5 | |

**Notes 1.** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

**2.** Total current flowing through the internal power supply ($V_{DD}$). Peripheral operation current is included (however, the current that flows through the pull-up resistors of ports is not included).

**3.** Peripheral operation current is included.

**4.** When the processor clock control register (PCC) is set to 00H.

**5.** When the processor clock control register (PCC) is set to 02H.

**6.** When crystal/ceramic oscillation clock, external clock input is selected as the system clock source using the option byte.

**7.** When the high-speed Ring-OSC is selected as the system clock source using the option byte.

**8.** The current that flows through the $AV_{REF}$ pin is included.

## AC Characteristics

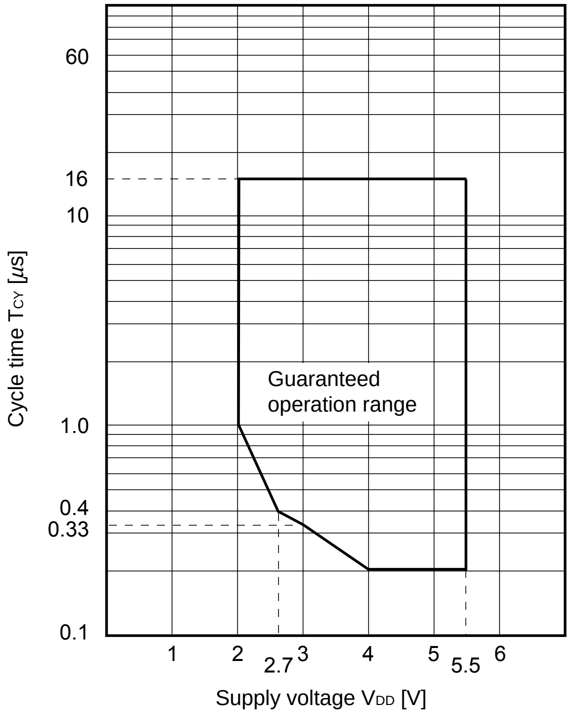
### (1) Basic operation ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V[Note 1], $V_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Cycle time (minimum instruction execution time) | $T_{CY}$ | Crystal/ceramic oscillation clock, external clock input | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V | 0.2 | | 16 | μs |
| | | | 3.0 V ≤ $V_{DD}$ < 4.0 V | 0.33 | | 16 | μs |
| | | | 2.7 V ≤ $V_{DD}$ < 3.0 V | 0.4 | | 16 | μs |
| | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1 | | 16 | μs |
| | | High-speed Ring-OSC clock | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V | 0.23 | | 4.22 | μs |
| | | | 2.7 V ≤ $V_{DD}$ < 4.0 V | 0.47 | | 4.22 | μs |
| | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 0.95 | | 4.22 | μs |
| TI000 input high-level width, low-level width | $t_{TIH}$,<br>$t_{TIL}$ | 4.0 V ≤ $V_{DD}$ ≤ 5.5 V | | 2/fsam+ 0.1[Note 2] | | | μs |
| | | 2.0 V ≤ $V_{DD}$ < 4.0 V | | 2/fsam+ 0.2[Note 2] | | | μs |
| Interrupt input high-level width, low-level width | $t_{INTH}$,<br>$t_{INTL}$ | | | 1 | | | μs |
| $\overline{\text{RESET}}$ input low-level width | $t_{RSL}$ | | | 2 | | | μs |

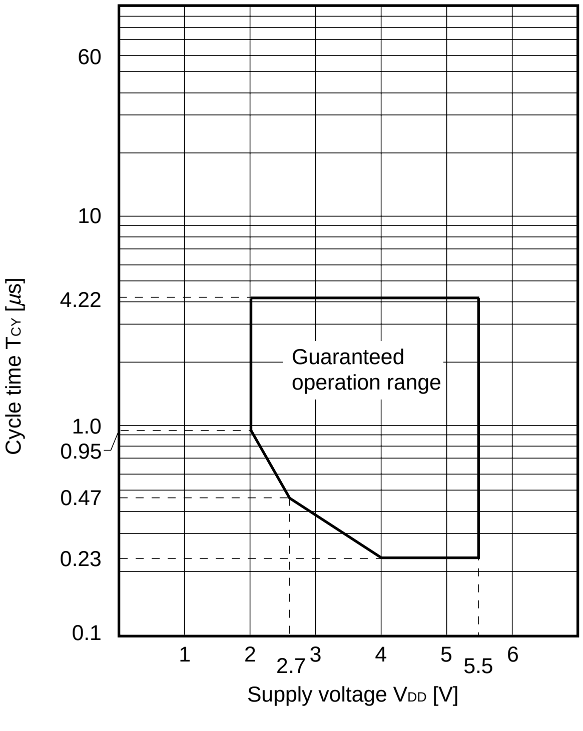
**Notes 1.** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

**2.** Selection of fsam = $f_{XP}$, $f_{XP}$/4, or $f_{XP}$/256 is possible using bits 0 and 1 (PRM000, PRM001) of prescaler mode register 00 (PRM00). Note that when selecting the TI000 valid edge as the count clock, fsam = $f_{XP}$.

### TCY vs. VDD (Crystal/Ceramic Oscillation Clock, External Clock Input)



### TCY vs. VDD (High-speed Ring-OSC Clock)



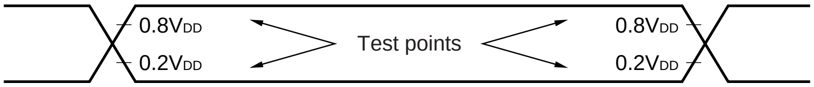
### (2) Serial interface ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.0 to 5.5 V$^{Note}$, $V_{SS}$ = 0 V)

**UART mode (UART6, dedicated baud rate generator output)**

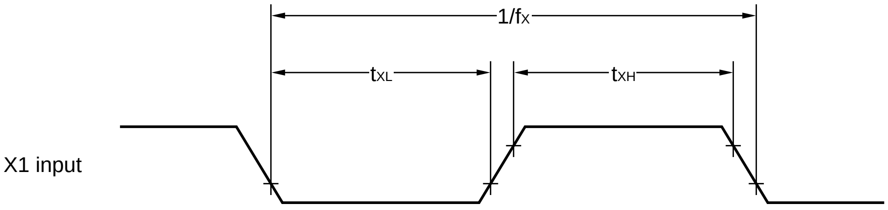
| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Transfer rate | | | | | 312.5 | kbps |

**Note** Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage ($V_{POC}$) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

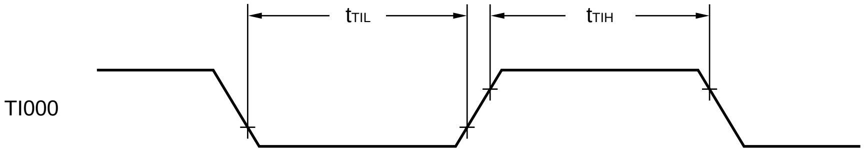
**AC Timing Test Points (Excluding X1 Input)**



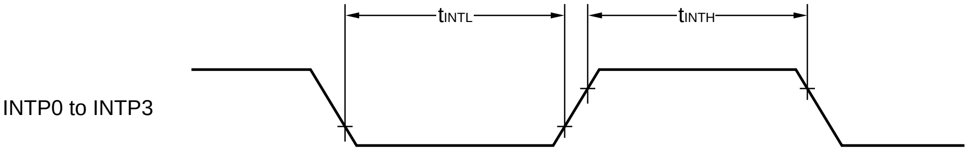
**Clock Timing**



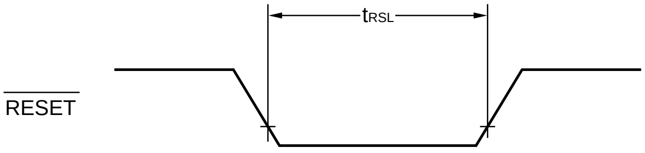
**TI000 Timing**



**Interrupt Input Timing**



**$\overline{\text{RESET}}$ Input Timing**



**A/D Converter Characteristics ($T_A$ = –40 to +85°C, 2.7 V ≤ $AV_{REF}$ ≤ $V_{DD}$ ≤ 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Resolution | | | 10 | 10 | 10 | bit |
| Overall error[Notes 1, 2] | AINL | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | ±0.2 | ±0.4 | %FSR |
| | | 2.7 V ≤ $AV_{REF}$ < 4.0 V | | ±0.3 | ±0.6 | %FSR |
| Conversion time | $t_{CONV}$ | 4.5 V ≤ $AV_{REF}$ ≤ 5.5 V | 3.0 | | 100 | μs |
| | | 4.0 V ≤ $AV_{REF}$ < 4.5 V | 4.8 | | 100 | μs |
| | | 2.85 V ≤ $AV_{REF}$ < 4.0 V | 6.0 | | 100 | μs |
| | | 2.7 V ≤ $AV_{REF}$ < 2.85 V | 14.0 | | 100 | μs |
| Zero-scale error[Notes 1, 2] | Ezs | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | ±0.4 | %FSR |
| | | 2.7 V ≤ $AV_{REF}$ < 4.0 V | | | ±0.6 | %FSR |
| Full-scale error[Notes 1, 2] | Efs | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | ±0.4 | %FSR |
| | | 2.7 V ≤ $AV_{REF}$ < 4.0 V | | | ±0.6 | %FSR |
| Integral non-linearity error[Note 1] | ILE | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | ±2.5 | LSB |
| | | 2.7 V ≤ $AV_{REF}$ < 4.0 V | | | ±4.5 | LSB |
| Differential non-linearity error[Note 1] | DLE | 4.0 V ≤ $AV_{REF}$ ≤ 5.5 V | | | ±1.5 | LSB |
| | | 2.7 V ≤ $AV_{REF}$ < 4.0 V | | | ±2.0 | LSB |
| Analog input voltage | $V_{AIN}$ | | $V_{SS}$ | | $AV_{REF}$ | V |

**Notes 1.** Excludes quantization error (±1/2 LSB).

**2.** This value is indicated as a ratio (%FSR) to the full-scale value.

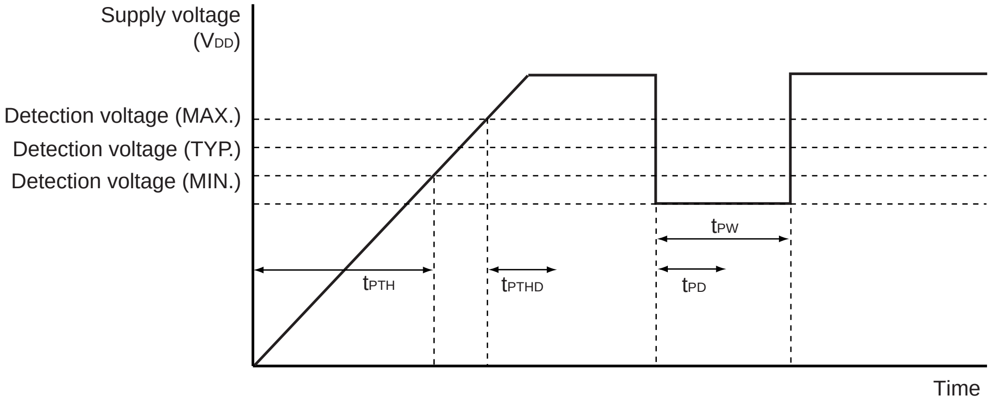
### POC Circuit Characteristics ($T_A$ = –40 to +85°C)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Detection voltage | $V_{POC}$ | | 2.0 | 2.1 | 2.2 | V |
| Power supply boot time | $t_{PTH}$ | $V_{DD}$: 0 V → 2.1 V | 1.5 | | | μs |
| Response delay time 1[Note 1] | $t_{PTHD}$ | When power supply rises, after reaching detection voltage (MAX.) | | | 3.0 | ms |
| Response delay time 2[Note 2] | $t_{PD}$ | When power supply falls | | | 1.0 | ms |
| Minimum pulse width | $t_{PW}$ | | 0.2 | | | ms |

**Notes 1.** Time required from voltage detection to internal reset release.

**2.** Time required from voltage detection to internal reset signal generation.

### POC Circuit Timing



### LVI Circuit Characteristics ($T_A$ = –40 to +85°C)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Detection voltage | $V_{LVI0}$ | | 4.1 | 4.3 | 4.5 | V |
| | $V_{LVI1}$ | | 3.9 | 4.1 | 4.3 | V |
| | $V_{LVI2}$ | | 3.7 | 3.9 | 4.1 | V |
| | $V_{LVI3}$ | | 3.5 | 3.7 | 3.9 | V |
| | $V_{LVI4}$ | | 3.3 | 3.5 | 3.7 | V |
| | $V_{LVI5}$ | | 3.15 | 3.3 | 3.45 | V |
| | $V_{LVI6}$ | | 2.95 | 3.1 | 3.25 | V |
| | $V_{LVI7}$ | | 2.7 | 2.85 | 3.0 | V |
| | $V_{LVI8}$ | | 2.5 | 2.6 | 2.7 | V |
| | $V_{LVI9}$ | | 2.25 | 2.35 | 2.45 | V |
| Response time[Note 1] | $t_{LD}$ | | | 0.2 | 2.0 | ms |
| Minimum pulse width | $t_{LW}$ | | 0.2 | | | ms |
| Operation stabilization wait time[Note 2] | $t_{LWAIT}$ | | | 0.1 | 0.2 | ms |

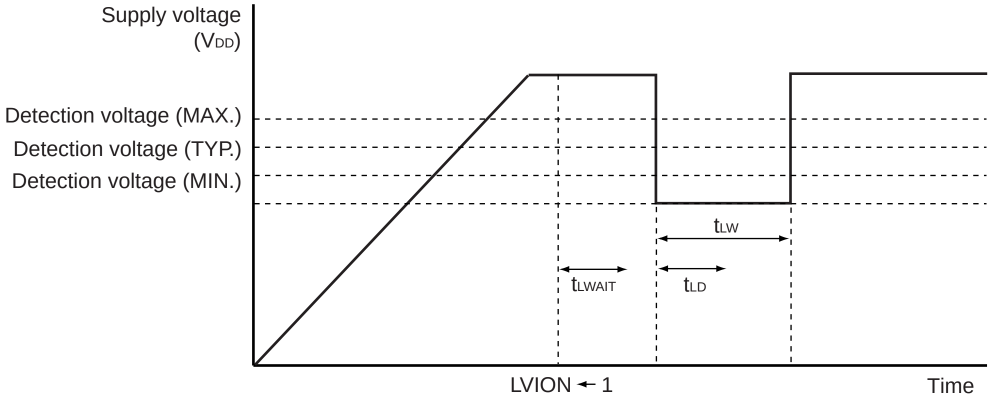
**Notes 1.** Time required from voltage detection to interrupt output or internal reset signal generation.

**2.** Time required from setting LVION to 1 to operation stabilization.

**Remarks 1.** $V_{LVI0} > V_{LVI1} > V_{LVI2} > V_{LVI3} > V_{LVI4} > V_{LVI5} > V_{LVI6} > V_{LVI7} > V_{LVI8} > V_{LVI9}$

**2.** $V_{POC} < V_{LVIm}$ (m = 0 to 9)

### LVI Circuit Timing



### Data Memory STOP Mode Low Supply Voltage Data Retention Characteristics ($T_A$ = –40 to +85°C)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention supply voltage | $V_{DDDR}$ | | 2.0 | | 5.5 | V |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |

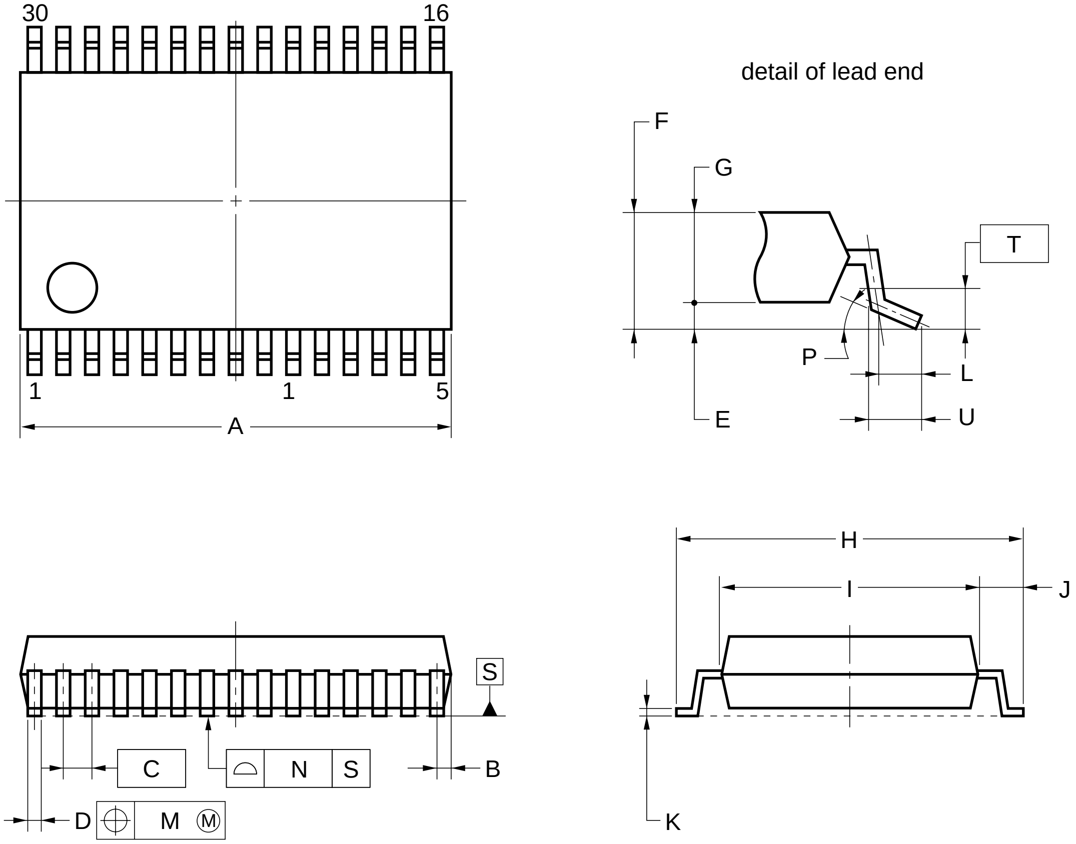
**Flash Memory Programming Characteristics ($T_A$ = –40 to +85°C, 2.7 V ≤ $V_{DD}$ ≤ 5.5 V, $V_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Supply current | $I_{DD}$ | $V_{DD}$ = 5.5 V | | | | 7.0 | mA |
| Erasure count[Note]<br>(per 1 block) | $N_{ERASE}$ | $T_A$ = –10 to +85°C | | | | 1000 | Times |
| | | $T_A$ = –40 to +85°C | | | | T.B.D. | Times |
| Chip erase time | $T_{CERASE}$ | $T_A$ = –10 to +85°C,<br>$N_{ERASE}$ ≤ 100 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.90 | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | 1.00 | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | 1.20 | s |
| | | $T_A$ = –10 to +85°C,<br>$N_{ERASE}$ ≤ 1000 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 3.52 | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | 3.92 | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | 4.69 | s |
| | | $T_A$ = –40 to +85°C,<br>$N_{ERASE}$ ≤ 100 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | T.B.D. | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | T.B.D. | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | T.B.D. | s |
| | | $T_A$ = –40 to +85°C,<br>$N_{ERASE}$ ≤ 1000 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | T.B.D. | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | T.B.D. | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | T.B.D. | s |
| Block erase time | $T_{BERASE}$ | $T_A$ = –10 to +85°C,<br>$N_{ERASE}$ ≤ 100 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.48 | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | 0.53 | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | 0.63 | s |
| | | $T_A$ = –10 to +85°C,<br>$N_{ERASE}$ ≤ 1000 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 1.86 | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | 2.07 | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | 2.48 | s |
| | | $T_A$ = –40 to +85°C,<br>$N_{ERASE}$ ≤ 100 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | T.B.D. | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | T.B.D. | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | T.B.D. | s |
| | | $T_A$ = –40 to +85°C,<br>$N_{ERASE}$ ≤ 1000 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | T.B.D. | s |
| | | | 3.5 V ≤ $V_{DD}$ < 4.5 V | | | T.B.D. | s |
| | | | 2.7 V ≤ $V_{DD}$ < 3.5 V | | | T.B.D. | s |
| Byte write time | $T_{WRITE}$ | $T_A$ = –10 to +85°C, $N_{ERASE}$ ≤ 1000 | | | | 150 | μs |
| | | $T_A$ = –40 to +85°C, $N_{ERASE}$ ≤ 1000 | | | | T.B.D. | μs |
| Internal verify | $T_{VERIFY}$ | Per 1 block | | | | 6.8 | ms |
| | | Per 1 byte | | | | 27 | μs |
| Blank check | $T_{BLKCHK}$ | Per 1 block | | | | 480 | μs |
| Retention years | | $T_A$ = –10 to +85°C, $N_{ERASE}$ ≤ 1000 | | 10 | | | Years |
| | | $T_A$ = –40 to +85°C, $N_{ERASE}$ ≤ 1000 | | T.B.D. | | | Years |

**Note** Depending on the erasure count ($N_{ERASE}$), the erase time varies. Refer to the chip erase time and block erase time parameters.

**Remark** When a product is first written after shipment, “erase → write” and “write only” are both taken as one rewrite.

## 30-PIN PLASTIC SSOP (7.62 mm (300))



**NOTE**

Each lead centerline is located within 0.13 mm of
its true position (T.P.) at maximum material condition.

| ITEM | MILLIMETERS |
|---|---|
| A | 9.85±0.15 |
| B | 0.45 MAX. |
| C | 0.65 (T.P.) |
| D | $0.24^{+0.08}_{-0.07}$ |
| E | 0.1±0.05 |
| F | 1.3±0.1 |
| G | 1.2 |
| H | 8.1±0.2 |
| I | 6.1±0.2 |
| J | 1.0±0.2 |
| K | 0.17±0.03 |
| L | 0.5 |
| M | 0.13 |
| N | 0.10 |
| P | $3°^{+5°}_{-3°}$ |
| T | 0.25 |
| U | 0.6±0.15 |

**S30MC-65-5A4-2**

# APPENDIX A DEVELOPMENT TOOLS

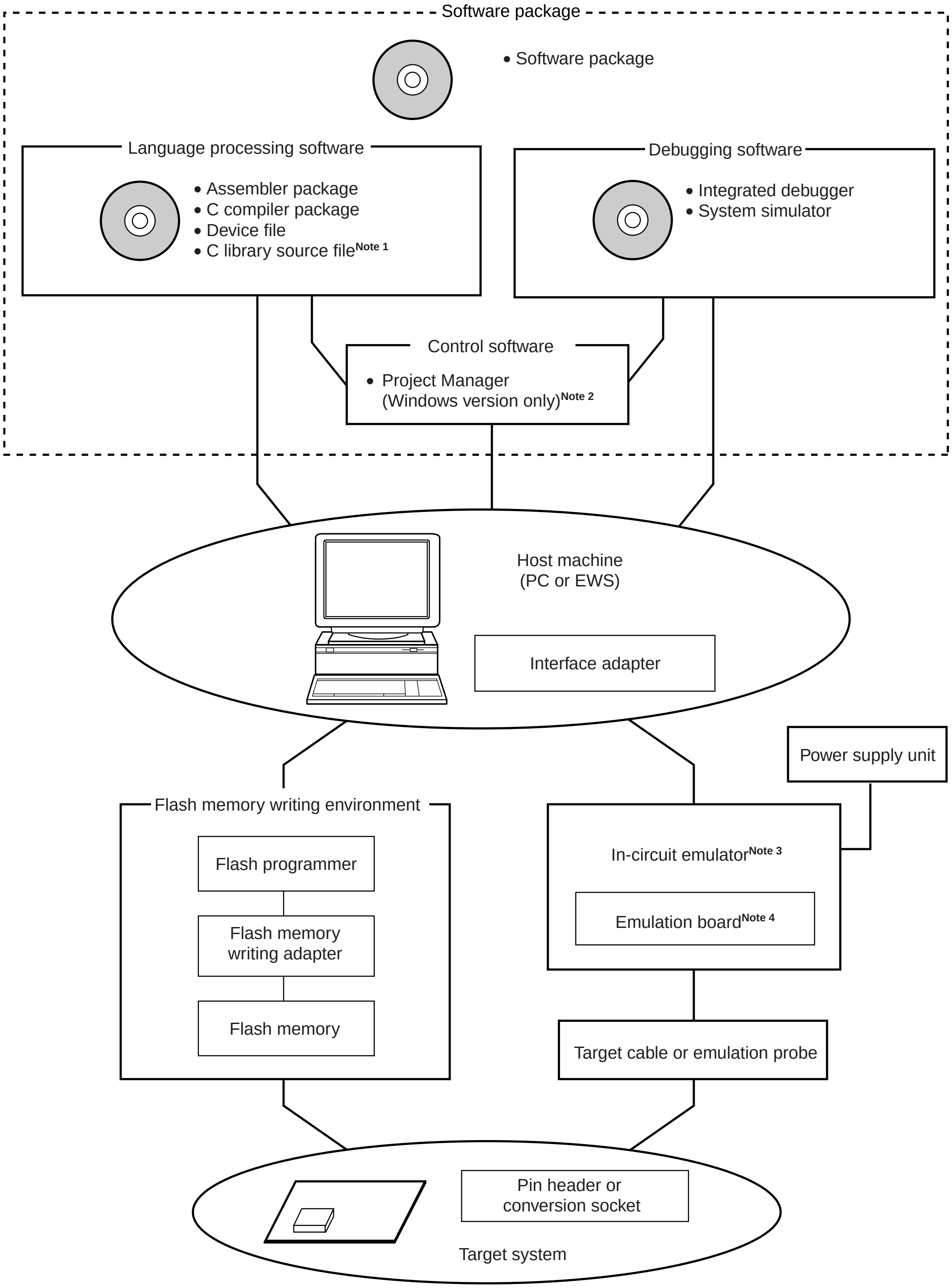
The following development tools are available for development of systems using the 78K0S/KB1+. Figure A-1 shows development tools.

- Compatibility with PC98-NX series
  Unless stated otherwise, products which are supported by IBM PC/AT[TM] and compatibles can also be used with the PC98-NX series. When using the PC98-NX series, therefore, refer to the explanations for IBM PC/AT and compatibles.

- Windows[TM]
  Unless stated otherwise, "Windows" refers to the following operating systems.
  - Windows 98
  - Windows NT[TM] Ver. 4.0
  - Windows 2000
  - Windows XP

### Figure A-1. Development Tools (1/2)

**(1) When using the in-circuit emulator IE-78K0S-NS or IE-78K0S-NS-A**



**Notes 1.** The C library source file is not included in the software package.

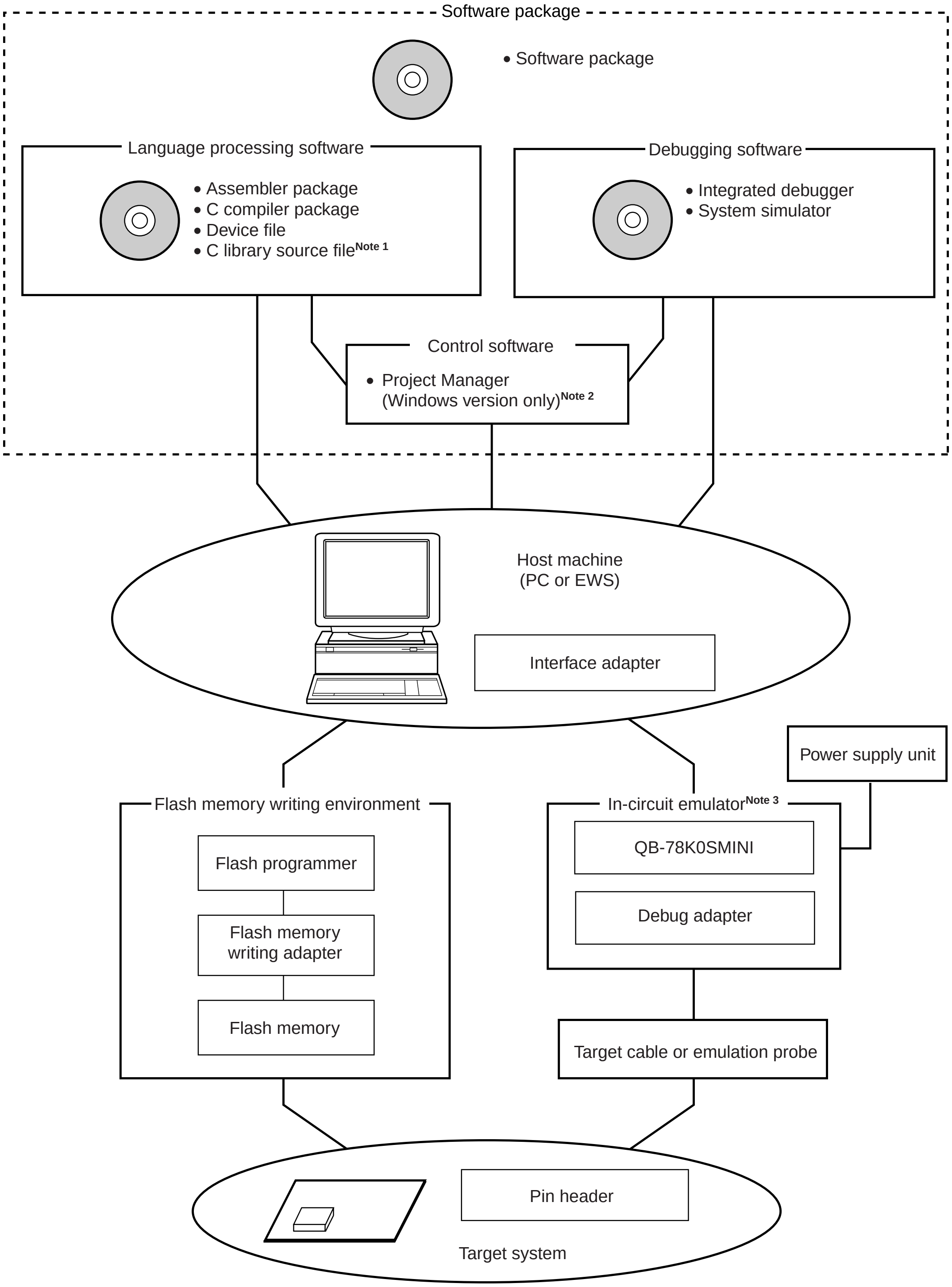
**2.** The Project Manager PM plus is included in the assembler package.
PM plus is used only in the Windows environment.

**3.** All products other than the in-circuit emulators IE-78K0S-NS and IE-78K0S-NS-A are optional.

**4.** The in-circuit emulator IE-789234-NS-EM1 is provided with the target cable.

**Figure A-1. Development Tools (2/2)**

**(2) When using the in-circuit emulator QB-78K0SKX1MINI**



**Notes 1.** The C library source file is not included in the software package.

**2.** The Project Manager PM plus is included in the assembler package.
PM plus is used only in the Windows environment.

**3.** The in-circuit emulator QB-78K0SKX1MINI is provided with the integrated debugger ID78K0S-QB, the flash memory programmer PG-FPL2, a power supply unit, and a target cable. Other products are optional.

## A.1 Software Package

| | |
|---|---|
| SP78K0S<br>Software package | This is a package that bundles the software tools required for development of the 78K/0S Series.<br>The following tools are included.<br>RA78K0S, CC78K0S, ID78K0S-NS, etc. |
| | Part number: μS××××SP78K0S |

**Remark** ×××× in the part number differs depending on the operating system to be used.

μS××××SP78K0S

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series, IBM PC/AT and compatibles | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |

## A.2 Language Processing Software

| | |
|---|---|
| RA78K0S<br>Assembler package | Program that converts program written in mnemonic into object code that can be executed by microcontroller.<br>In addition, automatic functions to generate symbol table and optimize branch instructions are also provided. Used in combination with device file (DF789234) (sold separately).<br>**<Caution when used in PC environment>**<br>The assembler package is a DOS-based application but may be used under the Windows environment by using PM plus (included in the assembler package). |
| | Part number: μS××××RA78K0S |
| CC78K0S<br>C library package | Program that converts program written in C language into object codes that can be executed by microcontroller.<br>Used in combination with assembler package (RA78K0S) and device file (DF789234) (both sold separately).<br>**<Caution when used in PC environment>**<br>The C compiler package is a DOS-based application but may be used under the Windows environment by using PM plus (included in the assembler package). |
| | Part number: μS××××CC78K0S |
| DF789234[Note 1]<br>Device file | File containing the information inherent to the device.<br>Used in combination with other tools (RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, or SM+ for 78K0S). |
| | Part number: μS××××DF789234 |
| CC78K0S-L[Note 2]<br>C library source file | Source file of functions constituting object library included in C compiler package.<br>Necessary for changing object library included in C compiler package according to customer's specifications.<br>Since this is the source file, its working environment does not depend on any particular operating system. |
| | Part number: μS××××CC78K0S-L |

**Notes 1.** DF789234 is a common file that can be used with RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, and SM+ for 78K0S.

**2.** CC78K0S-L is not included in the software package (SP78K0S).

**Remark** ×××× in the part number differs depending on the host machine and operating system to be used.

μS××××RA78K0S
μS××××CC78K0S
μS××××CC78K0S-L

| ×××× | Host Machine | OS | Supply Media |
|---|---|---|---|
| AB17 | PC-9800 series, IBM PC/AT and compatibles | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |
| 3P17 | HP9000 series 700™ | HP-UX™ (Rel. 10.10) | |
| 3K17 | SPARCstation™ | SunOS™ (Rel. 4.1.4),<br>Solaris™ (Rel. 2.5.1) | |

μS××××DF789234

| ×××× | Host Machine | OS | Supply Media |
|---|---|---|---|
| AB13 | PC-9800 series, IBM PC/AT and compatibles | Japanese Windows | 3.5” 2HD FD |
| BB13 | | English Windows | |

## A.3 Control Software

| | |
|---|---|
| PM plus<br>Project manager | This is control software designed so that the user program can be efficiently developed in the Windows environment. With this software, a series of user program development operations, including starting the editor, build, and starting the debugger, can be executed on PM plus.<br>**<Caution>**<br>PM plus is included in the assembler package (RA78K0S). It can be used only in the Windows environment. |

## A.4 Flash Memory Writing Tools

| | |
|---|---|
| Flashpro4 (FL-PR4, PG-FP4)<br>Flash memory programmer | Flash programmer dedicated to the microcontrollers incorporating a flash memory |
| PG-FPL2<br>Flash memory programmer | Flash programmer dedicated to the microcontrollers incorporating a flash memory<br>Provided with the in-circuit emulator QB-78K0SKX1MINI. |
| FA-30MC-5A4-A<br>Flash memory writing adapter | Flash memory writing adapter. Used in connection with Flash programmer.<br>Designed for use with a 30-pin plastic SSOP (MC-5A4 type). |

**Remark** FL-PR4 and FA-30MC-5A4-A are products of Naito Densei Machida Mfg. Co., Ltd.
For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (TEL +81-45-475-4191)

## A.5 Debugging Tools (Hardware)

### A.5.1 When using in-circuit emulator IE-78K0S-NS or IE-78K0S-NS-A

| | | |
|---|---|---|
| IE-78K0S-NS<br>In-circuit emulator | | In-circuit emulator for debugging hardware and software of application system using 78K/0S Series. Supports integrated debugger (ID78K0S-NS). Used in combination with AC adapter, emulation probe, and interface adapter for connecting the host machine. |
| IE-78K0S-NS-A<br>In-circuit emulator | | This in-circuit emulator has a coverage function in addition to the functions of the IE-78K0S-NS, and enhanced debugging functions such as an enhanced tracer function and timer function. |
| IE-70000-MC-PS-B<br>AC adapter | | Adapter for supplying power from 100 to 240 VAC outlet. |
| IE-70000-CD-IF-A<br>PC card interface | | PC card and interface cable required when using a notebook type PC as the host machine (PCMCIA socket supported). |
| IE-70000-PC-IF-C<br>Interface adapter | | Adapter required when using IBM PC/AT and compatibles as the host machine (ISA bus supported). |
| IE-70000-PCI-IF-A<br>Interface adapter | | Adapter required when using a personal computer incorporating the PCI bus is used as the host machine. |
| IE-789234-NS-EM1<br>Emulation board | | Emulation board for emulating the peripheral hardware inherent to the device.<br>Used in combination with in-circuit emulator. A target cable is provided. |
| NP-30MC<br>Emulation probe | | This probe is used to connect the in-circuit emulator to the target system and is designed for use with a 30-pin plastic SSOP (MC-5A4 type). |
| | NSPACK30BK<br>YSPACK30BK<br>Conversion connector | This conversion connector connects the NP-30MC to a target system board designed to mount a 30-pin plastic SSOP (MC-5A4 type).<br>• NSPACK30BK: Connector for connecting target<br>• YSPACK30BK: Connector for connecting emulator |
| QB-80-EP-01T<br>Emulation probe | | This emulation probe is flexible type and used to connect the in-circuit emulator and target system. This probe is designed for use with a 30-pin plastic SSOP (MC-5A4 type). |
| | QB-30MC-EA-01T | This exchange adapter is used to perform pin conversion from the in-circuit emulator to target connector. |
| | QB-30MC-YQ-01T | This YQ connector is used to connect the target connector and exchange adapter. |
| | QB-30MC-NQ-01T | This target connector is used to mount on the target system. |
| Specifications of pin header on target system | | 0.635 mm × 0.635 mm (height: 6 mm) |

**Remarks 1.** NP-30MC is a product of Naito Densei Machida Mfg. Co., Ltd.
For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (TEL +81-45-475-4191)

**2.** NSPACK30BK and YSPACK30BK are products of TOKYO ELETECH CORPORATION.
For further information, contact Daimaru Kogyo Co., Ltd.
Tokyo Electronics Department (TEL: +81-3-3820-7112)
Osaka Electronics Department (TEL: +81-6-6244-6672)

### A.5.2 When using in-circuit emulator QB-78K0SKX1MINI

| | |
|---|---|
| QB-78K0SKX1MINI<br>In-circuit emulator | In-circuit emulator for debugging hardware and software of application system using 78K0S/Kx1+ Series. Supports integrated debugger (ID78K0S-QB). Used in combination with AC adapter, target cable, and USB interface cable for connecting the host machine. |
| Specifications of pin header on target system | 0.635 mm × 0.635 mm (height: 6 mm) |

## A.6 Debugging Tools (Software)

| | |
|---|---|
| ID78K0S-NS<br>(supporting in-circuit emulator IE-78K0S-NS/ IE-78K0S-NS-A)<br>Integrated debugger | This debugger supports the in-circuit emulators for the 78K/0S Series. ID78K0S-NS is Windows-based software.<br>This debugger has enhanced debugging functions supporting C language. By using its window integration function that associates the source program, disassemble display, and memory display with trace results, the trace results can be displayed corresponding to the source program.<br>It is used with a device file (DF789234) (sold separately). |
| | Ordering number: μS××××ID78K0S-NS |
| ID78K0S-QB<br>(supporting in-circuit emulator QB-78K0SKX1MINI)<br>Integrated debugger | This debugger supports the in-circuit emulators for the 78K0S/Kx1+ Series. ID78K0S-QB is Windows-based software.<br>Provided with the debug function supporting C language, source programming, disassemble display, and memory display are possible. This is used with the device file (DF789234) (sold separately).<br>It is provided with the in-circuit emulator QB-78K0SKX1MINI. |
| | Ordering number: μS××××ID78K0S-QB (not for sale) |
| SM+ for 78K0S<sup>Note 1</sup><br>System simulator | This is a system simulator for the 78K/0S series. SM+ for 78K0S is Windows-based software.<br>This simulator can execute C-source-level or assembler-level debugging while simulating the operations of the target system on the host machine.<br>By using SM+ for 78K0S, the logic and performance of the application can be verified independently of hardware development. Therefore, the development efficiency can be enhanced and the software quality can be improved.<br>This simulator is used with a device file (DF789234) (sold separately). |
| | Ordering number: μS××××SM789234-B |
| DF789234<sup>Note 2</sup><br>Device file | This is a file that has device-specific information.<br>It is used with the RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, and SM+ for 78K0S (all sold separately). |
| | Ordering numberμS××××DF789234 |

**Notes 1.** Under development

**2.** DF789234 is a common file that can be used with the RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, and SM+ for 78K0S.

**Remark** ×××× in the part number differs depending on the operating system to be used and the supply medium.

μS××××ID78K0S-NS
μS××××ID78K0S-QB
μS××××SM789234-B

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| BB13 | PC-9800 series, IBM PC/AT and compatibles | English Windows | 3.5” 2HD FD |
| AB17 | | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |

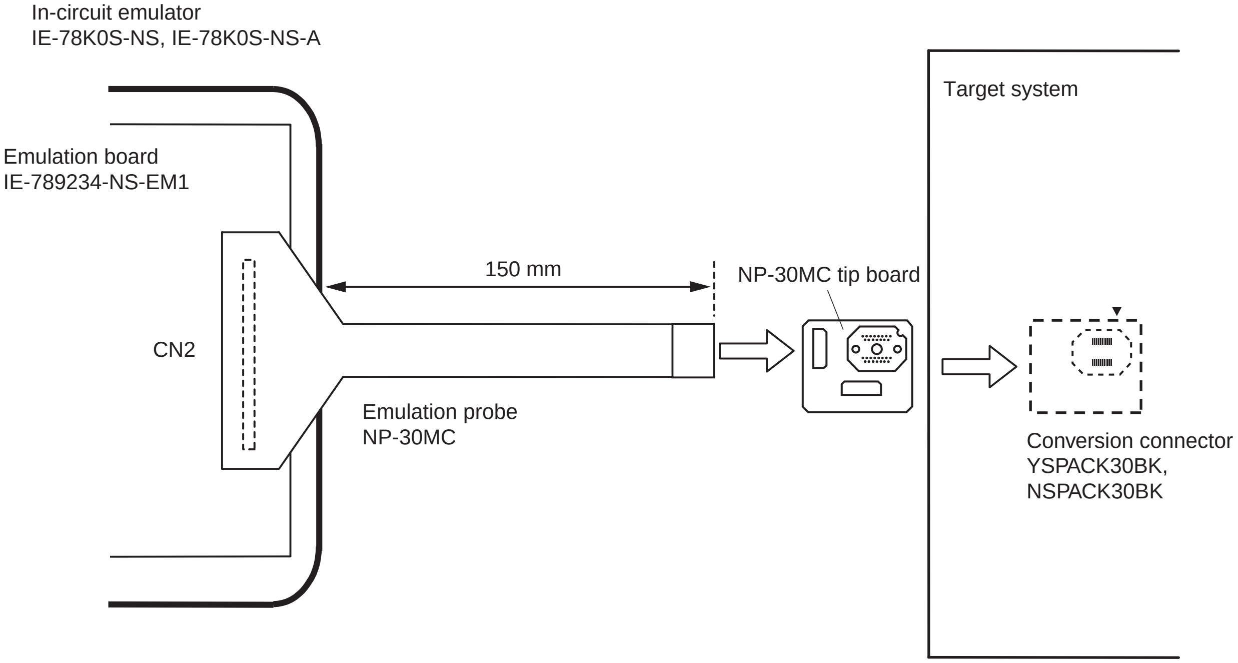
μS××××DF789234

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB13 | PC-9800 series, IBM PC/AT and compatibles | Japanese Windows | 3.5” 2HD FD |
| BB13 | | English Windows | |

# APPENDIX B NOTES ON TARGET SYSTEM DESIGN

The following show the conditions when connecting the emulation probe to the conversion connector and conversion socket in the case using in-circuit emulator IE-78K0S-NS or IE-78K0S-NS-A. Follow the configuration below and consider the shape of parts to be mounted on the target system when designing a system.

**Figure B-1. Distance Between In-Circuit Emulator and Conversion Connector (When Using NP-30MC)**



**Remarks 1.** The NP-30MC is a product made by Naito Densei Machida Mfg. Co., Ltd.

**2.** The YSPACK30BK and NSPACK30BK are products by TOKYO ELETECH CORPORATION.

**Figure B-2. Condition for Connecting Target System (When Using NP-30MC)**

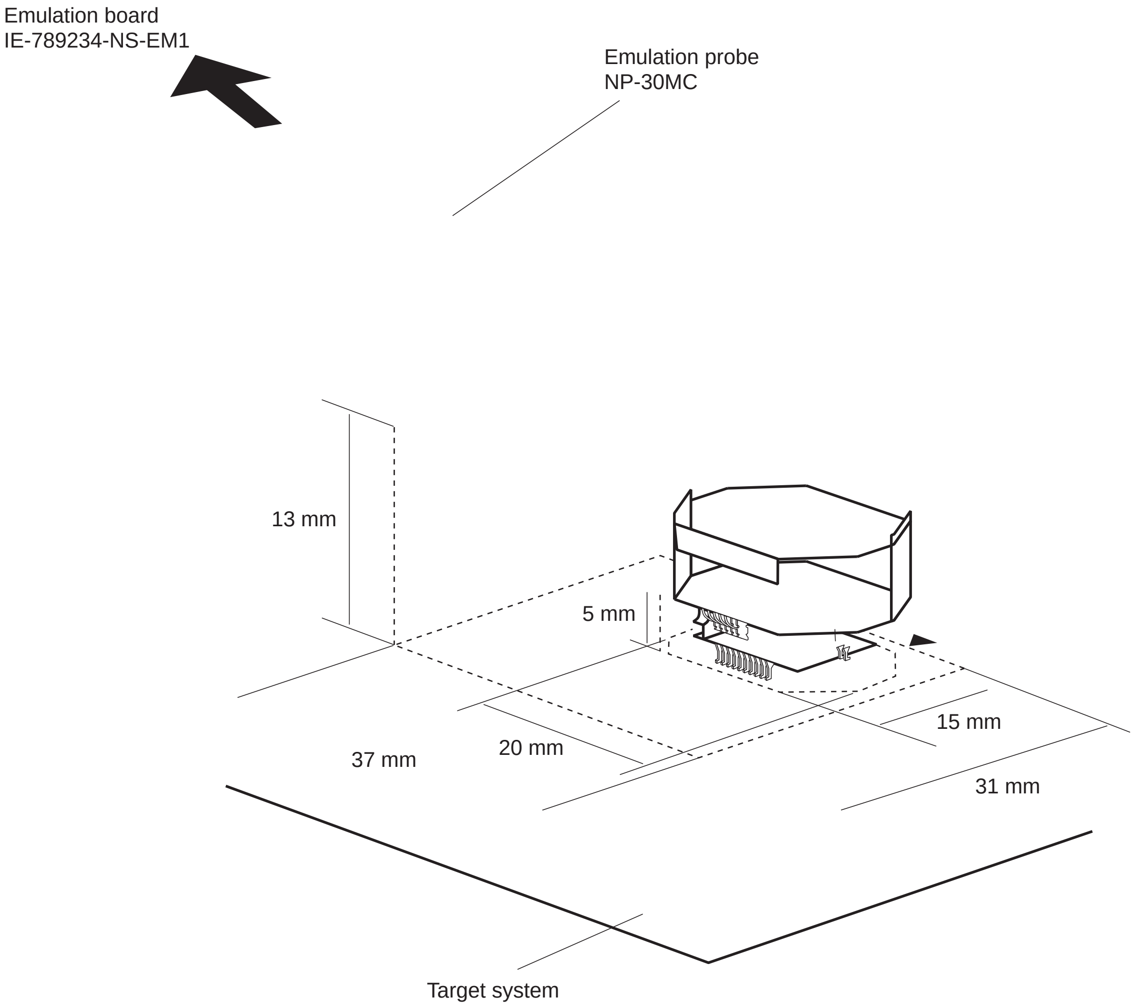


**Figure B-3. Distance Between In-Circuit Emulator and Conversion Connector (When Using QB-80-EP-01T)**

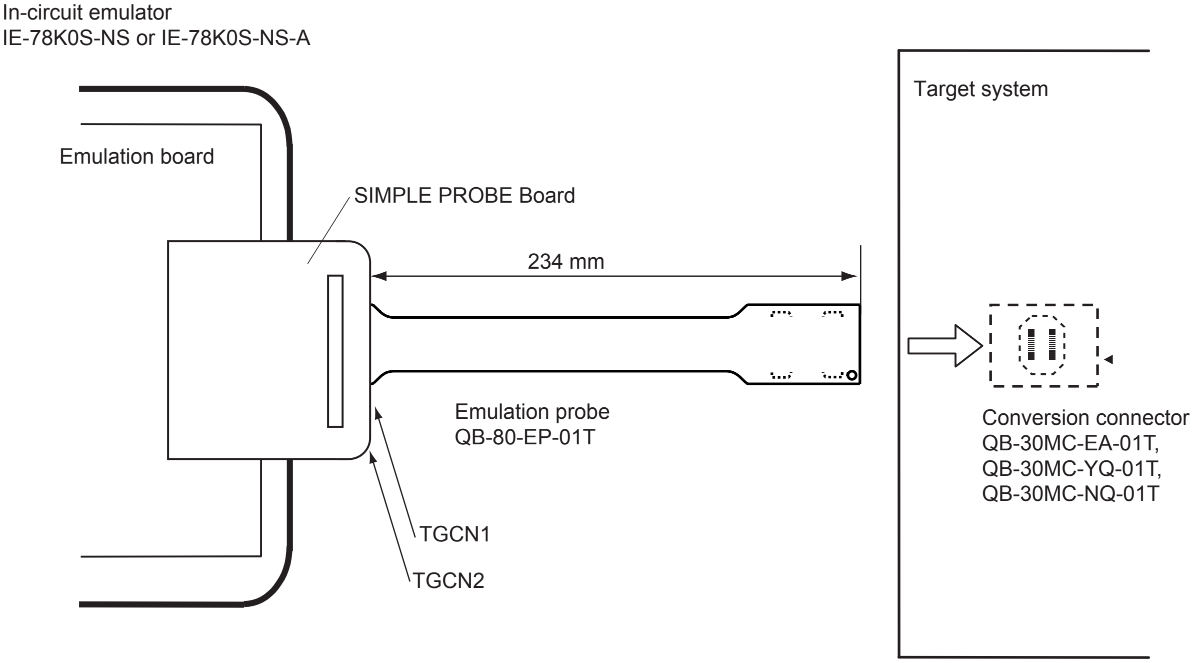
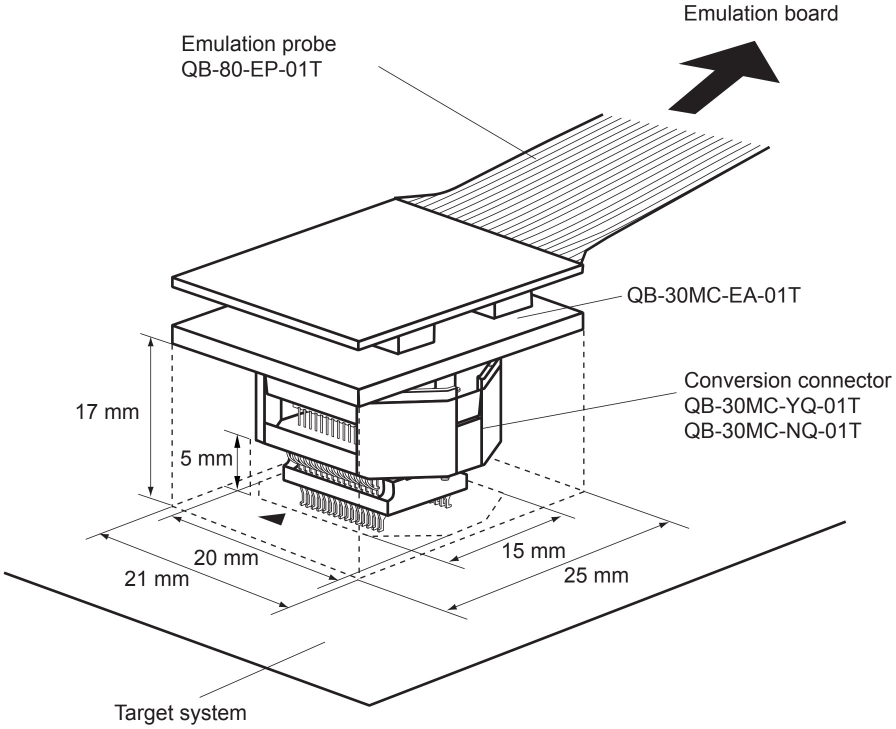


**Figure B-4. Condition for Connecting Target System (When Using QB-80-EP-01T)**



# APPENDIX C REGISTER INDEX

## C.1 Register Index (Register Name)

8-bit A/D conversion result register (ADCRH) … 168
8-bit compare register 80 (CR80) … 128
8-bit timer counter 80 (TM80) … 128
8-bit timer H compare register 01 (CMP01) … 135
8-bit timer H compare register 11 (CMP11) … 135
8-bit timer H mode register 1 (TMHMD1) … 136
8-bit timer mode control register 80 (TMC80) … 129
10-bit A/D conversion result register (ADCR) … 167
16-bit timer capture/compare register 000 (CR000) … 87
16-bit timer capture/compare register 010 (CR010) … 89
16-bit timer counter 00 (TM00) … 87
16-bit timer mode control register 00 (TMC00) … 90
16-bit timer output control register 00 (TOC00) … 93
16-bit multiplication result storage register H (MUL0H) … 37, 217
16-bit multiplication result storage register L (MUL0L) … 37, 217

**[A]**
A/D converter mode register (ADM) … 164
Analog input channel specification register (ADS) … 167
Asynchronous serial interface control register 6 (ASICL6) … 192
Asynchronous serial interface operation mode register 6 (ASIM6) … 186
Asynchronous serial interface reception error status register 6 (ASIS6) … 188
Asynchronous serial interface transmission status register 6 (ASIF6) … 189

**[B]**
Baud rate generator control register 6 (BRGC6) … 191

**[C]**
Capture/compare control register 00 (CRC00) … 92
Clock selection register 6 (CKSR6) … 190

**[E]**
External interrupt mode register 0 (INTM0) … 227
External interrupt mode register 1 (INTM1) … 228

**[F]**
Flash address pointer H (FLAPH) … 288
Flash address pointer L (FLAPL) … 288
Flash address pointer H compare register (FLAPHC) … 288
Flash address pointer L compare register (FLAPLC) … 288
Flash programming command register (FLCMD) … 287
Flash programming mode control register (FLPMC) … 283
Flash protect command register (PFCMD) … 285

Flash status register (PFS) … 286
Flash write buffer register (FLW) … 289

**[I]**
Input switch control register (ISC) … 194
Interrupt mask flag register 0 (MK0) … 226
Interrupt mask flag register 1 (MK1) … 226
Interrupt request flag register 0 (IF0) … 225
Interrupt request flag register 1 (IF1) … 225

**[L]**
Low voltage detect register (LVIM) … 257
Low voltage detection level select register (LVIS) … 258
Low-speed Ring-OSC mode register (LSRCM) … 73

**[M]**
Multiplication data register A (MRA0) … 217
Multiplication data register B (MRB0) … 217
Multiplier control register 0 (MULC0) … 219

**[O]**
Oscillation stabilization time select register (OSTS) … 74, 236

**[P]**
Port mode control register 2 (PMC2) … 66, 168
Port mode register 0 (PM0) … 64
Port mode register 2 (PM2) … 64, 168
Port mode register 3 (PM3) … 64, 95
Port mode register 4 (PM4) … 64, 138, 194
Port mode register 12 (PM12) … 64
Port register 0 (P0) … 65
Port register 2 (P2) … 65
Port register 3 (P3) … 65
Port register 4 (P4) … 65
Port register 12 (P12) … 65
Port register 13 (P13) … 65
Preprocessor clock control register (PPCC) … 72
Prescaler mode register 00 (PRM00) … 94
Processor clock control register (PCC) … 72
Pull-up resistor option register 0 (PU0) … 67
Pull-up resistor option register 2 (PU2) … 67
Pull-up resistor option register 3 (PU3) … 67
Pull-up resistor option register 4 (PU4) … 67
Pull-up resistor option register 12 (PU12) … 67

**[R]**
Receive buffer register 6 (RXB6) … 185
Receive shift register 6 (RXS6) … 185

Reset control flag register (RESF) … 251

[T]

Transmit buffer register 6 (TXB6) … 185

Transmit shift register 6 (TXS6) … 185

[W]

Watchdog timer enable register (WDTE) … 152

Watchdog timer mode register (WDTM) … 151

## C.2 Register Index (Symbol)

**[A]**

ADCR: 10-bit A/D conversion result register … 167
ADCRH: 8-bit A/D conversion result register … 168
ADM: A/D converter mode register … 164
ADS: Analog input channel specification register … 167
ASICL6: Asynchronous serial interface control register 6 … 192
ASIF6: Asynchronous serial interface transmission status register 6 … 189
ASIM6: Asynchronous serial interface operation mode register 6 … 186
ASIS6: Asynchronous serial interface reception error status register 6 … 188

**[B]**

BRGC6: Baud rate generator control register 6 … 191

**[C]**

CKSR6: Clock selection register 6 … 190
CMP01: 8-bit timer H compare register 01 … 135
CMP11: 8-bit timer H compare register 11 … 135
CR000: 16-bit timer capture/compare register 000 … 87
CR010: 16-bit timer capture/compare register 010 … 89
CR80: 8-bit compare register 80 … 128
CRC00: Capture/compare control register 00 … 92

**[F]**

FLAPH: Flash address pointer H … 288
FLAPHC: Flash address pointer H compare register … 288
FLAPL: Flash address pointer L … 288
FLAPLC: Flash address pointer L compare register … 288
FLCMD: Flash programming command register … 287
FLPMC: Flash programming mode control register … 283
FLW: Flash write buffer register … 289

**[I]**

IF0: Interrupt request flag register 0 … 225
IF1: Interrupt request flag register 1 … 225
INTM0: External interrupt mode register 0 … 227
INTM1: External interrupt mode register 1 … 228
ISC: Input switch control register … 194

**[L]**

LSRCM: Low-speed Ring-OSC mode register … 73
LVIM: Low voltage detect register … 257
LVIS: Low voltage detection level select register … 258

**[M]**

MK0: Interrupt mask flag register 0 … 226
MK1: Interrupt mask flag register 1 … 226
MRA0: Multiplication data register A … 217
MRB0: Multiplication data register B … 217
MUL0H: 16-bit multiplication result storage register H … 37, 217
MUL0L: 16-bit multiplication result storage register L … 37, 217
MULC0: Multiplier control register 0 … 219

**[O]**

OSTS: Oscillation stabilization time select register … 74, 236

**[P]**

P0: Port register 0 … 65
P2: Port register 2 … 65
P3: Port register 3 … 65
P4: Port register 4 … 65
P12: Port register 12 … 65
P13: Port register 13 … 65
PCC: Processor clock control register … 72
PFCMD: Flash protect command register … 285
PFS: Flash status register … 286
PM0: Port mode register 0 … 64
PM2: Port mode register 2 … 64, 168
PM3: Port mode register 3 … 64, 95
PM4: Port mode register 4 … 64, 138, 194
PM12: Port mode register 12 … 64
PMC2: Port mode control register 2 … 66, 168
PPCC: Preprocessor clock control register … 72
PRM00: Prescaler mode register 00 … 94
PU0: Pull-up resistor option register 0 … 67
PU2: Pull-up resistor option register 2 … 67
PU3: Pull-up resistor option register 3 … 67
PU4: Pull-up resistor option register 4 … 67
PU12: Pull-up resistor option register 12 … 67

**[R]**

RESF: Reset control flag register … 251
RXB6: Receive buffer register 6 … 185
RXS6: Receive shift register 6 … 185

**[T]**

TM00: 16-bit timer counter 00 … 87
TM80: 8-bit timer counter 80 … 128
TMC00: 16-bit timer mode control register 00 … 90
TMC80: 8-bit timer mode control register 80 … 129
TMHMD1: 8-bit timer H mode register 1 … 136
TOC00: 16-bit timer output control register 00 … 93

TXB6: Transmit buffer register 6 … 185

TXS6: Transmit shift register 6 … 185

**[W]**

WDTE: Watchdog timer enable register … 152

WDTM: Watchdog timer mode register … 151

# APPENDIX D LIST OF CAUTIONS

This appendix lists cautions described in this document.
"Classification (hard/soft)" in table is as follows.

Hard: Cautions for microcontroller internal/external hardware
Soft: Cautions for software such as register settings or programs

(1/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 1 | Hard | Pin Configuration | AV$_{SS}$ pin | Connect the AV$_{SS}$ pin to V$_{SS}$. | p.17 □ |
| Chapter 2 | Hard | Pin Functions | P121/X1 and P122/X2 pins | The P121/X1 and P122/X2 pins are pulled down during reset. | pp. 21, 22, 24, 25 □ |
| Chapter 3 | Hard | Memory space | Vector Table Address | No interrupt sources correspond to the vector table address 0014H. | p.30 □ |
|  | Soft |  | SP: Stack pointer | Since generation of reset signal makes the SP contents undefined, be sure to initialize the SP before using the stack memory. | p.34 □ |
| Chapter 4 | Hard | Port function | P121/X1 and P122/X2 pins | The P121/X1 and P122/X2 pins are pulled down during reset. | p.50 □ |
|  |  |  | P34 pin | Because the P34 pin functions alternately as the $\overline{\text{RESET}}$ pin, if it is used as an input port pin, the function to input an external reset signal to the $\overline{\text{RESET}}$ pin cannot be used. The function of the port is selected by the option byte. For details, refer to CHAPTER 18 OPTION BYTE.<br>If a low level is input to the $\overline{\text{RESET}}$ pin before the option byte is referenced again after reset is released by the POC circuit, the 78K0S/KB1+ is reset and is held in the reset state until a high level is input to the $\overline{\text{RESET}}$ pin. | p.56 □ |
|  |  |  | P30, P31, and P43 pins | Because P30, P31, and P43 are also used as external interrupt pins, the corresponding interrupt request flag is set if each of these pins is set to the output mode and its output level is changed. To use the port pin in the output mode, therefore, set the corresponding interrupt mask flag to 1 in advance. | p.64 □ |
|  |  |  | – | Although a 1-bit memory manipulation instruction manipulates 1 bit, it accesses a port in 8-bit units. Therefore, the contents of the output latch of a pin in the input mode, even if it is not subject to manipulation by the instruction, are undefined in a port with a mixture of inputs and outputs. | p.68 □ |
| Chapter 5 | Soft | System clock | PCC: Processor clock control register | Bits 7 to 2, and 0 must be set to 0. | p.72 □ |

(2/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 5 | Soft | Main clock | OSTS: Oscillation stabilization time select register | To set and then release the STOP mode, set the oscillation stabilization time as follows.<br>Expected oscillation stabilization time of resonator ≤ Oscillation stabilization time set by OSTS | p.74 □ |
| | | | | The wait time after the STOP mode is released does not include the time from the release of the STOP mode to the start of clock oscillation (“a” in the figure below), regardless of whether STOP mode was released by Reset signal generation or interrupt generation. | p.74 □ |
| | | | | The oscillation stabilization time that elapses on power application or after release of reset is selected by the option byte. For details, refer to CHAPTER 18 OPTION BYTE. | p.74 □ |
| | Hard | Crystal/ ceramic oscillator | – | When using the crystal/ceramic oscillator, wire as follows in the area enclosed by the broken lines in Figure 5-6 to avoid an adverse effect from wiring capacitance.<br>• Keep the wiring length as short as possible.<br>• 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.<br>• Always make the ground point of the oscillator capacitor the same potential as VSS. Do not ground the capacitor to a ground pattern through which a high current flows.<br>• Do not fetch signals from the oscillator. | p.75 □ |
| Chapter 6 | Hard | 16-bit timer/event counter 00 | TM00: 16-bit timer counter 00 | Even if TM00 is read, the value is not captured by CR010. | p.87 □ |
| | | | | During TM00 is read, the count clock is stopped. | p.87 □ |
| | Soft | | CR000: 16-bit timer capture/ compare register 000 | Set CR000 to other than 0000H in the clear & start mode entered on match between TM00 and CR000. This means a 1-pulse count operation cannot be performed when this register is used as an external event counter. However, in the free-running mode and in the clear & start mode using the valid edge of TI000 pin, if CR000 is set to 0000H, an interrupt request (INTTM000) is generated when CR000 changes from 0000H to 0001H following overflow (FFFFH). | p.88 □ |
| | | | | If the new value of CR000 is less than the value of 16-bit timer counter 0 (TM00), TM00 continues counting, overflows, and then starts counting from 0 again. If the new value of CR000 is less than the old value, therefore, the timer must be reset to be restarted after the value of CR000 is changed. | p.88 □ |
| | | | | The value of CR000 after 16-bit timer/event counter 00 has stopped is not guaranteed. | p.88 □ |
| | Hard | | | The capture operation may not be performed for CR000 set in compare mode even if a capture trigger is input. | p.88 □ |
| | | | | When P31 is used as the input pin for the valid edge of TI010, it cannot be used as a timer output (TO00). Moreover, when P31 is used as TO00, it cannot be used as the input pin for the valid edge of TI010. | p.88 □ |

(3/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 6 | Hard | 16-bit timer/event counter 00 | CR000: 16-bit timer capture/ compare register 000 | If the register read period and the input of the capture trigger conflict when CR000 is used as a capture register, the capture trigger input takes precedence and the read data is undefined. Also, if the count stop of the timer and the input of the capture trigger conflict, the capture trigger is undefined. | p.88 □ |
| | Soft | | | Changing the CR000 setting may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation. | p.88 □ |
| | | | CR010: 16-bit timer capture/ compare register 010 | In the free-running mode and in the clear & start mode using the valid edge of the TI000 pin, if CR010 is set to 0000H, an interrupt request (INTTM010) is generated when CR010 changes from 0000H to 0001H following overflow (FFFFH). | p.89 □ |
| | | | | If the new value of CR010 is less than the value of 16-bit timer counter 0 (TM00), TM00 continues counting, overflows, and then starts counting from 0 again. If the new value of CR010 is less than the old value, therefore, the timer must be reset to be restarted after the value of CR010 is changed. | p.89 □ |
| | | | | The value of CR010 after 16-bit timer/event counter 00 has stopped is not guaranteed. | p.89 □ |
| | Hard | | | The capture operation may not be performed for CR010 set in compare mode even if a capture trigger is input. | p.89 □ |
| | | | | If the register read period and the input of the capture trigger conflict when CR010 is used as a capture register, the capture trigger input takes precedence and the read data is undefined. Also, if the timer count stop and the input of the capture trigger conflict, the capture data is undefined. | p.89 □ |
| | Soft | | | Changing the CR010 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation. | p.89 □ |
| | | | TMC00: 16-Bit Timer Mode Control Register 00 | 16-bit timer counter 00 (TM00) starts operation at the moment TMC002 and TMC003 (operation stop mode) are set to a value other than 0, 0, respectively. Set TMC002 and TMC003 to 0, 0 to stop the operation. | p.90 □ |
| | | | | The timer operation must be stopped before writing to bits other than the OVF00 flag. | p.91 □ |
| | Hard | | | Regardless of the CPU’s operation mode, when the timer stops, the signals input to pins TI000/TI010 are not acknowledged. | p.91 □ |
| | Soft | | | Except when TI000 pin valid edge is selected as the count clock, stop the timer operation before setting STOP mode or system clock stop mode; otherwise the timer may malfunction when the system clock starts. | p.91 □ |
| | | | | Set the valid edge of the TI000 pin with bits 4 and 5 of prescaler mode register 00 (PRM00) after stopping the timer operation. | p.91 □ |
| | | | | If the clear & start mode entered on a match between TM00 and CR000, clear & start mode at the valid edge of the TI000 pin, or free-running mode is selected, when the set value of CR000 is FFFFH and the TM00 value changes from FFFFH to 0000H, the OVF00 flag is set to 1. | p.91 □ |

(4/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 6 | Soft | 16-bit timer/event counter 00 | TMC00: 16-Bit Timer Mode Control Register 00 | Even if the OVF00 flag is cleared before the next count clock is counted (before TM00 becomes 0001H) after the occurrence of a TM00 overflow, the OVF00 flag is re-set newly and clear is disabled. | p.91 □ |
| | | | | The capture operation is performed at the fall of the count clock. An interrupt request input (INTTM0n0), however, occurs at the rise of the next count clock. | p.91 □ |
| | | | CRC00: Capture /compare control register 00 | The timer operation must be stopped before setting CRC00. | p.92 □ |
| | | | | When the clear & start mode entered on a match between TM00 and CR000 is selected by 16-bit timer mode control register 00 (TMC00), CR000 should not be specified as a capture register. | p.92 □ |
| | Hard | | | To ensure the reliability of the capture operation, the capture trigger requires a pulse longer than two cycles of the count clock selected by prescaler mode register 00 (PRM00) (refer to Figure 6-17). | p.92 □ |
| | Soft | | TOC00: 16-bit timer output control register 00 | Timer operation must be stopped before setting other than OSPT00. | p.93 □ |
| | | | | If LVS00 and LVR00 are read, 0 is read. | p.93 □ |
| | | | | OSPT00 is automatically cleared after data is set, so 0 is read. | p.93 □ |
| | | | | Do not set OSPT00 to 1 other than in one-shot pulse output mode. | p.93 □ |
| | Hard | | | A write interval of two cycles or more of the count clock selected by prescaler mode register 00 (PRM00) is required to write to OSPT00 successively. | p.93 □ |
| | Soft | | | When the TOE00 is 0, set the TOE00, LVS00, and LVR00 at the same time with the 8-bit memory manipulation instruction. When the TOE00 is 1, the LVS00 and LVR00 can be set with the 1-bit memory manipulation instruction. | p.93 □ |
| | | | PRM00: Prescaler mode register 00 | Always set data to PRM00 after stopping the timer operation. | p.95 □ |
| | | | | If the valid edge of the TI000 pin is to be set as the count clock, do not set the clear/start mode and the capture trigger at the valid edge of the TI000 pin. | p.95 □ |
| | Hard | | | In the following cases, note with caution that the valid edge of the TI0n0 pin is detected.<br><1> Immediately after a system reset, if a high level is input to the TI0n0 pin, the operation of the 16-bit timer counter 00 (TM00) is enabled<br>→ If the rising edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.<br><2> If the TM00 operation is stopped while the TI0n0 pin is high level, TM00 operation is then enabled after a low level is input to the TI0n0 pin<br>→ If the falling edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a falling edge is detected immediately after the TM00 operation is enabled.<br><3> If the TM00 operation is stopped while the TI0n0 pin is low level, TM00 operation is then enabled after a high level is input to the TI0n0 pin<br>→ If the rising edge or both rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled. | p.95 □ |

(5/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 6 | Hard | 16-bit timer/event counter 00 | PRM00: Prescaler mode register 00 | The sampling clock used to eliminate noise differs when a TI000 valid edge is used as the count clock and when it is used as a capture trigger. In the former case, the count clock is fXP, and in the latter case the count clock is selected by prescaler mode register 00 (PRM00). The capture operation is not performed until the valid edge is sampled and the valid level is detected twice, thus eliminating noise with a short pulse width. | p.95 □ |
| | | | | When using P31 as the input pin (TI010) of the valid edge, it cannot be used as a timer output (TO00). When using P31 as the timer output pin (TO00), it cannot be used as the input pin (TI010) of the valid edge. | p.95 □ |
| | Soft | | Interval timer | Changing the CR000 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation. | p.96 □ |
| | | | External Event Counter | When reading the external event counter count value, TM00 should be read. | p.101 □ |
| | | | Pulse width measurement | To use two capture registers, set the TI000 and TI010 pins. | p.102 □ |
| | | | Square-wave output | Changing the CR000 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation. | p.110 □ |
| | | | PPG output | Changing the CRC0n0 setting during TM00 operation may cause a malfunction. To change the setting, refer to 6.5 Cautions Related to 16-Bit Timer/Event Counter 00 (17) Changing compare register during timer operation. | p.112 □ |
| | | | | Values in the following range should be set in CR000 and CR010.<br>0000H < CR010 < CR000 ≤ FFFFH | p.113 □ |
| | | | | The cycle of the pulse generated through PPG output (CR000 setting value + 1) has a duty of (CR010 setting value + 1)/(CR000 setting value + 1). | p.113 □ |
| | | | One-shot pulse output with software trigger | Do not set the OSPT00 bit to 1 again 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. | p.115 □ |
| | Hard | | | When using the one-shot pulse output of 16-bit timer/event counter 00 with a software trigger, do not change the level of the TI000 pin or its alternate-function port pin.<br>Because the external trigger is valid even in this case, the timer is cleared and started even at the level of the TI000 pin or its alternate-function port pin, resulting in the output of a pulse at an undesired timing. | p.115 □ |
| | Soft | | | Do not set 0000H to the CR000 and CR010 registers. | p.116 □ |
| | | | | 16-bit timer counter 00 starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC003 and TMC002 bits. | p.117 □ |

(6/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 6 | Hard | 16-bit timer/event counter 00 | One-shot pulse output with external trigger | Do not input the external trigger again 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. | p.117 □ |
| | Soft | | | Do not set the CR000 and CR010 registers to 0000H. | p.118 □ |
| | | | | 16-bit timer counter 00 starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC002 and TMC003 bits. | p.119 □ |
| | Hard | | Timer start errors | An error of up to one clock may occur in the time required for a match signal to be generated after timer start. This is because 16-bit timer counter 00 (TM00) is started asynchronously to the count clock. | p.120 □ |
| | Soft | | One-shot pulse output | One-shot pulse output normally operates only in the free-running mode or in the clear & start mode at the valid edge of the TI000 pin. Because an overflow does not occur in the clear & start mode on a match between TM00 and CR000, one-shot pulse output is not possible. | p.121 □ |
| | | | Capture operation | If both the rising and falling edges are selected as the valid edges of the TI000 pin, capture is not performed. | p.123 □ |
| | | | | When the CRC001 bit value is 1, the TM00 count value is not captured in the CR000 register when a valid edge of the TI010 pin is detected, but the input from the TI010 pin can be used as an external interrupt source because INTTM000 is generated at that timing. | p.123 □ |
| | | | Changing compare register during timer operation | With the 16-bit timer capture/compare register 0n0 (CR0n0) used as a compare register, when changing CR0n0 around the timing of a match between 16-bit timer counter 00 (TM00) and 16-bit timer capture/compare register 0n0 (CR0n0) during timer counting, the change timing may conflict with the timing of the match, so the operation is not guaranteed in such cases. To change CR0n0 during timer counting, follow the procedure below using an INTTM000 interrupt. | p.124 □ |
| | | | | If CR010 is changed during timer counting without performing processing <1> above, the value in CR010 may be rewritten twice or more, causing an inversion of the output level of the TO00 pin at each rewrite. | p.124 □ |
| Chapter 7 | Soft | 8-bit timer 80 | CR80: 8-bit compare register 80 | When changing the value of CR80, be sure to stop the timer operation. If the value of CR80 is changed with the timer operation enabled, a match interrupt request signal may be generated immediately. | p.128 □ |
| | | | TMC80: 8-bit timer mode control register 80 | Be sure to set TMC80 after stopping the timer operation. | p.129 □ |
| | | | | Be sure to clear bits 0 and 6 to 0. | p.129 □ |
| | | | Interval timer | When changing the value of CR80, be sure to stop the timer operation. If the value of CR80 is changed with the timer operation enabled, a match interrupt request signal may be generated immediately. | p.130 □ |
| | | | | If the count clock of TMC80 is set and the operation of TM80 is enabled at the same time by using an 8-bit memory manipulation instruction, the error of one cycle after the timer is started may be 1 clock or more. Therefore, be sure to follow the above sequence when using TM80 as an interval timer. | p.130 □ |

(7/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 7 | Hard | 8-bit timer 80 | Error when timer starts | The time from starting the timer to generation of the match signal includes an error of up to 1.5 clocks. This is because, if the timer is started while the count clock is high, the rising edge may be immediately detected and the counter may be incremented (refer to Figure 7-6). | p.132 □ |
| | Soft | | CR80: 8-bit compare register 80 | 8-bit compare register 80 (CR80) can be set to 00H. | p.132 □ |
| | | | STOP mode | Before executing the STOP instruction, be sure to stop the timer operation (TCE80 = 0). | p.132 □ |
| Chapter 8 | Soft | 8-bit timer H1 | CMP01: 8-bit timer H compare register 01 | CMP01 cannot be rewritten during timer count operation. | p.135 □ |
| | | | CMP11: 8-bit timer H compare register 11 | In the PWM output mode, be sure to set CMP11 when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to CMP11). | p.135 □ |
| | | | TMHMD1: 8-bit timer H mode register 1 | When TMHE1 = 1, setting the other bits of the TMHMD1 register is prohibited. | p.137 □ |
| | | | | In the PWM output mode, be sure to set 8-bit timer H compare register 11 (CMP11) when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register). | p.137 □ |
| | Hard | | PWM output | In PWM output mode, three operation clocks (signal selected using the CKS12 to CKS10 bits of the TMHMD1 register) are required to transfer the CMP11 register value after rewriting the register. | p.143 □ |
| | Soft | | | Be sure to set the CMP11 register when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register). | p.143 □ |
| | | | | Make sure that the CMP11 register setting value (M) and CMP01 register setting value (N) are within the following range.<br>00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH | p.144 □ |
| Chapter 9 | Soft | Watchdog timer | WDTM : Watchdog timer mode register | Set bits 7, 6, and 5 to 0, 1, and 1, respectively (when “low-speed Ring-OSC cannot be stopped” is selected by the option byte, other values are ignored). | p.152 □ |
| | | | | After reset is released, WDTM can be written only once by an 8-bit memory manipulation instruction. If writing is attempted a second time, an internal reset signal is generated. | p.152 □ |
| | | | | WDTM cannot be set by a 1-bit memory manipulation instruction. | p.152 □ |
| | | | | When using the flash memory self programming by self writing, set the overflow time for the watchdog timer so that enough everflow time is secured (Example 1-byte writing: 200 μs MIN., 1-block deletion: 10 ms MIN.). | p.152 □ |
| | | | | If a value other than ACH is written to WDTE, an internal reset signal is generated. | p.152 □ |
| | | | | If a 1-bit memory manipulation instruction is executed for WDTE, an internal reset signal is generated. | p.152 □ |

(8/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 9 | Hard | Watchdog timer | When "low-speed Ring-OSC cannot be stopped" is selected by option byte | In this mode, operation of the watchdog timer cannot be stopped even during STOP instruction execution. For 8-bit timer H1 (TMH1), a division of the low-speed Ring-OSC clock can be selected as the count source, so clear the watchdog timer using the interrupt request of TMH1 before the watchdog timer overflows after STOP instruction execution. If this processing is not performed, an internal reset signal is generated when the watchdog timer overflows after STOP instruction execution. | p.153 □ |
| | | | When "low-speed Ring-OSC can be stopped by software" is selected by option byte | In this mode, watchdog timer operation is stopped during HALT/STOP instruction execution. After HALT/STOP mode is released, counting is started again using the operation clock of the watchdog timer set before HALT/STOP instruction execution by WDTM. At this time, the counter is not cleared to 0 but holds its value. | p.155 □ |
| Chapter 10 | Soft | A/D Converter | Sampling time and conversion time | The above sampling time and conversion time do not include the clock frequency error. Select the conversion time taking the clock frequency error into consideration. | p.161 □ |
| | | | ADM : A/D converter mode register | The above sampling time and conversion time do not include the clock frequency error. Select the conversion time taking the clock frequency error into consideration. | p.166 □ |
| | | | | If a bit other than ADCS of ADM is manipulated while A/D conversion is stopped (ADCS = 0) and then A/D conversion is started, execute two NOP instructions or an instruction equivalent to two machine cycles, and set ADCS to 1. | p.166 □ |
| | | | | A/D conversion must be stopped (ADCS = 0) before rewriting bits FR0 to FR2. | p.166 □ |
| | | | | Be sure to clear bits 6, 2, and 1 to 0. | p.166 □ |
| | | | ADS: Analog input channel specification register | Be sure to clear bits 2 to 7 of ADS to 0. | p.167 □ |
| | | | ADCR: 10-bit A/D conversion result register | When writing to the A/D converter mode register (ADM) and analog input channel specification register (ADS), the contents of ADCR may become undefined. Read the conversion result following conversion completion before writing to ADM and ADS. Using timing other than the above may cause an incorrect conversion result to be read. | p.167 □ |
| | | | PMC2: Port mode control register 2 | When PMC20 to PMC23 are set to 1, the P20/ANI0 to P23/ANI3 pins cannot be used as port pins. | p.168 □ |
| | | | A/D converter operations | Make sure the period of <1> to <4> is 1 μs or more. | p.173 □ |
| | | | | It is no problem if the order of <1> and <2> is reversed. | p.173 □ |
| | | | | <1> can be omitted. However, ignore the data resulting from the first conversion after <4> in this case. | p.173 □ |
| | | | | The period from <5> to <8> differs from the conversion time set using bits 5 to 3 (FR2 to FR0) of ADM. The period from <7> to <8> is the conversion time set using FR2 to FR0. | p.173 □ |

(9/20)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 10 | Hard | A/D Converter | Operating current in STOP mode | The A/D converter stops operating in the STOP mode. At this time, the operating current can be reduced by clearing bit 7 (ADCS) and bit 0 (ADCE) of the A/D converter mode register (ADM) to 0. | p.176 □ |
| | | | Input range of ANI0 to ANI3 | Observe the rated range of the ANI0 to ANI3 input voltage. If a voltage of $AV_{REF}$ or higher and $AV_{SS}$ or lower (even in the range of absolute maximum ratings) is input to an analog input channel, the converted value of that channel becomes undefined. In addition, the converted values of the other channels may also be affected. | p.176 □ |
| | Soft | | Conflicting operations | ADCR, ADCRH read has priority. After the read operation, the new conversion result is written to ADCR, ADCRH. | p.176 □ |
| | | | | ADM or ADS write has priority. ADCR, ADCRH write is not performed, nor is the conversion end interrupt signal (INTAD) generated. | p.176 □ |
| | Hard | | Noise countermeasures | To maintain the 10-bit resolution, attention must be paid to noise input to the $AV_{REF}$ pin and pins ANI0 to ANI3.<br><1> Connect a capacitor with a low equivalent resistance and a high frequency response to the power supply.<br><2> Because the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally, as shown in Figure 10-19, to reduce noise.<br><3> Do not switch the A/D conversion function of the ANI0 to ANI3 pins to their alternate functions during conversion.<br><4> The conversion accuracy can be improved by setting HALT mode immediately after the conversion starts. | p.176 □ |
| | | | ANI0/P20 to ANI3/P23 | The analog input pins (ANI0 to ANI3) are also used as input port pins (P20 to P23).<br>When A/D conversion is performed with any of ANI0 to ANI3 selected, do not access port 2 (P20 to P23) while conversion is in progress; otherwise the conversion resolution may be degraded. | p.177 □ |
| | | | | If a digital pulse is applied to the pins adjacent to the pins currently used for A/D conversion, the expected value of the A/D conversion may not be obtained due to coupling noise. Therefore, do not apply a pulse to the pins adjacent to the pin undergoing A/D conversion. | p.177 □ |
| | | | Input impedance of ANI0 to ANI3 pins | In this A/D converter, the internal sampling capacitor is charged and sampling is performed for approx. one sixth of the conversion time.<br>Since only the leakage current flows other than during sampling and the current for charging the capacitor also flows during sampling, the input impedance fluctuates during sampling and in the other state.<br>If the shortest conversion time of the reference voltage is used, to perform sufficient sampling, it is recommended to make the output impedance of the analog input source 1 kΩ or lower, or attach a capacitor of around 0.01 μF to 0.1 μF to the ANI0 to ANI3 pins (see Figure 10-19). | p.177 □ |

(10/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 10 | Soft | A/D converter | ADIF: Interrupt request flag | The interrupt request flag (ADIF) is not cleared even if the analog input channel specification register (ADS) is changed.<br>Therefore, if an analog input pin is changed during A/D conversion, the A/D conversion result and ADIF for the pre-change analog input may be set just before the ADS rewrite. Caution is therefore required since, at this time, when ADIF is read immediately after the ADS rewrite, ADIF is set despite the fact A/D conversion for the post-change analog input has not ended.<br>When A/D conversion is stopped and then resumed, clear ADIF before the A/D conversion operation is resumed. | p.177 □ |
| | | | Conversion results just after A/D conversion start | The first A/D conversion value immediately after A/D conversion starts may not fall within the rating range if the ADCS bit is set to 1 within 1 $\mu$s after the ADCE bit was set to 1, or if the ADCS bit is set to 1 with the ADCE bit = 0. Take measures such as polling the A/D conversion end interrupt request (INTAD) and removing the first conversion result. | p.178 □ |
| | | | A/D conversion result register (ADCR, ADCRH) read operation | When a write operation is performed to the A/D converter mode register (ADM) and analog input channel specification register (ADS), the contents of ADCR and ADCRH may become undefined. Read the conversion result following conversion completion before writing to ADM and ADS. Using a timing other than the above may cause an incorrect conversion result to be read. | p.178 □ |
| Chapter 11 | Hard | Serial interface UART6 | UART mode | The TxD6 output inversion function inverts only the transmission side and not the reception side. To use this function, the reception side must be ready for reception of inverted data. | p.179 □ |
| | Soft | | | If clock supply to serial interface UART6 is not stopped (e.g., in the HALT mode), normal operation continues. If clock supply to serial interface UART6 is stopped (e.g., in the STOP mode), each register stops operating, and holds the value immediately before clock supply was stopped. The TxD6 pin also holds the value immediately before clock supply was stopped and outputs it. However, the operation is not guaranteed after clock supply is resumed. Therefore, reset the circuit so that POWER6 = 0, RXE6 = 0, and TXE6 = 0. | p.179 □ |
| | | | | If data is continuously transmitted, the communication timing from the stop bit to the next start bit is extended two operating clocks of the macro. However, this does not affect the result of communication because the reception side initializes the timing when it has detected a start bit. Do not use the continuous transmission function if the interface is incorporated in LIN. | p.179 □ |
| | | | TXB6: Transmit buffer register 6 | Do not write data to TXB6 when bit 1 (TXBF6) of asynchronous serial interface transmission status register 6 (ASIF6) is 1. | p.185 □ |
| | | | | Do not refresh (write the same value to) TXB6 by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of asynchronous serial interface operation mode register 6 (ASIM6) are 1 or when bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 are 1). | p.185 □ |

(11/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 11 | Soft | Serial interface UART6 | ASIM6: Asynchronous serial interface operation mode register 6 | At startup, set POWER6 to 1 and then set TXE6 to 1. To stop the operation, clear TXE6 to 0, and then clear POWER6 to 0. | p.187 □ |
| | | | | At startup, set POWER6 to 1 and then set RXE6 to 1. To stop the operation, clear RXE6 to 0, and then clear POWER6 to 0. | p.187 □ |
| | | | | Set POWER6 to 1 and then set RXE6 to 1 while a high level is input to the RxD6 pin. If POWER6 is set to 1 and RXE6 is set to 1 while a low level is input, reception is started. | p.187 □ |
| | | | | Clear the TXE6 and RXE6 bits to 0 before rewriting the PS61, PS60, and CL6 bits. | p.187 □ |
| | | | | Fix the PS61 and PS60 bits to 0 when mounting the device on LIN. | p.187 □ |
| | | | | Make sure that TXE6 = 0 when rewriting the SL6 bit. Reception is always performed with "the number of stop bits = 1", and therefore, is not affected by the set value of the SL6 bit. | p.187 □ |
| | | | | Make sure that RXE6 = 0 when rewriting the ISRM6 bit. | p.187 □ |
| | | | ASIS6: Asynchronous serial interface reception error status register 6 | The operation of the PE6 bit differs depending on the set values of the PS61 and PS60 bits of asynchronous serial interface operation mode register 6 (ASIM6). | p.188 □ |
| | | | | The first bit of the receive data is checked as the stop bit, regardless of the number of stop bits. | p.188 □ |
| | | | | If an overrun error occurs, the next receive data is not written to receive buffer register 6 (RXB6) but discarded. | p.188 □ |
| | | | ASIF6: Asynchronous serial interface transmission status register 6 | To transmit data continuously, write the first transmit data (first byte) to the TXB6 register. Be sure to check that the TXBF6 flag is "0". If so, write the next transmit data (second byte) to the TXB6 register. If data is written to the TXB6 register while the TXBF6 flag is "1", the transmit data cannot be guaranteed. | p.189 □ |
| | | | | To initialize the transmission unit upon completion of continuous transmission, be sure to check that the TXSF6 flag is "0" after generation of the transmission completion interrupt, and then execute initialization. If initialization is executed while the TXSF6 flag is "1", the transmit data cannot be guaranteed. | p.189 □ |
| | | | CKSR6: Clock selection register 6 | Make sure POWER6 = 0 when rewriting TPS63 to TPS60. | p.190 □ |
| | | | BRGC6: Baud rate generator control register 6 | Make sure that bit 6 (TXE6) and bit 5 (RXE6) of the ASIM6 register = 0 when rewriting the MDL67 to MDL60 bits. | p.191 □ |
| | Hard | | | The baud rate is the output clock of the 8-bit counter divided by 2. | p.191 □ |
| | Soft | | ASICL6: Asynchronous serial interface control register 6 | ASICL6 can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1 or bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1). However, if the SBRT6 = 1 and SBTT = 1 are set in the refresh operation during the SBF reception (SBRF6 = 1) or SBF transmission (between the SBTT6 setting (1) and the INTST6 occurrence), it triggers the SBF reception and SBF transmission again, so do not set. | p.192 □ |
| | | | | In the case of an SBF reception error, return the mode to the SBF reception mode again and hold (1) the status of the SBRF6 flag. | p.193 □ |

(12/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 11 | Soft | Serial interface UART6 | ASICL6: Asynchronous serial interface control register 6 | Before setting the SBRT6 bit to 1, make sure that bit 7 (POWER6) and bit 5 (RXE6) of ASIM6 = 1. Moreover, after setting the SBRT6 bit to 1, do not clear the SBRT6 bit to 0 before the SBF reception ends (an interrupt request signal is generated). | p.193 □ |
| | | | | The read value of the SBRT6 bit is always 0. SBRT6 is automatically cleared to 0 after SBF reception has been correctly completed. | p.193 □ |
| | | | | Before setting the SBTT6 bit to 1, make sure that bit 7 (POWER6) and bit 6 (TXE6) of ASIM6 = 1. Moreover, after setting the SBTT6 bit to 1, do not clear the SBTT6 bit to 0 before the SBF transmission ends (an interrupt request signal is generated). | p.193 □ |
| | | | | The read value of the SBTT6 bit is always 0. SBTT6 is automatically cleared to 0 at the end of SBF transmission. | p.193 □ |
| | | | | Before rewriting the DIR6 and TXDLV6 bits, clear the TXE6 and RXE6 bits to 0. | p.193 □ |
| | | | Bits 7, 6, and 5 (POWER6, TXE6, and RXE6) of ASIM6 | Clear POWER6 to 0 after clearing TXE6 and RXE6 to 0 to set the operation stop mode.<br>To start the operation, set POWER6 to 1, and then set TXE6 and RXE6 to 1. | p.195 □ |
| | | | UART mode | Take relationship with the other party of communication into consideration when setting the port mode register and port register. | p.196 □ |
| | | | Parity types and operation | Fix the PS61 and PS60 bits to 0 when the device is incorporated in LIN. | p.200 □ |
| | | | Continuous transmission | The TXBF6 and TXSF6 flags of the ASIF6 register change from “10” to “11”, and to “01” during continuous transmission. To check the status, therefore, do not use a combination of the TXBF6 and TXSF6 flags for judgment. Judge whether continuous transmission is possible or not by reading only the TXBF flag. | p.202 □ |
| | | | | When the device is incorporated in a LIN, the continuous transmission function cannot be used. Make sure that asynchronous serial interface transmission status register 6 (ASIF6) is 00H before writing transmit data to transmit buffer register 6 (TXB6). | p.202 □ |
| | | | TXBF6 during Continuous Transmission: Bit 1 of ASIF6 | To transmit data continuously, write the first transmit data (first byte) to the TXB6 register. Be sure to check that the TXBF6 flag is “0”. If so, write the next transmit data (second byte) to the TXB6 register. If data is written to the TXB6 register while the TXBF6 flag is “1”, the transmit data cannot be guaranteed. | p.202 □ |
| | | | TXSF6 during Continuous Transmission: Bit 0 of ASIF6 | To initialize the transmission unit upon completion of continuous transmission, be sure to check that the TXSF6 flag is “0” after generation of the transmission completion interrupt, and then execute initialization. If initialization is executed while the TXSF6 flag is “1”, the transmit data cannot be guaranteed. | p.202 □ |
| | | | | During continuous transmission, an overrun error may occur, which means that the next transmission was completed before execution of INTST6 interrupt servicing after transmission of one data frame. An overrun error can be detected by developing a program that can count the number of transmit data and by referencing the TXSF6 flag. | p.202 □ |

(13/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 11 | Soft | Serial interface UART6 | Normal reception | Be sure to read receive buffer register 6 (RXB6) even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the reception error status will persist. | p.206 □ |
| | | | | Reception is always performed with the “number of stop bits = 1”. The second stop bit is ignored. | p.206 □ |
| | | | | Be sure to read asynchronous serial interface reception error status register 6 (ASIS6) before reading RXB6. | p.206 □ |
| | | | Generation of serial clock | Keep the baud rate error during transmission to within the permissible error range at the reception destination. | p.212 □ |
| | | | | Make sure that the baud rate error during reception satisfies the range shown in (4) Permissible baud rate range during reception. | p.212 □ |
| | | | Permissible baud rate range during reception | Make sure that the baud rate error during reception is within the permissible error range, by using the calculation expression shown below. | p.214 □ |
| Chapter 12 | Soft | Multiplier | MUL0: 16-bit multiplication result storage register 0 | Although this register is manipulated with a 16-bit memory manipulation instruction, it can be also manipulated with an 8-bit memory manipulation instruction. When using an 8-bit memory manipulation instruction, however, access the register by means of direct addressing. | p.217 □ |
| | | | MULC0: control register 0 | Be sure to clear bits 1 to 7 to 0. | p.219 □ |
| Chapter 13 | Hard | Interrupt function | Vector table address | No interrupt sources correspond to the vector table address 0014H. | p.222 □ |
| | Soft | | IF0, IF1: Interrupt request flag registers 0, 1<br>MK0, MK1: Interrupt mask flag registers 0, 1 | Because P30, P31, P41, and P43 have an alternate function as external interrupt inputs, when the output level is changed by specifying the output mode of the port function, an interrupt request flag is set. Therefore, the interrupt mask flag should be set to 1 before using the output mode. | pp. 225, 226 □ |
| | | | INTM0: External interrupt mode register 0 | Be sure to clear bits 0 and 1 to 0. | p.227 □ |
| | | | | Before setting the INTM0 register, be sure to set the corresponding interrupt mask flag (××MK× = 1) to disable interrupts. After setting the INTM0 register, clear the interrupt request flag (××IF× = 0), then clear the interrupt mask flag (××MK× = 0), which will enable interrupts. | p.227 □ |
| | | | INTM1: External interrupt mode register 1 | Be sure to clear bits 2 to 7 to 0. | p.228 □ |
| | | | | Before setting INTM1, set PMK3 to 1 to disable interrupts.<br>To enable interrupts, clear PIF3 to 0, then clear PMK3 to 0. | p.228 □ |
| | | | Interrupt request pending | Interrupt requests will be held pending while the interrupt request flag registers 0, 1 (IF0, IF1) or interrupt mask flag registers 0, 1 (MK0, MK1) are being accessed. | p.231 □ |

(14/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 14 | Soft | Standby function | – | The LSRSTOP setting is valid only when “Can be stopped by software” is set for the low-speed Ring-OSC oscillator by the option byte. | p.234 □ |
| | Hard | | STOP mode | When shifting to the STOP mode, be sure to stop the peripheral hardware operation before executing STOP instruction (except the peripheral hardware that operates on the low-speed Ring-OSC clock). | p.235 □ |
| | | | STOP mode, HALT mode | The following sequence is recommended for operating current reduction of the A/D converter when the standby function is used: First clear bit 7 (ADCS) and bit 0 (ADCE) of the A/D converter mode register (ADM) to 0 to stop the A/D conversion operation, and then execute the HALT or STOP instruction. | p.235 □ |
| | | | STOP mode | If the low-speed Ring-OSC oscillator is operating before the STOP mode is set, oscillation of the low-speed Ring-OSC clock cannot be stopped in the STOP mode (refer to Table 14-1). | p.235 □ |
| | Soft | | OSTS: Oscillation stabilization time select register | To set and then release the STOP mode, set the oscillation stabilization time as follows.<br>Expected oscillation stabilization time of resonator ≤ Oscillation stabilization time set by OSTS | p.236 □ |
| | Hard | | | The wait time after the STOP mode is released does not include the time from the release of the STOP mode to the start of clock oscillation (“a” in the figure below), regardless of whether STOP mode was released by reset signal generation or interrupt generation. | p.236 □ |
| | Soft | | | The oscillation stabilization time that elapses on power application or after release of reset is selected by the option byte. For details, refer to CHAPTER 18 OPTION BYTE. | p.236 □ |
| | | | Settings and operating statuses in HALT mode | Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. | p.237 □ |
| | | | Settings and operating statuses in STOP mode | Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. Thus, in the STOP mode, the normal operation mode is restored after the STOP instruction is executed and then the operation is stopped for 34 $\mu$s (TYP.) (after an additional wait time for stabilizing the oscillation set by the oscillation stabilization time select register (OSTS) has elapsed when crystal/ceramic oscillation is used). | p.240 □ |
| Chapter 15 | Hard | Reset function | – | For an external reset, input a low level for 2 $\mu$s or more to the $\overline{\text{RESET}}$ pin. | p.244 □ |
| | | | | During reset signal generation, the system clock and low-speed Ring-OSC clock stop oscillating. | p.244 □ |
| | | | | When the $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34), the 78K0S/KB1+ is reset if a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the POC circuit and before the option byte is referenced again. The reset status is retained until a high level is input to the $\overline{\text{RESET}}$ pin. | p.244 □ |
| | | | | The LVI circuit is not reset by the internal reset signal of the LVI circuit. | p.245 □ |
| | | | Timing of reset by overflow of watchdog timer | The watchdog timer is also reset in the case of an internal reset of the watchdog timer. | p.247 □ |

(15/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 15 | Soft | Reset function | RESF: Reset control flag register | Do not read data by a 1-bit memory manipulation instruction. | p.251 □ |
| Chapter 16 | Soft | Power-on-clear circuit | Functions of power-on-clear circuit | If an internal reset signal is generated in the POC circuit, the reset control flag register (RESF) is cleared to 00H. | p.252 □ |
| | Hard | | | Because the detection voltage ($V_{POC}$) of the POC circuit is in a range of 2.1 V ±0.1 V, use a voltage in the range of 2.2 to 5.5 V. | p.252 □ |
| | Soft | | Cautions for power-on-clear circuit | In a system where the supply voltage ($V_{DD}$) fluctuates for a certain period in the vicinity of the POC detection voltage ($V_{POC}$), the system may be repeatedly reset and released from the reset status. In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking the following action. | p.254 □ |
| Chapter 17 | Soft | Low-voltage detector | LVIM: Low-voltage detect register | To stop LVI, follow either of the procedures below.<br>• When using 8-bit manipulation instruction: Write 00H to LVIM.<br>• When using 1-bit memory manipulation instruction: Clear LVION to 0. | p.257 □ |
| | | | | Be sure to set bits 2 to 6 to 0. | p.257 □ |
| | | | LVIS: Low-voltage detection level select register | Bits 4 to 7 must be set to 0. | p.258 □ |
| | | | When used as reset | <1> must always be executed. When LVIMK = 0, an interrupt may occur immediately after the processing in <3>. | p.259 □ |
| | | | | If supply voltage ($V_{DD}$) ≥ detection voltage ($V_{LVI}$) when LVIM is set to 1, an internal reset signal is not generated. | p.259 □ |
| | | | Cautions for low-voltage detector | In a system where the supply voltage ($V_{DD}$) fluctuates for a certain period in the vicinity of the LVI detection voltage ($V_{LVI}$), the operation is as follows depending on how the low-voltage detector is used.<br><1> When used as reset<br>The system may be repeatedly reset and released from the reset status.<br>In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking action (1) below.<br><2> When used as interrupt<br>Interrupt requests may be frequently generated. Take action (2) below. | p.262 □ |
| Chapter 18 | Hard | Option byte | Low-speed Ring-OSC clock oscillation | If it is selected that low-speed Ring-OSC clock oscillation cannot be stopped, the count clock to the watchdog timer (WDT) is fixed to low-speed Ring-OSC. | p.266 □ |
| | | | | If it is selected that low-speed Ring-OSC can be stopped by software, supply of the count clock to WDT is stopped in the HALT/STOP mode, regardless of the setting of bit 0 (LSRSTOP) of the low-speed Ring-OSC mode register (LSRCM). Similarly, clock supply is also stopped when a clock other than the low-speed Ring-OSC is selected as a count clock to WDT. If low-speed Ring-OSC is selected as the count clock to 8-bit timer H1, however, the count clock is supplied in the HALT/STOP mode while low-speed Ring-OSC operates (LSRSTOP = 0). | p.266 □ |

(16/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 18 | Hard | Option byte | Selection of system clock source | Because the X1 and X2 pins are also used as the P121 and P122 pins, the conditions under which the X1 and X2 pins can be used differ depending on the selected system clock source.<br>(1) High-speed Ring-OSC clock<br>P121 and P122 can be used as I/O port pins.<br>(2) Crystal/ceramic oscillation clock<br>The X1 and X2 pins cannot be used as I/O port pins because they are used as clock input pins.<br>(3) External clock input<br>Because the X1 pin is used as an external clock input pin, P121 cannot be used as an I/O port pin. | p.266 □ |
| | | | Control of $\overline{\text{RESET}}$ pin | If a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the power-on clear function and before the option byte is referenced again, the 78K0S/KB1+ is reset, and the status is held until a high level is input to the $\overline{\text{RESET}}$ pin. | p.266 □ |
| | | | Oscillation stabilization time on power application or after reset release | The setting of this option is valid only when the crystal/ceramic oscillation clock is selected as the system clock source. No wait time elapses if the high-speed Ring-OSC or external clock input is selected as the system clock source. | p.267 □ |
| Chapter 19 | Soft | Flash memory | PG-FP4 GUI Software setting value example | The above is a recommendation value. A value may change according to the environment to be used.<br>Set up after surely performing sufficient evaluation. | p.275 □ |
| | | | Security settings | The security setting is valid when the programming mode is set next time. Therefore, when the security setting command is executed, exit from the programming mode, then set the programming mode again. | p.279 □ |
| | | | | After the security setting of the batch erase is set, erasure cannot be performed for the device. In addition, even if a write command is executed, data different from that which has already been written to the flash memory cannot be written because the erase command is disabled. | p.279 □ |
| | | | Self programming function | Self programming processing must be included in the program before performing self writing. | p.280 □ |
| | | | | If an interrupt occurs during self programming, the interrupt request flag is set (1), and interrupt servicing is performed after the self programming mode is released. To avoid this operation, disable interrupt servicing (by setting MK0 and MK1 to FFH, and executing the DI instruction) during self programming or before a mode is shifted from the normal mode to the self programming mode with a specific sequence. | p.283 □ |
| | | | | No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 19-11 for the time taken for the execution of self programming. | p.283 □ |
| | | | | RAM is not used while a self programming command is being executed. | p.283 □ |
| | | | | If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed. | p.283 □ |
| | | | | The value of the blank data set during block erasure is FFH. | p.283 □ |

(17/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 19 | Soft | Flash memory | Self programming function | When the oscillator or the external clock is selected as the main clock, a wait time of 16 $\mu$s is required starting from the setting of the self programming mode to the execution of the HALT instruction. | p.283 □ |
| | | | | The state of the pins in self programming mode is the same as that in HALT mode. | p.283 □ |
| | | | | Since the security function set via on-board/off-board programming is disabled in self programming mode, the self programming command can be executed regardless of the security function setting. To disable write or erase processing during self programming, set the protect byte. | p.283 □ |
| | | | | Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command. | p.283 □ |
| | | | Format of flash programming mode control register (FLPMC) | Note the following when setting the self programming mode.<br>• If an interrupt occurs during self programming, the interrupt request flag is set (1), and interrupt servicing is performed after the self programming mode is released. To avoid this operation, disable interrupt servicing (by setting MK0 and MK1 to FFH, and executing the DI instruction) during self programming or before a mode is shifted from the normal mode to the self programming mode with a specific sequence.<br>• No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 19-11 for the time taken for the execution of self programming.<br>• If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed. | p.284 □ |
| | | | | When the oscillator or the external clock is selected as the main clock, a wait time of 16 $\mu$s is required from setting FLSPM to 1 to execution of the HALT instruction. | p.284 □ |
| | | | PFCMD: Flash protect command register | Disable interrupt servicing (by setting MK0 and MK1 to FFH and executing the DI instruction) while the specific sequence is under execution. | p.285 □ |
| | | | FLAPH and FLAPL: Flash address pointers H and L | Be sure to clear bits 4 to 7 of FLAPH and FLAPHC to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command. | p.288 □ |
| | | | FLAPHC and FLAPLC: Flash address pointer H compare register and flash address pointer L compare register | Be sure to clear bits 4 to 7 of FLAPH and FLAPHC to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command. | p.288 □ |
| | | | | Set the number of the block subject to a block erase, write, verify, or blank check (same value as FLAPH) to FLAPHC. | p.288 □ |
| | | | | Clear FLAPLC to 00H when a block erase is performed, and FFH when a blank check is performed. | p.288 □ |

(18/18)

| Chapter | Classification | Function | Details of Function | Cautions | Page |
|---|---|---|---|---|---|
| Chapter 19 | Soft | Flash memory | Shifting to self programming mode | Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written. | pp. 291, 292, 294, 295 □ |
| | | | Shifting to normal mode | | |
| | | | Byte write operation | If a write results in failure, erase the block once and write to it again. | p.303 □ |
| Chapter 21 | Hard | Electrical specifications | Absolute maximum ratings | 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. | p.337 □ |
| | | | X1 Oscillator | When using the X1 oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.<br>• Keep the wiring length as short as possible.<br>• Do not cross the wiring with the other signal lines.<br>• Do not route the wiring near a signal line through which a high fluctuating current flows.<br>• Always make the ground point of the oscillator capacitor the same potential as $V_{SS}$.<br>• Do not ground the capacitor to a ground pattern through which a high current flows.<br>• Do not fetch signals from the oscillator. | p.338 □ |