REJ09B0436-0100

Everywhere you imagine. RENESAS

16

# H8/38537 Group

## Hardware Manual

Hardware Manual

Renesas 16-Bit Single-Chip Microcomputer
H8 Family / H8/300H Super Low Power Series

H8/38532
H8/38533
H8/38534
H8/38535
H8/38536
H8/38537

Rev.1.00
Revision Date: May 30, 2008

Renesas Technology
www.renesas.com

## Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com )
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

# General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

     In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

     In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

# How to Use This Manual

1. Objective and Target Users

   This manual was written to explain the hardware functions and electrical characteristics of this LSI to the target users, i.e. those who will be using this LSI in the design of application systems. Target users are expected to understand the fundamentals of electrical circuits, logic circuits, and microcomputers.

   This manual is organized in the following items: an overview of the product, descriptions of the CPU, system control functions, and peripheral functions, electrical characteristics of the device, and usage notes.

   When designing an application system that includes this LSI, take all points to note into account. Points to note are given in their contexts and at the final part of each section, and in the section giving usage notes.

   The list of revisions is a summary of major points of revision or addition for earlier versions. It does not cover all revised items. For details on the revised points, see the actual locations in the manual.

   The following documents have been prepared for the H8/38537 Group. Before using any of the documents, please visit our web site to verify that you have the most up-to-date available version of the document.

| Document Type | Contents | Document Title | Document No. |
|---|---|---|---|
| Data Sheet | Overview of hardware and electrical characteristics | — | — |
| Hardware Manual | Hardware specifications (pin assignments, memory maps, peripheral specifications, electrical characteristics, and timing charts) and descriptions of operation | H8/38537 Group Hardware Manual | This manual |
| Software Manual | Detailed descriptions of the CPU and instruction set | H8/300H Series Software Manual | REJ09B0213 |
| Application Note | Examples of applications and sample programs | The latest versions are available from our web site. | |
| Renesas Technical Update | Preliminary report on the specifications of a product, document, etc. | | |

## 2. Description of Numbers and Symbols

Aspects of the notations for register names, bit names, numbers, and symbolic names in this manual are explained below.

(1) Overall notation

In descriptions involving the names of bits and bit fields within this manual, the modules and registers to which the bits belong may be clarified by giving the names in the forms "module name"."register name"."bit name" or "register name"."bit name".

(2) Register notation

The style "register name"_"instance number" is used in cases where there is more than one instance of the same function or similar functions.

[Example] CMCSR_0: Indicates the CMCSR register for the compare-match timer of channel 0.

(3) Number notation

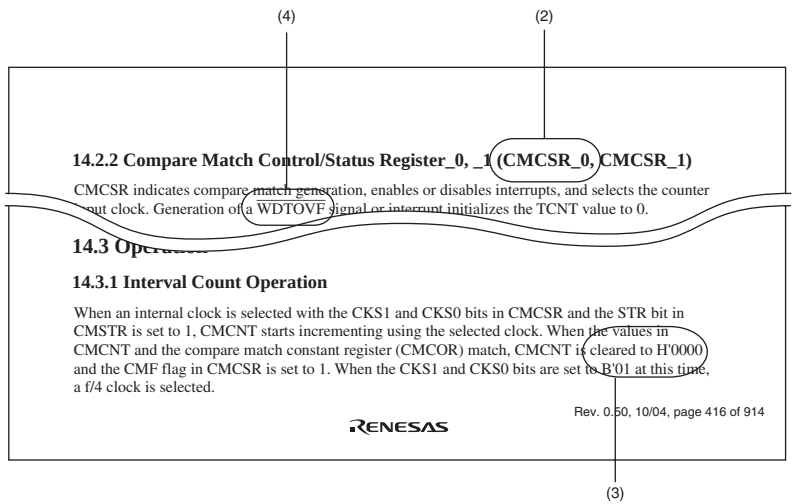
Binary numbers are given as B'nnnn (B' may be omitted if the number is obviously binary), hexadecimal numbers are given as H'nnnn or 0xnnnn, and decimal numbers are given as nnnn.

[Examples] Binary: B'11 or 11
Hexadecimal: H'EFA0 or 0xEFA0
Decimal: 1234

(4) Notation for active-low

An overbar on the name indicates that a signal or pin is active-low.

[Example] $\overline{\text{WDTOVF}}$




Note: The bit names and sentences in the above figure are examples and have nothing to do with the contents of this manual.

3. Description of Registers

Each register description includes a bit chart, illustrating the arrangement of bits, and a table of bits, describing the meanings of the bit settings. The standard format and notation for bit charts and tables are described below.

**[Table of Bits]**

| Bit (1) | Bit Name (2) | Initial Value (3) | R/W (4) | Description (5) |
|---|---|---|---|---|
| 15 | – | 0 | R | Reserved |
| 14 | – | 0 | R | These bits are always read as 0. |
| 13 to 11 | ASID2 to ASID0 | All 0 | R/W | Address Identifier<br>These bits enable or disable the pin function. |
| 10 | – | 0 | R | Reserved<br>This bit is always read as 0. |
| 9 | – | 1 | R | Reserved<br>This bit is always read as 1. |
| | – | 0 | | |

Note: The bit names and sentences in the above figure are examples, and have nothing to do with the contents of this manual.

(1) Bit
Indicates the bit number or numbers.
In the case of a 32-bit register, the bits are arranged in order from 31 to 0. In the case of a 16-bit register, the bits are arranged in order from 15 to 0.

(2) Bit name
Indicates the name of the bit or bit field.
When the number of bits has to be clearly indicated in the field, appropriate notation is included (e.g., ASID[3:0]).
A reserved bit is indicated by "–".
Certain kinds of bits, such as those of timer counters, are not assigned bit names. In such cases, the entry under Bit Name is blank.

(3) Initial value
Indicates the value of each bit immediately after a power-on reset, i.e., the initial value.
0: The initial value is 0
1: The initial value is 1
–: The initial value is undefined

(4) R/W
For each bit and bit field, this entry indicates whether the bit or field is readable or writable, or both writing to and reading from the bit or field are impossible.
The notation is as follows:

R/W: The bit or field is readable and writable.
R/(W): The bit or field is readable and writable.
However, writing is only performed to flag clearing.
R: The bit or field is readable.
"R" is indicated for all reserved bits. When writing to the register, write the value under Initial Value in the bit chart to reserved bits or fields.
W: The bit or field is writable.

(5) Description
Describes the function of the bit or field and specifies the values for writing.

4. Description of Abbreviations

The abbreviations used in this manual are listed below.

- Abbreviations used in this manual

| Abbreviation | Description |
|---|---|
| ACIA | Asynchronous communication interface adapter |
| bps | Bits per second |
| CRC | Cyclic redundancy check |
| DMA | Direct memory access |
| DMAC | Direct memory access controller |
| GSM | Global System for Mobile Communications |
| Hi-Z | High impedance |
| IEBus | Inter Equipment Bus (IEBus is a trademark of NEC Electronics Corporation.) |
| I/O | Input/output |
| IrDA | Infrared Data Association |
| LSB | Least significant bit |
| MSB | Most significant bit |
| NC | No connection |
| PLL | Phase-locked loop |
| PWM | Pulse width modulation |
| SFR | Special function register |
| SIM | Subscriber Identity Module |
| UART | Universal asynchronous receiver/transmitter |
| VCO | Voltage-controlled oscillator |

5. List of Product Specifications

Below is a table listing the product specifications for each group.

| Item | | H8/38537 Group | |
|---|---|---|---|
| | | **Flash Memory** | **Mask ROM** |
| Memory | ROM | 32 K, 60 Kbytes | 16 K, 24 K, 32 K, 40 K, 48 K, 60 Kbytes |
| | RAM | 2 Kbytes | 1 Kbyte, 2 Kbytes |
| Operating voltage and operating frequency | 4.5 to 5.5 V | 16 MHz | 16 MHz |
| | 2.7 to 5.5 V | 16 MHz | 16 MHz |
| | 1.8 to 5.5 V | — | — |
| | 2.7 to 3.6 V | — | — |
| | 1.8 to 3.6 V | — | — |
| I/O ports | Input | 9 | 9 |
| | Output | — | — |
| | I/O | 55 | 55 |
| Timers | Clock (timer A) | 1 | 1 |
| | Reload (timer C) | 1 | 1 |
| | Compare (timer F) | 1 | 1 |
| | Capture (timer G) | 1 | 1 |
| | AEC | 1 | 1 |
| | WDT | — | — |
| | WDT (discrete) | 1 | 1 |
| SCI | UART/Synchronous | 2 ch | 2 ch |
| A-D (resolution × input channels) | | 10 bit × 8 ch | 10 bit × 8 ch |
| LCD | seg | 32 | 32 |
| | com | 4 | 4 |
| External interrupt (internal wakeup) | | 13(8) | 13(8) |
| Package | | FP-80A | FP-80A |
| | | TFP-80C | TFP-80C |
| Operating temperature | | Standard specifications: –20 to 75°C, WTR: –40 to 85°C | |

All trademarks and registered trademarks are the property of their respective owners.

# Contents


Section 1 Overview ..... 1
1.1 Features ..... 1
1.1.1 Application ..... 1
1.1.2 Overview of Specifications ..... 2
1.2 List of Products ..... 6
1.3 Block Diagram ..... 8
1.4 Pin Assignment ..... 9
1.5 Pin Functions ..... 10

Section 2 CPU ..... 15
2.1 Address Space and Memory Map ..... 16
2.2 Register Configuration ..... 23
2.2.1 General Registers ..... 24
2.2.2 Program Counter (PC) ..... 25
2.2.3 Condition-Code Register (CCR) ..... 25
2.3 Data Formats ..... 27
2.3.1 General Register Data Formats ..... 27
2.3.2 Memory Data Formats ..... 29
2.4 Instruction Set ..... 30
2.4.1 Table of Instructions Classified by Function ..... 30
2.4.2 Basic Instruction Formats ..... 40
2.5 Addressing Modes and Effective Address Calculation ..... 41
2.5.1 Addressing Modes ..... 41
2.5.2 Effective Address Calculation ..... 45
2.6 Basic Bus Cycle ..... 47
2.6.1 Access to On-Chip Memory (RAM, ROM) ..... 47
2.6.2 On-Chip Peripheral Modules ..... 48
2.7 CPU States ..... 49
2.8 Usage Notes ..... 51
2.8.1 Notes on Data Access to Empty Areas ..... 51
2.8.2 EEPMOV Instruction ..... 51
2.8.3 Bit-Manipulation Instruction ..... 51
2.8.4 Notes on Use of the EEPMOV Instruction ..... 57

Section 3 Exception Handling ..... 59
3.1 Overview ..... 59

3.2 Reset ... 59
3.2.1 Overview ... 59
3.2.2 Reset Sequence ... 59
3.2.3 Interrupt Immediately after Reset ... 60
3.3 Interrupts ... 61
3.3.1 Overview ... 61
3.3.2 Interrupt Control Registers ... 63
3.3.3 External Interrupts ... 73
3.3.4 Internal Interrupts ... 74
3.3.5 Interrupt Operations ... 74
3.3.6 Interrupt Response Time ... 79
3.4 Application Notes ... 80
3.4.1 Notes on Stack Area Use ... 80
3.4.2 Notes on Rewriting Port Mode Registers ... 81
3.4.3 Method for Clearing Interrupt Request Flags ... 83

Section 4 Clock Pulse Generators ... 85
4.1 Overview ... 85
4.1.1 Block Diagram ... 85
4.1.2 System Clock and Subclock ... 85
4.2 System Clock Generator ... 86
4.3 Subclock Generator ... 89
4.4 Prescalers ... 91
4.5 Note on Oscillators ... 92
4.5.1 Definition of Oscillation Stabilization Wait Time ... 93
4.5.2 Notes on Use of Crystal Oscillator Element (Excluding Ceramic Oscillator Element) ... 95

Section 5 Power-Down Modes ... 97
5.1 Overview ... 97
5.1.1 System Control Registers ... 100
5.2 Sleep Mode ... 104
5.2.1 Transition to Sleep Mode ... 104
5.2.2 Clearing Sleep Mode ... 105
5.2.3 Clock Frequency in Sleep (Medium-Speed) Mode ... 105
5.3 Standby Mode ... 105
5.3.1 Transition to Standby Mode ... 105
5.3.2 Clearing Standby Mode ... 106
5.3.3 Oscillator Settling Time after Standby Mode is Cleared ... 107
5.3.4 Standby Mode Transition and Pin States ... 107

5.3.5 Notes on External Input Signal Changes before/after Standby Mode ..... 108
5.4 Watch Mode ..... 110
5.4.1 Transition to Watch Mode ..... 110
5.4.2 Clearing Watch Mode ..... 110
5.4.3 Oscillator Settling Time after Watch Mode is Cleared ..... 110
5.4.4 Notes on External Input Signal Changes before/after Watch Mode ..... 111
5.5 Subsleep Mode ..... 111
5.5.1 Transition to Subsleep Mode ..... 111
5.5.2 Clearing Subsleep Mode ..... 111
5.6 Subactive Mode ..... 112
5.6.1 Transition to Subactive Mode ..... 112
5.6.2 Clearing Subactive Mode ..... 112
5.6.3 Operating Frequency in Subactive Mode ..... 112
5.7 Active (Medium-Speed) Mode ..... 113
5.7.1 Transition to Active (Medium-Speed) Mode ..... 113
5.7.2 Clearing Active (Medium-Speed) Mode ..... 113
5.7.3 Operating Frequency in Active (Medium-Speed) Mode ..... 113
5.8 Direct Transfer ..... 114
5.8.1 Overview of Direct Transfer ..... 114
5.8.2 Direct Transition Times ..... 115
5.8.3 Notes on External Input Signal Changes before/after Direct Transition ..... 117
5.9 Module Standby Mode ..... 118
5.9.1 Setting Module Standby Mode ..... 118
5.9.2 Clearing Module Standby Mode ..... 118
5.9.3 Usage Note ..... 120

Section 6 ROM ..... 121
6.1 Overview ..... 121
6.2 Flash Memory Overview ..... 121
6.2.1 Features ..... 121
6.2.2 Block Diagram ..... 122
6.2.3 Block Configuration ..... 123
6.2.4 Register Configuration ..... 125
6.3 Descriptions of Registers of the Flash Memory ..... 126
6.3.1 Flash Memory Control Register 1 (FLMCR1) ..... 126
6.3.2 Flash Memory Control Register 2 (FLMCR2) ..... 129
6.3.3 Erase Block Register (EBR) ..... 130
6.3.4 Flash Memory Power Control Register (FLPWCR) ..... 131
6.3.5 Flash Memory Enable Register (FENR) ..... 132

6.4 On-Board Programming Modes ..... 133
6.4.1 Boot Mode ..... 133
6.4.2 Programming/Erasing in User Program Mode ..... 136
6.5 Flash Memory Programming/Erasing ..... 137
6.5.1 Program/Program-Verify ..... 137
6.5.2 Erase/Erase-Verify ..... 140
6.5.3 Interrupt Handling when Programming/Erasing Flash Memory ..... 140
6.6 Program/Erase Protection ..... 142
6.6.1 Hardware Protection ..... 142
6.6.2 Software Protection ..... 142
6.6.3 Error Protection ..... 142
6.7 Programmer Mode ..... 143
6.7.1 Socket Adapter ..... 143
6.7.2 Programmer Mode Commands ..... 143
6.7.3 Memory Read Mode ..... 146
6.7.4 Auto-Program Mode ..... 149
6.7.5 Auto-Erase Mode ..... 151
6.7.6 Status Read Mode ..... 152
6.7.7 Status Polling ..... 154
6.7.8 Programmer Mode Transition Time ..... 155
6.7.9 Notes on Memory Programming ..... 155
6.8 Power-Down States for Flash Memory ..... 156

Section 7 RAM ..... 157
7.1 Overview ..... 157
7.1.1 Block Diagram ..... 157

Section 8 I/O Ports ..... 159
8.1 Overview ..... 159
8.2 Port 1 ..... 161
8.2.1 Overview ..... 161
8.2.2 Register Configuration and Description ..... 161
8.2.3 Pin Functions ..... 166
8.2.4 Pin States ..... 168
8.2.5 MOS Input Pull-Up ..... 168
8.3 Port 3 ..... 169
8.3.1 Overview ..... 169
8.3.2 Register Configuration and Description ..... 169
8.3.3 Pin Functions ..... 175
8.3.4 Pin States ..... 177

8.3.5 MOS Input Pull-Up ..... 177
8.4 Port 4 ..... 178
8.4.1 Overview ..... 178
8.4.2 Register Configuration and Description ..... 178
8.4.3 Pin Functions ..... 179
8.4.4 Pin States ..... 180
8.5 Port 5 ..... 181
8.5.1 Overview ..... 181
8.5.2 Register Configuration and Description ..... 181
8.5.3 Pin Functions ..... 184
8.5.4 Pin States ..... 184
8.5.5 MOS Input Pull-Up ..... 185
8.6 Port 6 ..... 185
8.6.1 Overview ..... 185
8.6.2 Register Configuration and Description ..... 186
8.6.3 Pin Functions ..... 187
8.6.4 Pin States ..... 188
8.6.5 MOS Input Pull-Up ..... 188
8.7 Port 7 ..... 189
8.7.1 Overview ..... 189
8.7.2 Register Configuration and Description ..... 189
8.7.3 Pin Functions ..... 191
8.7.4 Pin States ..... 191
8.8 Port 8 ..... 192
8.8.1 Overview ..... 192
8.8.2 Register Configuration and Description ..... 192
8.8.3 Pin Functions ..... 194
8.8.4 Pin States ..... 195
8.9 Port A ..... 195
8.9.1 Overview ..... 195
8.9.2 Register Configuration and Description ..... 196
8.9.3 Pin Functions ..... 197
8.9.4 Pin States ..... 198
8.10 Port B ..... 198
8.10.1 Overview ..... 198
8.10.2 Register Configuration and Description ..... 199
8.11 Input/Output Data Inversion Function ..... 200
8.11.1 Overview ..... 200
8.11.2 Register Configuration and Descriptions ..... 200
8.11.3 Note on Modification of Serial Port Control Register ..... 203

8.12 Application Note ..... 203
8.12.1 The Management of the Un-Use Terminal ..... 203

Section 9 Timers ..... 205
9.1 Overview ..... 205
9.2 Timer A ..... 207
9.2.1 Overview ..... 207
9.2.2 Register Descriptions ..... 209
9.2.3 Timer Operation ..... 213
9.2.4 Timer A Operation States ..... 214
9.2.5 Application Note ..... 214
9.3 Timer C ..... 215
9.3.1 Overview ..... 215
9.3.2 Register Descriptions ..... 217
9.3.3 Timer Operation ..... 220
9.3.4 Timer C Operation States ..... 222
9.3.5 Usage Note ..... 223
9.4 Timer F ..... 224
9.4.1 Overview ..... 224
9.4.2 Register Descriptions ..... 227
9.4.3 CPU Interface ..... 235
9.4.4 Operation ..... 237
9.4.5 Application Notes ..... 241
9.5 Timer G ..... 245
9.5.1 Overview ..... 245
9.5.2 Register Descriptions ..... 247
9.5.3 Noise Canceler ..... 253
9.5.4 Operation ..... 255
9.5.5 Application Notes ..... 260
9.5.6 Timer G Application Example ..... 264
9.6 Watchdog Timer ..... 265
9.6.1 Overview ..... 265
9.6.2 Register Descriptions ..... 266
9.6.3 Timer Operation ..... 271
9.6.4 Watchdog Timer Operation States ..... 272
9.7 Asynchronous Event Counter (AEC) ..... 273
9.7.1 Overview ..... 273
9.7.2 Register Descriptions ..... 275
9.7.3 Operation ..... 280
9.7.4 Asynchronous Event Counter Operation Modes ..... 282

9.7.5 Application Notes ..... 282

Section 10 Serial Communication Interface ..... 285
10.1 Overview ..... 285
10.1.1 Features ..... 285
10.1.2 Block Diagram ..... 287
10.1.3 Pin Configuration ..... 288
10.1.4 Register Configuration ..... 288
10.2 Register Descriptions ..... 289
10.2.1 Receive Shift Register (RSR) ..... 289
10.2.2 Receive Data Register (RDR) ..... 289
10.2.3 Transmit Shift Register (TSR) ..... 290
10.2.4 Transmit Data Register (TDR) ..... 290
10.2.5 Serial Mode Register (SMR) ..... 291
10.2.6 Serial Control Register 3 (SCR3) ..... 294
10.2.7 Serial Status Register (SSR) ..... 298
10.2.8 Bit Rate Register (BRR) ..... 303
10.2.9 Clock Stop Register 1 (CKSTPR1) ..... 308
10.2.10 Serial Port Control Register (SPCR) ..... 309
10.3 Operation ..... 312
10.3.1 Overview ..... 312
10.3.2 Operation in Asynchronous Mode ..... 317
10.3.3 Operation in Synchronous Mode ..... 326
10.4 Interrupts ..... 334
10.5 Application Notes ..... 335

Section 11 14-Bit PWM ..... 341
11.1 Overview ..... 341
11.1.1 Features ..... 341
11.1.2 Block Diagram ..... 342
11.1.3 Pin Configuration ..... 342
11.1.4 Register Configuration ..... 343
11.2 Register Descriptions ..... 343
11.2.1 PWM Control Register (PWCR) ..... 343
11.2.2 PWM Data Registers U and L (PWDRU, PWDRL) ..... 344
11.2.3 Clock Stop Register 2 (CKSTPR2) ..... 345
11.3 Operation ..... 346
11.3.1 Operation ..... 346
11.3.2 PWM Operation Modes ..... 347

Section 12 A/D Converter ..... 349
12.1 Overview ..... 349
12.1.1 Features ..... 349
12.1.2 Block Diagram ..... 350
12.1.3 Pin Configuration ..... 351
12.1.4 Register Configuration ..... 351
12.2 Register Descriptions ..... 352
12.2.1 A/D Result Registers (ADRRH, ADRRL) ..... 352
12.2.2 A/D Mode Register (AMR) ..... 352
12.2.3 A/D Start Register (ADSR) ..... 354
12.2.4 Clock Stop Register 1 (CKSTPR1) ..... 355
12.3 Operation ..... 356
12.3.1 A/D Conversion Operation ..... 356
12.3.2 Start of A/D Conversion by External Trigger Input ..... 356
12.3.3 A/D Converter Operation Modes ..... 357
12.4 Interrupts ..... 357
12.5 Typical Use ..... 358
12.6 Application Notes ..... 362
12.6.1 Application Notes ..... 362
12.6.2 Permissible Signal Source Impedance ..... 362
12.6.3 Influences on Absolute Precision ..... 363

Section 13 LCD Controller/Driver ..... 365
13.1 Overview ..... 365
13.1.1 Features ..... 365
13.1.2 Block Diagram ..... 366
13.1.3 Pin Configuration ..... 367
13.1.4 Register Configuration ..... 367
13.2 Register Descriptions ..... 368
13.2.1 LCD Port Control Register (LPCR) ..... 368
13.2.2 LCD Control Register (LCR) ..... 370
13.2.3 LCD Control Register 2 (LCR2) ..... 372
13.2.4 Clock Stop Register 2 (CKSTPR2) ..... 374
13.3 Operation ..... 375
13.3.1 Settings Up to LCD Display ..... 375
13.3.2 Relationship between LCD RAM and Display ..... 378
13.3.3 Luminance Adjustment Function ($V_0$ Pin) ..... 381
13.3.4 Low-Power-Consumption LCD Drive System ..... 382
13.3.5 Operation in Power-Down Modes ..... 386
13.3.6 Boosting the LCD Drive Power Supply ..... 388

Section 14 Power Supply Circuit ..... 389
14.1 Overview ..... 389
14.2 When Using Internal Power Supply Step-Down Circuit ..... 389
14.3 When Not Using Internal Power Supply Step-Down Circuit ..... 390

Section 15 List of Registers ..... 391
15.1 Register Addresses (Address Order) ..... 392
15.2 Register Bits ..... 395
15.3 Register States in Each Operating Mode ..... 398

Section 16 Electrical Characteristics ..... 401
16.1 Absolute Maximum Ratings ..... 401
16.2 Electrical Characteristics ..... 402
16.2.1 Power Supply Voltage and Operating Ranges ..... 402
16.2.2 DC Characteristics ..... 404
16.2.3 AC Characteristics ..... 412
16.2.4 A/D Converter Characteristics ..... 414
16.2.5 LCD Characteristics ..... 416
16.2.6 Flash Memory Characteristics ..... 417
16.3 Operation Timing ..... 419
16.4 Output Load Circuit ..... 422
16.5 Resonator ..... 422
16.6 Usage Note ..... 423

Appendix ..... 425
A. Instruction Set ..... 425
A.1 Instruction List ..... 425
A.2 Operation Code Map ..... 440
A.3 Number of Execution States ..... 443
A.4 Combinations of Instructions and Addressing Modes ..... 454
B. I/O Port Block Diagrams ..... 455
B.1 Block Diagrams of Port 1 ..... 455
B.2 Block Diagrams of Port 3 ..... 459
B.3 Block Diagrams of Port 4 ..... 467
B.4 Block Diagram of Port 5 ..... 471
B.5 Block Diagram of Port 6 ..... 472
B.6 Block Diagram of Port 7 ..... 473
B.7 Block Diagrams of Port 8 ..... 474
B.8 Block Diagram of Port A ..... 475
B.9 Block Diagram of Port B ..... 476

C. Port States in the Different Processing States ..... 477
D. List of Product Codes ..... 478
E. Package Dimensions ..... 480

Index .....483

# Section 1 Overview

## 1.1 Features

Microcontrollers of the H8/38537 Group are CISC (complex instruction set computer) microcontrollers whose core is an H8/300H CPU, which has an internal 32-bit architecture. The H8/300H CPU provides upward compatibility with the H8/300 CPUs of other Renesas Technology-original microcontrollers.

As peripheral functions, each LSI of this Group includes various timer functions that realize low-cost configurations for end systems. The power consumption of these modules can be kept down dynamically by power-down mode.

### 1.1.1 Application

Examples of the applications of this LSI include motor control, power meter, and health equipment.

### 1.1.2 Overview of Specifications

Table 1.1 lists the functions of H8/38537 Group products in outline.

**Table 1.1 Overview of Functions**

| Classification | Module/ Function | Description |
|---|---|---|
| Memory | ROM | • ROM lineup: Flash memory version and mask Rom version<br>• ROM capacity: 16 K, 24 K, 32 K, 40 K, 48 K, and 60 Kbytes |
| | RAM | • RAM capacity: 1 Kbyte and 2 Kbytes |
| CPU | CPU | • H8/300H CPU (CISC type)<br>Upward compatibility for H8/300 CPU at object level<br>• Sixteen 16-bit general registers<br>• Eight addressing modes<br>• 64-Kbyte address space<br>Program: 64 Kbytes available<br>Data: 64 Kbytes available<br>• 62 basic instructions, classifiable as bit arithmetic and logic instructions, multiply and divide instructions, bit manipulation instructions, and others<br>• Minimum instruction execution time: 400 ns (for an ADD instruction while system clock $\phi$ = 5 MHz and $V_{CC}$ = 2.7 to 3.6 V)<br>• On-chip multiplier (16 $\times$ 16 $\rightarrow$ 32 bits) |
| | Operating mode | • Normal mode |
| | MCU operating mode | Mode: Single-chip mode<br>• Low power consumption state (transition driven by the SLEEP instruction) |
| Interrupt (source) | Interrupt controller (INTC) | • Thirteen external interrupt pins ($\overline{IRQ4}$ to $\overline{IRQ0}$, $\overline{WKP7}$ to $\overline{WKP0}$)<br>• 23 internal interrupt sources<br>• Independent vector addresses |

| Classification | Module/ Function | Description |
|---|---|---|
| Clock | Clock pulse generator (CPG) | • Two clock generation circuits available<br>• Separate clock signals are provided for each of functional modules<br>• Includes frequency division circuit, so the operating frequency is selectable<br>• Seven low-power-consumption modes: Active (medium speed) mode, sleep (high speed or medium speed) mode, subactive mode, subsleep mode, standby mode, and watch mode |
| A/D converter | A/D converter (ADC) | • 10-bit resolution × eight input channels<br>• Sample and hold function included<br>• Conversion time:<br>12.4 μs per channel (with φ at 5-MHz operation)<br>• A/D conversion can be started by external trigger input |
| Timer | 14-bit PWM | • 14 bits × one channel<br>• Four conversion periods selectable<br>• Pulse division method for less ripple |
| | Timer A | • 8-bit timer<br>• Interval timer functionality: Eight internal clock sources are selectable<br>• Clock time base functionality: Four overflow periods are selectable<br>• Generates an interrupt upon overflow<br>• Timer output clock signals are selectable |
| | Timer C | • 8-bit timer<br>• Eight clocks are selectable<br>• Auto-reload function supported<br>• Generates an interrupt upon overflow<br>• Up/down-counter switching is possible |
| | Timer F | • 16-bit timer (also can be used as two independent 8-bit timers)<br>• Five clocks are selectable<br>• Output compare function supported<br>• Toggle output function supported<br>• Two interrupt sources: Compare match and overflow |

| Classification | Module/ Function | Description |
|---|---|---|
| Timer | Timer G | • 8-bit timer<br>• Four counter input clocks are selectable<br>• Input capture functions supported (a built-in noise canceller)<br>• Level detection at counter overflow is possible<br>• Counter clearing option<br>• Two interrupt sources: Input capture and overflow |
| | Asynchron-ous event counter (AEC) | • 16-bit pulse timer (also can be used as 8 bits × two channels)<br>• Can count asynchronously-input external events |
| Watchdog timer | Watchdog timer (WDT) | 8 bits × one channel (selectable from two counter input clocks) |
| Serial interface | Serial communi-cation interface 3 (SCI3) | • Two serial communication interfaces, SCI3-1 and SCI3-2, that have identical functions<br>• For both asynchronous and clock synchronous serial communications<br>• Full-duplex communications capability<br>• Select the desired bit rate<br>• Six interrupt sources |
| I/O ports | | • Nine CMOS input-only pins<br>• 55 CMOS input/output pins<br>• 32 pull-up resistors |
| LCD (Liquid Crystal Display) drive | LCD controller/ driver | • A maximum of 32 segment pins and four common pins<br>• Choice of four duty cycles (static, 1/2, 1/3, or 1/4)<br>• LCD RAM capacity: 8 bits × 32 bytes (256 bits)<br>• Word access to LCD RAM<br>• All eight segment output pins can be used individually as port pins<br>• Common output pins not used because of the duty cycle can be used for common double-buffering (parallel connection)<br>• Display possible in operating modes other than standby mode<br>• Choice of 11 frame frequencies<br>• Built-in power supply split-resistance, supplying LCD drive power<br>• A or B waveform selectable by software |

| Classification | Module/ Function | Description |
|---|---|---|
| Internal power supply step-down circuit | Power supply circuit | • The internal power supply can be fixed at a constant level of approximately 3.0 V to 3.2 V, independently of the voltage of the power supply connected to the external $V_{CC}$ pin<br>• It is also possible to use the same level of external power supply voltage and internal power supply voltage without using the internal power supply step-down circuit |
| Package | | • QFP-80: package code: FP-80A<br>(package dimensions: 14 × 14 mm, pin pitch: 0.65 mm)<br>• TQFP-80: package code: TFP-80C<br>(package dimensions: 12 × 12 mm, pin pitch: 0.50 mm) |
| Operating frequency/ Power supply voltage | | • Operating frequency: 2 to 16 MHz<br>• Power supply voltage:<br>Vcc = 2.7 to 5.5 V, AVcc = 2.7 to 5.5 V<br>• Supply current:<br>Flash memory version: 4.9 mA (typ.)<br>(Vcc = 5.0 V, AVcc = 5.0 V, ϕ = 10 MHz)<br>Mask ROM version: 4.0 mA (typ.)<br>(Vcc = 5.0 V, AVcc = 5.0 V, ϕ = 10 MHz) |
| Operating peripheral temperature (°C) | | • –20 to +75°C (regular specifications)<br>• –40 to +85°C (wide-range specifications) |

## 1.2 List of Products

Table 1.2 and figure 1.1 show the list of products and the structure of a product number, respectively.

**Table 1.2 List of Products**

| Group | Product Type | ROM Size | RAM Size | Package | Remarks |
|---|---|---|---|---|---|
| H8/38537 Group | HD64F38537 | 60 Kbytes | 2 Kbytes | FP-80A, TFP-80C | Flash memory version |
| | HD64338537 | 60 Kbytes | 2 Kbytes | | Mask ROM version |
| | HD64338536 | 48 Kbytes | 2 Kbytes | | Mask ROM version |
| | HD64338535 | 40 Kbytes | 2 Kbytes | | Mask ROM version |
| | HD64F38534 | 32 Kbytes | 2 Kbytes | | Flash memory version |
| | HD64338534 | 32 Kbytes | 2 Kbytes | | Mask ROM version |
| | HD64338533 | 24 Kbytes | 1 Kbyte | | Mask ROM version |
| | HD64338532 | 16 Kbytes | 1 Kbyte | | Mask ROM version |

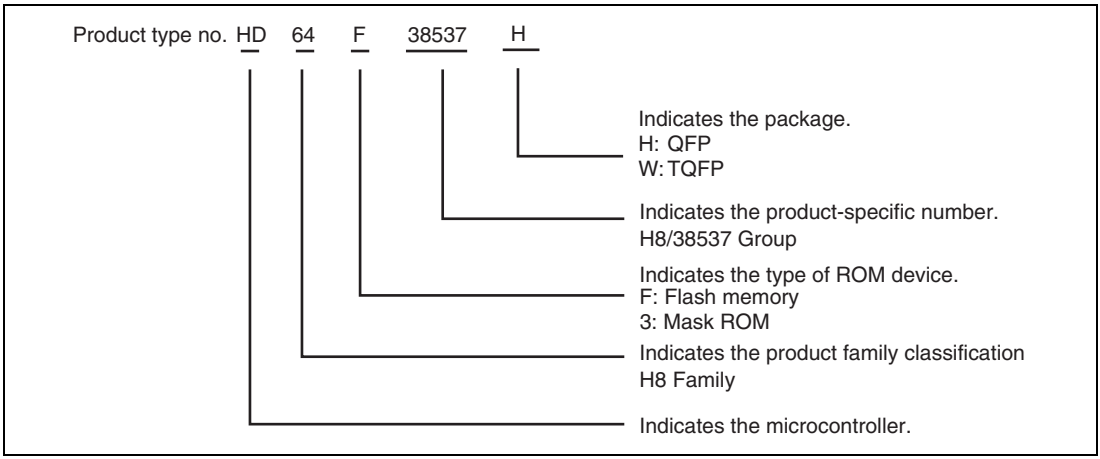
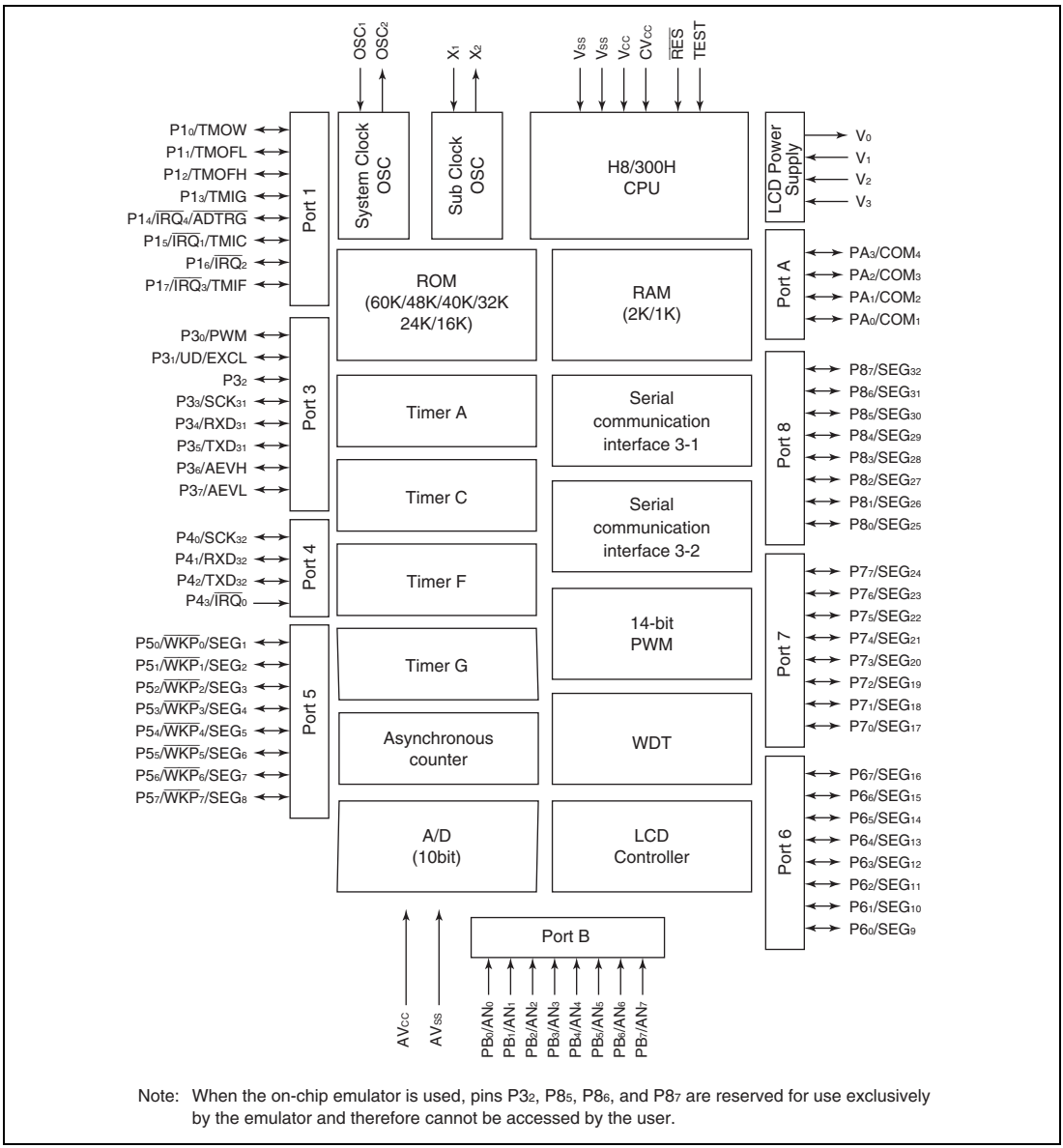



**Figure 1.1 How to Read the Product Name Code**

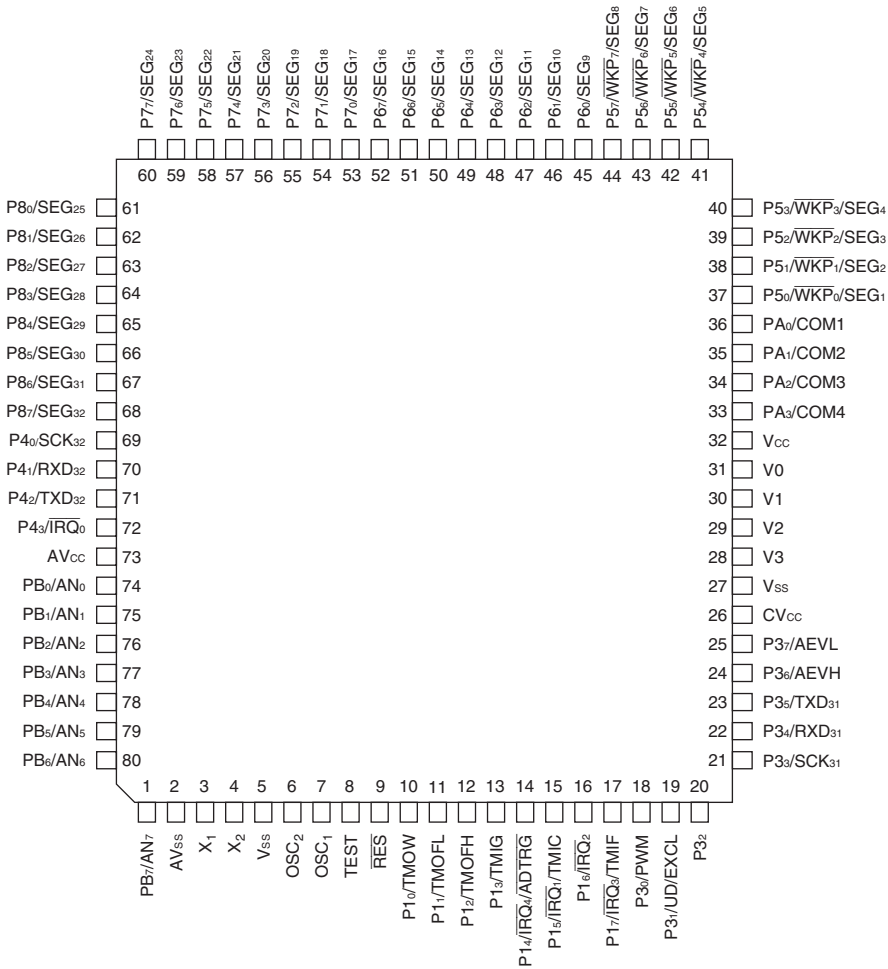
## 1.3 Block Diagram




Note: When the on-chip emulator is used, pins P3$_2$, P8$_5$, P8$_6$, and P8$_7$ are reserved for use exclusively by the emulator and therefore cannot be accessed by the user.

**Figure 1.2 Block Diagram of H8/38537 Group**

## 1.4 Pin Assignment




Notes: When the on-chip emulator is used, pins $P3_2$, $P8_5$, $P8_6$, and $P8_7$ are reserved for use exclusively by the emulator and therefore cannot be accessed by the user.

**Figure 1.3 Pin Assignment of H8/38537 Group (FP-80A and TFP-80C)**

## 1.5 Pin Functions

### Table 1.3 Pin Functions

| Type | Symbol | Pin No. FP-80A, TFP-80C | I/O | Name and Functions |
|---|---|---|---|---|
| Power source pins | $V_{CC}$<br>$CV_{CC}$ | 32<br>26 | Input | **Power supply:** All $V_{CC}$ pins should be connected to the system power supply.<br>See section 14, Power Supply Circuit, for a $CV_{CC}$ pin. |
| | $V_{SS}$ | 5<br>27 | Input | **Ground:** All $V_{SS}$ pins should be connected to the system power supply (0 V). |
| | $AV_{CC}$ | 73 | Input | **Analog power supply:** This is the power supply pin for the A/D converter. When the A/D converter is not used, connect this pin to the system power supply. |
| | $AV_{SS}$ | 2 | Input | **Analog ground:** This is the A/D converter ground pin. It should be connected to the system power supply (0V). |
| | V0 | 31 | Output | **LCD power supply:** These are the power supply pins for the LCD controller/driver. They incorporate a power supply split-resistance, and are normally used with V0 and V1 shorted. |
| | V1<br>V2<br>V3 | 30<br>29<br>28 | Input | |
| Clock pins | OSC1 | 7 | Input | These pins connect to a crystal or ceramic oscillator, or can be used to input an external clock. See section 4, Clock Pulse Generators, for a typical connection diagram. |
| | OSC2 | 6 | Output | |
| | X1 | 3 | Input | These pins connect to a 32.768-kHz or 38.4-kHz crystal oscillator.<br>See section 4, Clock Pulse Generators, for a typical connection diagram. |
| | X2 | 4 | Output | |
| | EXCL | 19 | Input | This pin connects to a 32.768-kHz or 38.4-kHz external clock. See section 4, Clock Pulse Generators, for typical connection diagram. |

| Type | Symbol | Pin No. FP-80A, TFP-80C | I/O | Name and Functions |
|---|---|---|---|---|
| System control | $\overline{RES}$ | 9 | Input | **Reset:** When this pin is driven low, the chip is reset |
| | TEST | 8 | Input | **Test pin:** This pin is reserved and cannot be used. It should be connected to $V_{SS}$. |
| Interrupt pins | $\overline{IRQ}$0<br>$\overline{IRQ}$1<br>$\overline{IRQ}$2<br>$\overline{IRQ}$3<br>$\overline{IRQ}$4 | 72<br>15<br>16<br>17<br>14 | Input | **IRQ interrupt request 0 to 4:** These are input pins for edge-sensitive external interrupts, with a selection of rising or falling edge |
| | $\overline{WKP}$7 to $\overline{WKP}$0 | 44 to 37 | Input | **Wakeup interrupt request 0 to 7:** These are input pins for rising or falling-edge-sensitive external interrupts. |
| Timer pins | TMOW | 10 | Output | **Clock output:** This is an output pin for waveforms generated by the timer A output circuit. |
| | AEVL<br>AEVH | 25<br>24 | Input | **Asynchronous event counter event input:** This is an event input pin for input to the asynchronous event counter. |
| | TMIC | 15 | Input | **Timer C event input:** This is an event input pin for input to the timer C counter. |
| | UD | 19 | Input | **Timer C up/down select:** This pin selects up- or down-counting for the timer C counter. The counter operates as a down-counter when this pin is high, and as an up-counter when low. |
| | TMIF | 17 | Input | **Timer F event input:** This is an event input pin for input to the timer F counter. |
| | TMOFL | 11 | Output | **Timer FL output:** This is an output pin for waveforms generated by the timer FL output compare function. |
| | TMOFH | 12 | Output | **Timer FH output:** This is an output pin for waveforms generated by the timer FH output compare function. |
| | TMIG | 13 | Input | **Timer G capture input:** This is an input pin for timer G input capture. |
| 14-bit PWM pin | PWM | 18 | Output | **14-bit PWM output:** This is an output pin for waveforms generated by the 14-bit PWM |

| Type | Symbol | Pin No. FP-80A, TFP-80C | I/O | Name and Functions |
|---|---|---|---|---|
| I/O ports | PB7 to PB0 | 1, 80 to 74 | Input | **Port B:** This is an 8-bit input port. |
| | P43 | 72 | Input | **Port 4 (bit 3):** This is a 1-bit input port. |
| | P42 to P40 | 71 to 69 | I/O | **Port 4 (bits 2 to 0):** This is a 3-bit I/O port. Input or output can be designated for each bit by means of port control register 4 (PCR4). |
| | PA3 to PA0 | 33 to 36 | I/O | **Port A:** This is a 4-bit I/O port. Input or output can be designated for each bit by means of port control register A (PCRA). |
| | P17 to P10 | 17 to 10 | I/O | **Port 1:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 1 (PCR1). |
| | P37 to P30 | 25 to 18 | I/O | **Port 3:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 3 (PCR3). When the on-chip emulator is used, pin P32 is reserved for use exclusively by the emulator and therefore cannot be accessed by the user. With the flash memory version, pull up pin P32 to high level to cancel a reset in the in the user mode. |
| | P57 to P50 | 44 to 37 | I/O | **Port 5:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 5 (PCR5). |
| | P67 to P60 | 52 to 45 | I/O | **Port 6:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 6 (PCR6). |
| | P77 to P70 | 60 to 53 | I/O | **Port 7:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 7 (PCR7). |
| | P87 to P80 | 68 to 61 | I/O | **Port 8:** This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 8 (PCR8). When the on-chip emulator is used, pins P85, P86, and P87 are reserved for use exclusively by the emulator and therefore cannot be accessed by the user. |

| Type | Symbol | Pin No. FP-80A, TFP-80C | I/O | Name and Functions |
|---|---|---|---|---|
| Serial communi-cation interface (SCI) | RXD31 | 22 | Input | **SCI31 receive data input:** This is the SCI31 data input pin. |
| | TXD31 | 23 | Output | **SCI31 transmit data output:** This is the SCI31 data output pin. |
| | SCK31 | 21 | I/O | **SCI31 clock I/O:** This is the SCI31 clock I/O pin. |
| | RXD32 | 70 | Input | **SCI32 receive data input:** This is the SCI32 data input pin. |
| | TXD32 | 71 | Output | **SCI32 transmit data output:** This is the SCI32 data output pin. |
| | SCK32 | 69 | I/O | **SCI32 clock I/O:** This is the SCI32 clock I/O pin. |
| A/D converter | AN7 to AN0 | 1<br>80 to 74 | Input | **Analog input channels 7 to 0:** These are analog data input channels to the A/D converter |
| | $\overline{\text{ADTRG}}$ | 14 | Input | **A/D converter trigger input:** This is the external trigger input pin to the A/D converter |
| LCD controller/ driver | COM4 to COM1 | 33 to 36 | Output | **LCD common output:** These are the LCD common output pins. |
| | SEG32 to SEG1 | 68 to 37 | Output | **LCD segment output:** These are the LCD segment output pins. |

# Section 2 CPU

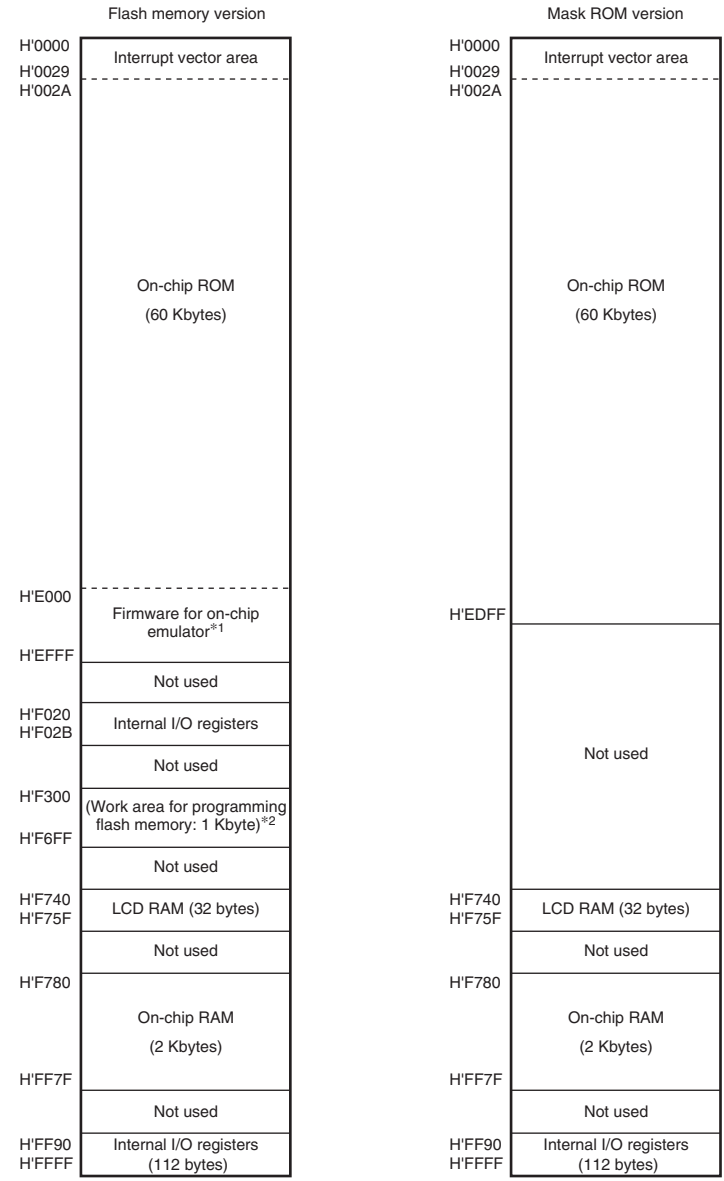
This LSI has an H8/300H CPU with an internal 32-bit architecture that is upward-compatible with the H8/300 CPU, and supports only normal mode, which has a 64-Kbyte address space.

- Upward-compatible with H8/300 CPUs
  Can execute H8/300 CPUs object programs
  Additional eight 16-bit extended registers
  32-bit transfer and arithmetic and logic instructions are added
  Signed multiply and divide instructions are added.
- General-register architecture
  Sixteen 16-bit general registers also usable as sixteen 8-bit registers and eight 16-bit registers, or eight 32-bit registers
- Sixty-two basic instructions
  8/16/32-bit data transfer and arithmetic and logic instructions
  Multiply and divide instructions
  Powerful bit-manipulation instructions
- Eight addressing modes
  Register direct [Rn]
  Register indirect [@ERn]
  Register indirect with displacement [@(d:16,ERn) or @(d:24,ERn)]
  Register indirect with post-increment or pre-decrement [@ERn+ or @–ERn]
  Absolute address [@aa:8, @aa:16, @aa:24]
  Immediate [#xx:8, #xx:16, or #xx:32]
  Program-counter relative [@(d:8,PC) or @(d:16,PC)]
  Memory indirect [@@aa:8]
- 64-Kbyte address space
- High-speed operation
  All frequently-used instructions execute in one or two states
  8/16/32-bit register-register add/subtract : 2 state
  8 × 8-bit register-register multiply : 14 states
  16 ÷ 8-bit register-register divide : 14 states
  16 × 16-bit register-register multiply : 22 states
  32 ÷ 16-bit register-register divide : 22 states

- Power-down state
  Transition to power-down state by SLEEP instruction

## 2.1 Address Space and Memory Map

The memory map of the H8/38537 is shown in figure 2.1 (1), that of the H8/38536 in figure 2.1 (2), that of the H8/38535 in figure 2.1 (3), that of the H8/38534 in figure 2.1 (4), that of the H8/38533 in figure 2.1 (5), and that of the H8/38532 in figure 2.1 (6).




Notes: 1. Not accessible by the user when the on-chip emulator is used.
2. A programming control program is used to program flash memory. Do not use a user program to perform programming when the on-chip emulator is used. This area is not used in the mask ROM version.

**Figure 2.1 (1) H8/38537 Memory Map**

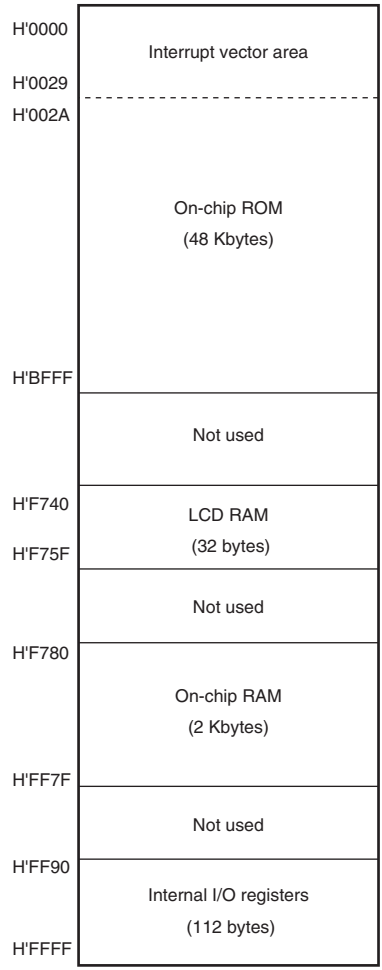



**Figure 2.1 (2) H8/38536 Memory Map**

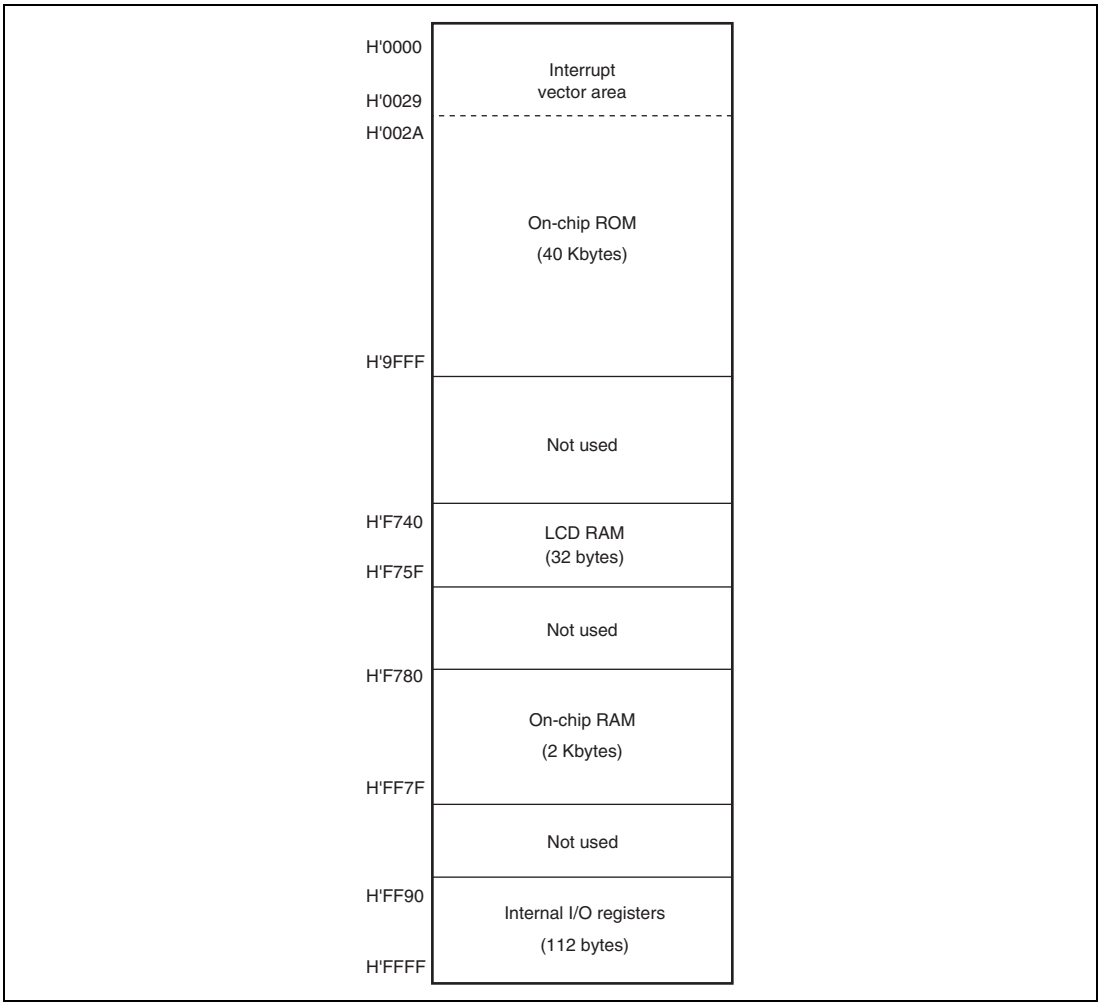
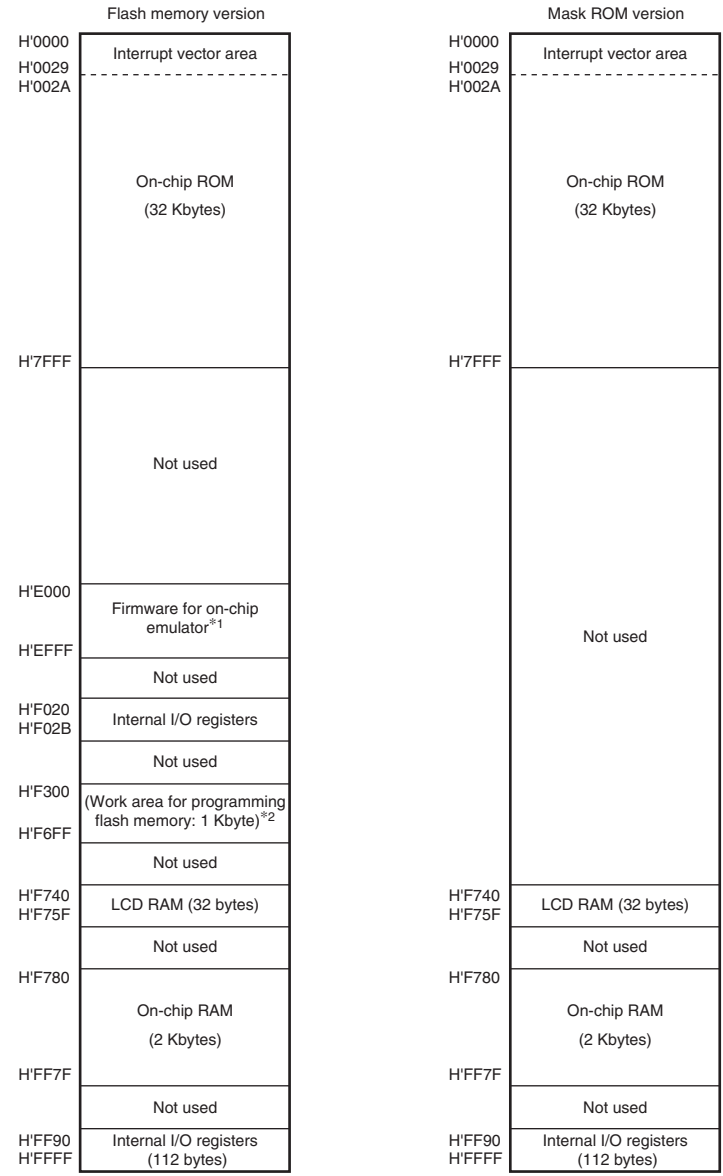



**Figure 2.1 (3) H8/38535 Memory Map**




Notes: 1. Not accessible by the user when the on-chip emulator is used.
2. A programming control program is used to program flash memory. Do not use a user program to perform programming when the on-chip emulator is used. This area is not used in the mask ROM version.

**Figure 2.1 (4) H8/38534 Memory Map**

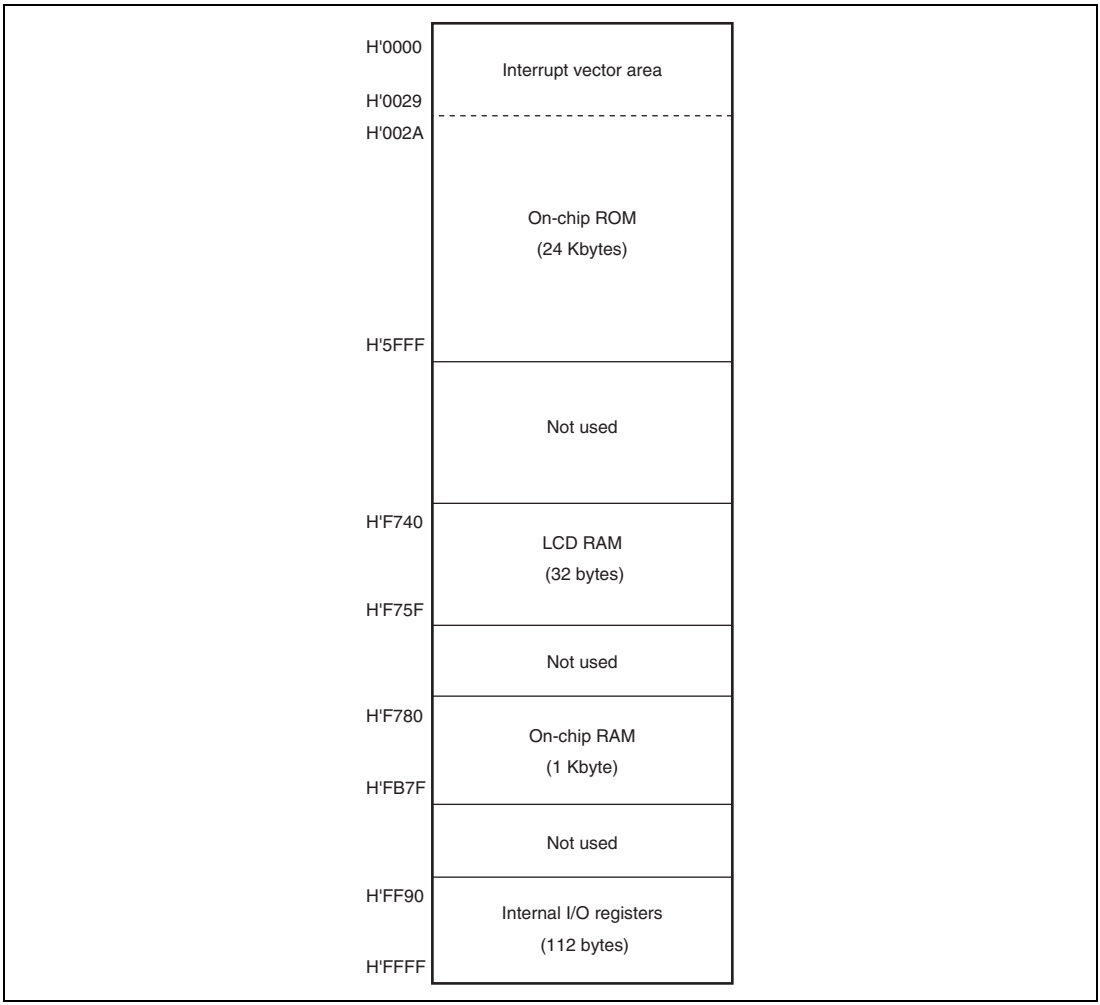



**Figure 2.1 (5) H8/38533 Memory Map**

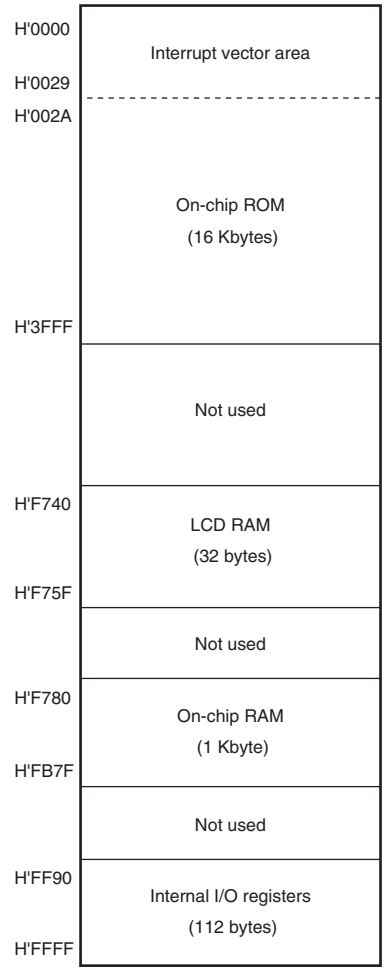



**Figure 2.1 (6) H8/38532 Memory Map**

## 2.2 Register Configuration

The H8/300H CPU has the internal registers shown in figure 2.2. There are two types of registers; general registers and control registers. The control registers are a 24-bit program counter (PC), and an 8-bit condition-code register (CCR).

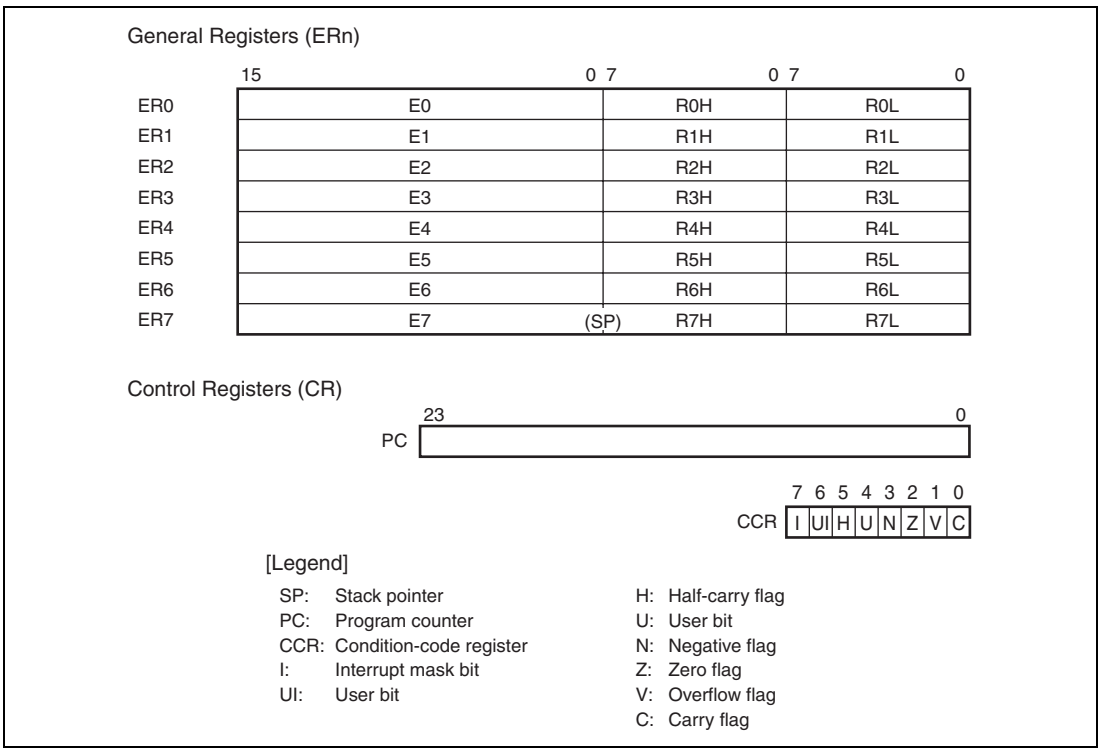



**Figure 2.2 CPU Registers**

### 2.2.1 General Registers

The H8/300H CPU has eight 32-bit general registers. These general registers are all functionally identical and can be used as both address registers and data registers. When a general register is used as a data register, it can be accessed as a 32-bit, 16-bit, or 8-bit register. Figure 2.3 illustrates the usage of the general registers. When the general registers are used as 32-bit registers or address registers, they are designated by the letters ER (ER0 to ER7).

The ER registers divide into 16-bit general registers designated by the letters E (E0 to E7) and R (R0 to R7). These registers are functionally equivalent, providing a maximum of sixteen 16-bit registers. The E registers (E0 to E7) are also referred to as extended registers.

The R registers divide into 8-bit registers designated by the letters RH (R0H to R7H) and RL (R0L to R7L). These registers are functionally equivalent, providing a maximum of sixteen 8-bit registers.

The usage of each register can be selected independently.

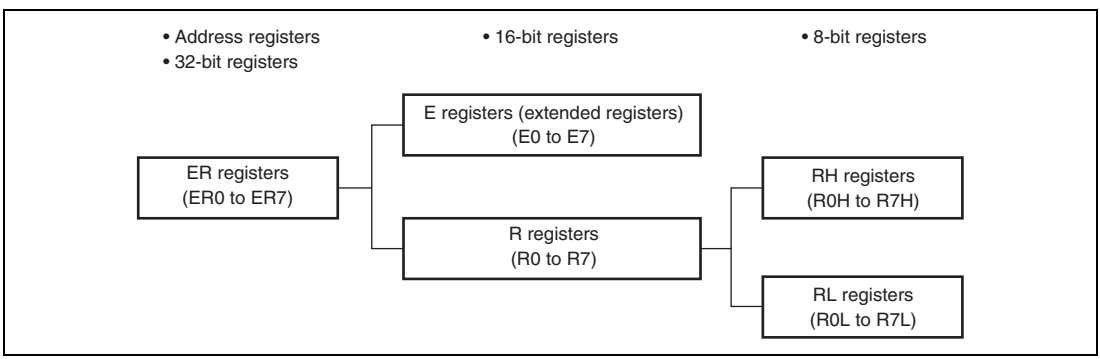



**Figure 2.3 Usage of General Registers**

General register ER7 has the function of the stack pointer (SP) in addition to its general-register function, and is used implicitly in exception handling and subroutine calls. Figure 2.4 shows the relationship between the stack pointer and the stack area.

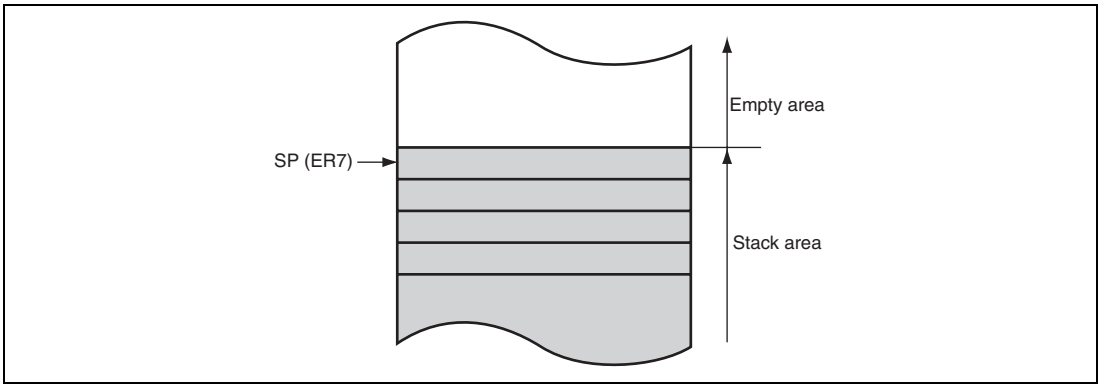



**Figure 2.4 Relationship between Stack Pointer and Stack Area**

### 2.2.2 Program Counter (PC)

This 24-bit counter indicates the address of the next instruction the CPU will execute. The length of all CPU instructions is 2 bytes (one word), so the least significant PC bit is ignored. (When an instruction is fetched, the least significant PC bit is regarded as 0). The PC is initialized when the start address is loaded by the vector address generated during reset exception-handling sequence.

### 2.2.3 Condition-Code Register (CCR)

This 8-bit register contains internal CPU status information, including an interrupt mask bit (I) and half-carry (H), negative (N), zero (Z), overflow (V), and carry (C) flags. The I bit is initialized to 1 by reset exception-handling sequence, but other bits are not initialized.

Some instructions leave flag bits unchanged. Operations can be performed on the CCR bits by the LDC, STC, ANDC, ORC, and XORC instructions. The N, Z, V, and C flags are used as branching conditions for conditional branch (Bcc) instructions.

For the action of each instruction on the flag bits, see appendix A.1, Instruction List.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | I | 1 | R/W | Interrupt Mask Bit<br>Masks interrupts other than NMI when set to 1. NMI is accepted regardless of the I bit setting. The I bit is set to 1 at the start of an exception-handling sequence. |
| 6 | UI | Undefined | R/W | User Bit<br>Can be written and read by software using the LDC, STC, ANDC, ORC, and XORC instructions. |
| 5 | H | Undefined | R/W | Half-Carry Flag<br>When the ADD.B, ADDX.B, SUB.B, SUBX.B, CMP.B, or NEG.B instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 3, and cleared to 0 otherwise. When the ADD.W, SUB.W, CMP.W, or NEG.W instruction is executed, the H flag is set to 1 if there is a carry or borrow at bit 11, and cleared to 0 otherwise. When the ADD.L, SUB.L, CMP.L, or NEG.L instruction is executed, the H flag is set to 1 if there is a carry or borrow at bit 27, and cleared to 0 otherwise. |
| 4 | U | Undefined | R/W | User Bit<br>Can be written and read by software using the LDC, STC, ANDC, ORC, and XORC instructions. |
| 3 | N | Undefined | R/W | Negative Flag<br>Stores the value of the most significant bit of data as a sign bit. |
| 2 | Z | Undefined | R/W | Zero Flag<br>Set to 1 to indicate zero data, and cleared to 0 to indicate non-zero data. |
| 1 | V | Undefined | R/W | Overflow Flag<br>Set to 1 when an arithmetic overflow occurs, and cleared to 0 at other times. |
| 0 | C | Undefined | R/W | Carry Flag<br>Set to 1 when a carry occurs, and cleared to 0 otherwise. Used by:<br>• Add instructions, to indicate a carry<br>• Subtract instructions, to indicate a borrow<br>• Shift and rotate instructions, to indicate a carry<br>The carry flag is also used as a bit accumulator by bit manipulation instructions. |

## 2.3 Data Formats

The H8/300H CPU can process 1-bit, 4-bit (BCD), 8-bit (byte), 16-bit (word), and 32-bit (longword) data. Bit-manipulation instructions operate on 1-bit data by accessing bit n (n = 0, 1, 2, …, 7) of byte operand data. The DAA and DAS decimal-adjust instructions treat byte data as two digits of 4-bit BCD data.

### 2.3.1 General Register Data Formats

Figure 2.5 shows the data formats in general registers.

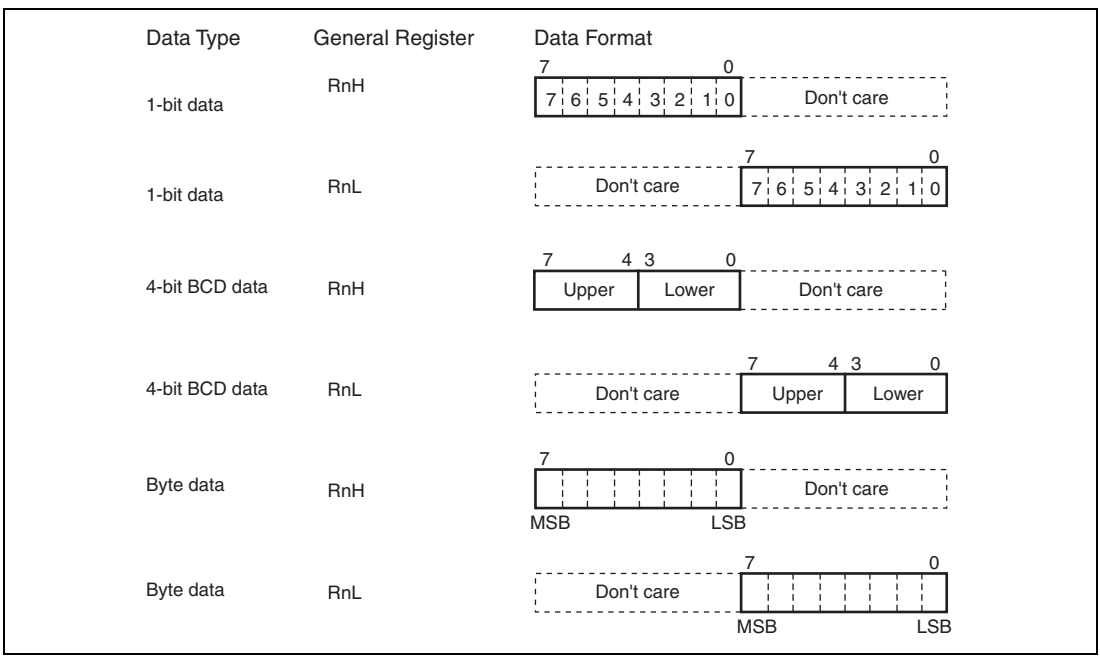



**Figure 2.5 General Register Data Formats (1)**

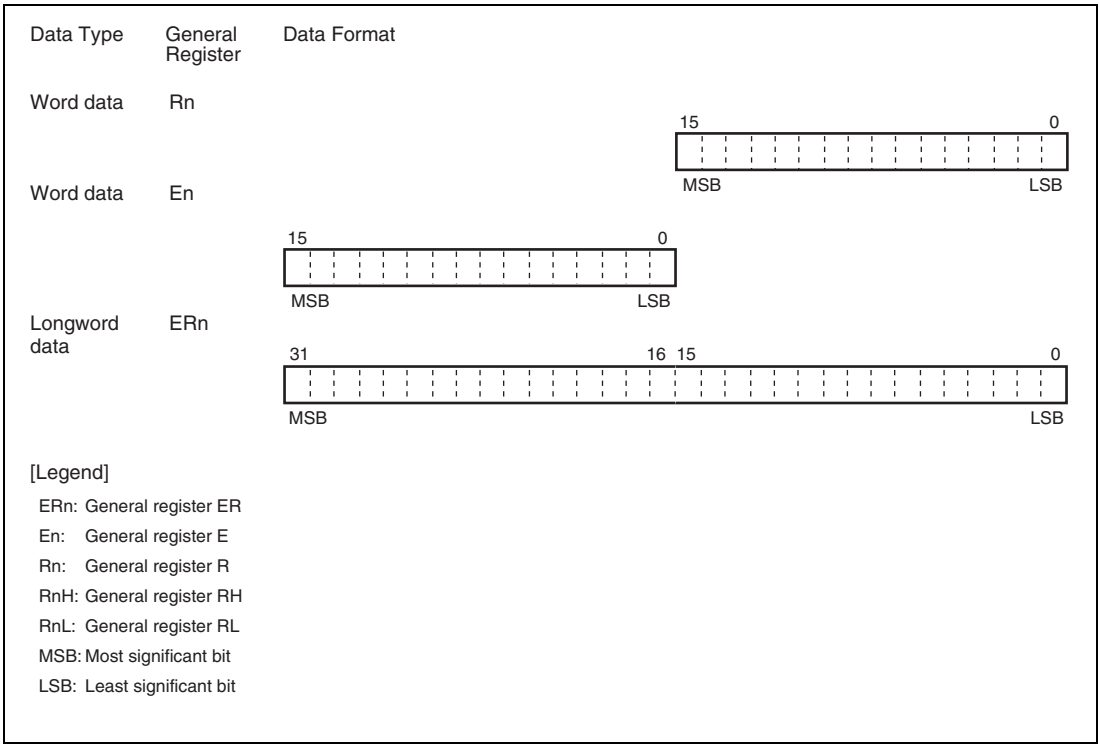



**Figure 2.5 General Register Data Formats (2)**

### 2.3.2 Memory Data Formats

Figure 2.6 shows the data formats in memory. The H8/300H CPU can access word data and longword data in memory, however word or longword data must begin at an even address. If an attempt is made to access word or longword data at an odd address, an address error does not occur, however the least significant bit of the address is regarded as 0, so access begins the preceding address. This also applies to instruction fetches.

When ER7 (SP) is used as an address register to access the stack area, the operand size should be word or longword.

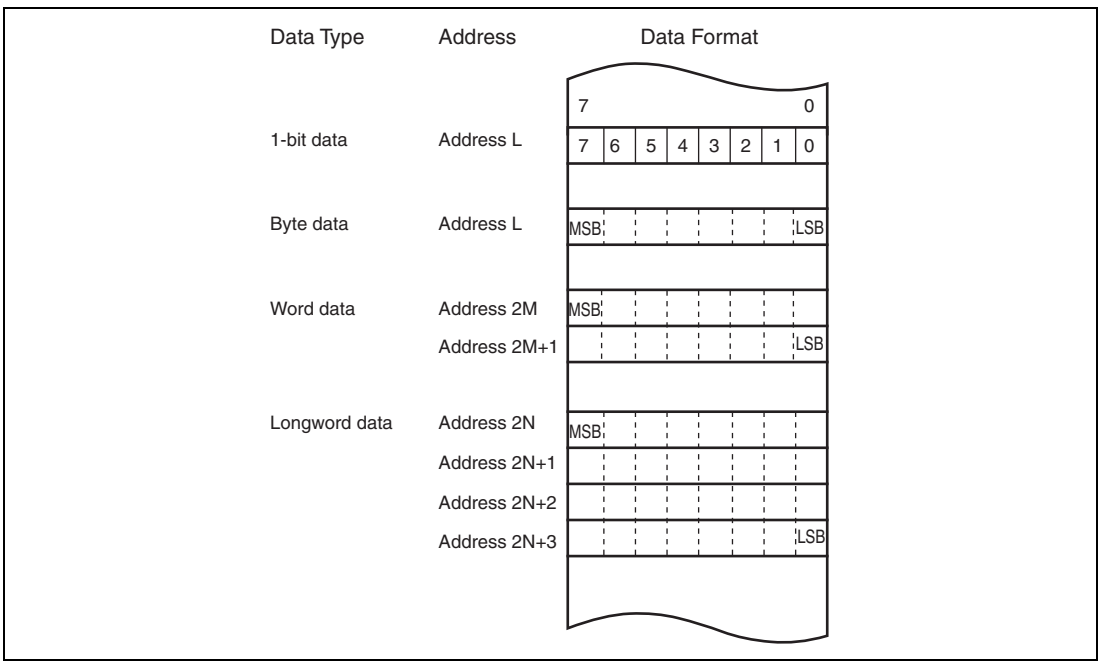



**Figure 2.6 Memory Data Formats**

## 2.4 Instruction Set

### 2.4.1 Table of Instructions Classified by Function

The H8/300H CPU has 62 instructions. Tables 2.2 to 2.9 summarize the instructions in each functional category. The notation used in tables 2.2 to 2.9 is defined in table 2.1.

**Table 2.1 Operation Notation**

| Symbol | Description |
|---|---|
| Rd | General register (destination)* |
| Rs | General register (source)* |
| Rn | General register* |
| ERn | General register (32-bit register or address register) |
| (EAd) | Destination operand |
| (EAs) | Source operand |
| CCR | Condition-code register |
| N | N (negative) flag in CCR |
| Z | Z (zero) flag in CCR |
| V | V (overflow) flag in CCR |
| C | C (carry) flag in CCR |
| PC | Program counter |
| SP | Stack pointer |
| #IMM | Immediate data |
| disp | Displacement |
| + | Addition |
| – | Subtraction |
| × | Multiplication |
| ÷ | Division |
| ∧ | Logical AND |
| ∨ | Logical OR |
| ⊕ | Logical XOR |
| → | Move |
| ¬ | NOT (logical complement) |

| Symbol | Description |
|---|---|
| :3/:8/:16/:24 | 3-, 8-, 16-, or 24-bit length |

Note: * General registers include 8-bit registers (R0H to R7H, R0L to R7L), 16-bit registers (R0 to R7, E0 to E7), and 32-bit registers/address register (ER0 to ER7).

**Table 2.2 Data Transfer Instructions**

| Instruction | Size* | Function |
|---|---|---|
| MOV | B/W/L | (EAs) → Rd, Rs → (EAd)<br>Moves data between two general registers or between a general register and memory, or moves immediate data to a general register. |
| MOVFPE | B | (EAs) → Rd<br>Cannot be used in this LSI. |
| MOVTPE | B | Rs → (EAs)<br>Cannot be used in this LSI. |
| POP | W/L | @SP+ → Rn<br>Pops a general register from the stack. POP.W Rn is identical to MOV.W @SP+, Rn. POP.L ERn is identical to MOV.L @SP+, ERn. |
| PUSH | W/L | Rn → @–SP<br>Pushes a general register onto the stack. PUSH.W Rn is identical to MOV.W Rn, @–SP. PUSH.L ERn is identical to MOV.L ERn, @–SP. |

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

**Table 2.3 Arithmetic Operations Instructions (1)**

| Instruction | Size* | Function |
|---|---|---|
| ADD<br>SUB | B/W/L | Rd ± Rs → Rd, Rd ± #IMM → Rd<br>Performs addition or subtraction on data in two general registers, or on immediate data and data in a general register (immediate byte data cannot be subtracted from byte data in a general register. Use the SUBX or ADD instruction.) |
| ADDX<br>SUBX | B | Rd ± Rs ± C → Rd, Rd ± #IMM ± C → Rd<br>Performs addition or subtraction with carry on byte data in two general registers, or on immediate data and data in a general register. |
| INC<br>DEC | B/W/L | Rd ± 1 → Rd, Rd ± 2 → Rd<br>Increments or decrements a general register by 1 or 2. (Byte operands can be incremented or decremented by 1 only.) |
| ADDS<br>SUBS | L | Rd ± 1 → Rd, Rd ± 2 → Rd, Rd ± 4 → Rd<br>Adds or subtracts the value 1, 2, or 4 to or from data in a 32-bit register. |
| DAA<br>DAS | B | Rd (decimal adjust) → Rd<br>Decimal-adjusts an addition or subtraction result in a general register by referring to the CCR to produce 4-bit BCD data. |
| MULXU | B/W | Rd × Rs → Rd<br>Performs unsigned multiplication on data in two general registers: either 8 bits × 8 bits → 16 bits or 16 bits × 16 bits → 32 bits. |
| MULXS | B/W | Rd × Rs → Rd<br>Performs signed multiplication on data in two general registers: either 8 bits × 8 bits → 16 bits or 16 bits × 16 bits → 32 bits. |
| DIVXU | B/W | Rd ÷ Rs → Rd<br>Performs unsigned division on data in two general registers: either 16 bits ÷ 8 bits → 8-bit quotient and 8-bit remainder or 32 bits ÷ 16 bits → 16-bit quotient and 16-bit remainder. |

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

**Table 2.3 Arithmetic Operations Instructions (2)**

| Instruction | Size* | Function |
|---|---|---|
| DIVXS | B/W | Rd ÷ Rs → Rd<br>Performs signed division on data in two general registers: either 16 bits ÷ 8 bits → 8-bit quotient and 8-bit remainder or 32 bits ÷ 16 bits → 16-bit quotient and 16-bit remainder. |
| CMP | B/W/L | Rd – Rs, Rd – #IMM<br>Compares data in a general register with data in another general register or with immediate data, and sets CCR bits according to the result. |
| NEG | B/W/L | 0 – Rd → Rd<br>Takes the two's complement (arithmetic complement) of data in a general register. |
| EXTU | W/L | Rd (zero extension) → Rd<br>Extends the lower 8 bits of a 16-bit register to word size, or the lower 16 bits of a 32-bit register to longword size, by padding with zeros on the left. |
| EXTS | W/L | Rd (sign extension) → Rd<br>Extends the lower 8 bits of a 16-bit register to word size, or the lower 16 bits of a 32-bit register to longword size, by extending the sign bit. |

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

**Table 2.4 Logic Operations Instructions**

| Instruction | Size* | Function |
|---|---|---|
| AND | B/W/L | Rd ∧ Rs → Rd, Rd ∧ #IMM → Rd<br>Performs a logical AND operation on a general register and another general register or immediate data. |
| OR | B/W/L | Rd ∨ Rs → Rd, Rd ∨ #IMM → Rd<br>Performs a logical OR operation on a general register and another general register or immediate data. |
| XOR | B/W/L | Rd ⊕ Rs → Rd, Rd ⊕ #IMM → Rd<br>Performs a logical exclusive OR operation on a general register and another general register or immediate data. |
| NOT | B/W/L | ¬ (Rd) → (Rd)<br>Takes the one's complement (logical complement) of general register contents. |

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

**Table 2.5 Shift Instructions**

| Instruction | Size* | Function |
|---|---|---|
| SHAL<br>SHAR | B/W/L | Rd (shift) → Rd<br>Performs an arithmetic shift on general register contents. |
| SHLL<br>SHLR | B/W/L | Rd (shift) → Rd<br>Performs a logical shift on general register contents. |
| ROTL<br>ROTR | B/W/L | Rd (rotate) → Rd<br>Rotates general register contents. |
| ROTXL<br>ROTXR | B/W/L | Rd (rotate) → Rd<br>Rotates general register contents through the carry flag. |

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

**Table 2.6 Bit Manipulation Instructions (1)**

| Instruction | Size* | Function |
|---|---|---|
| BSET | B | 1 → (<bit-No.> of <EAd>)<br>Sets a specified bit in a general register or memory operand to 1. The bit number is specified by 3-bit immediate data or the lower three bits of a general register. |
| BCLR | B | 0 → (<bit-No.> of <EAd>)<br>Clears a specified bit in a general register or memory operand to 0. The bit number is specified by 3-bit immediate data or the lower three bits of a general register. |
| BNOT | B | ¬ (<bit-No.> of <EAd>) → (<bit-No.> of <EAd>)<br>Inverts a specified bit in a general register or memory operand. The bit number is specified by 3-bit immediate data or the lower three bits of a general register. |
| BTST | B | ¬ (<bit-No.> of <EAd>) → Z<br>Tests a specified bit in a general register or memory operand and sets or clears the Z flag accordingly. The bit number is specified by 3-bit immediate data or the lower three bits of a general register. |
| BAND | B | C ∧ (<bit-No.> of <EAd>) → C<br>ANDs the carry flag with a specified bit in a general register or memory operand and stores the result in the carry flag. |
| BIAND | B | C ∧ ¬ (<bit-No.> of <EAd>) → C<br>ANDs the carry flag with the inverse of a specified bit in a general register or memory operand and stores the result in the carry flag.<br>The bit number is specified by 3-bit immediate data. |
| BOR | B | C ∨ (<bit-No.> of <EAd>) → C<br>ORs the carry flag with a specified bit in a general register or memory operand and stores the result in the carry flag. |
| BIOR | B | C ∨ ¬ (<bit-No.> of <EAd>) → C<br>ORs the carry flag with the inverse of a specified bit in a general register or memory operand and stores the result in the carry flag.<br>The bit number is specified by 3-bit immediate data. |

Note: * Refers to the operand size.
B: Byte

**Table 2.6 Bit Manipulation Instructions (2)**

| Instruction | Size* | Function |
|---|---|---|
| BXOR | B | C ⊕ (<bit-No.> of <EAd>) → C<br>XORs the carry flag with a specified bit in a general register or memory operand and stores the result in the carry flag. |
| BIXOR | B | C ⊕ ¬ (<bit-No.> of <EAd>) → C<br>XORs the carry flag with the inverse of a specified bit in a general register or memory operand and stores the result in the carry flag.<br>The bit number is specified by 3-bit immediate data. |
| BLD | B | (<bit-No.> of <EAd>) → C<br>Transfers a specified bit in a general register or memory operand to the carry flag. |
| BILD | B | ¬ (<bit-No.> of <EAd>) → C<br>Transfers the inverse of a specified bit in a general register or memory operand to the carry flag.<br>The bit number is specified by 3-bit immediate data. |
| BST | B | C → (<bit-No.> of <EAd>)<br>Transfers the carry flag value to a specified bit in a general register or memory operand. |
| BIST | B | ¬ C → (<bit-No.> of <EAd>)<br>Transfers the inverse of the carry flag value to a specified bit in a general register or memory operand.<br>The bit number is specified by 3-bit immediate data. |

Note: * Refers to the operand size.
B: Byte

**Table 2.7 Branch Instructions**

| Instruction | Size | Function |
|---|---|---|
| Bcc* | — | Branches to a specified address if a specified condition is true. The branching conditions are listed below. |

| Mnemonic | Description | Condition |
|---|---|---|
| BRA(BT) | Always (true) | Always |
| BRN(BF) | Never (false) | Never |
| BHI | High | $C \vee Z = 0$ |
| BLS | Low or same | $C \vee Z = 1$ |
| BCC(BHS) | Carry clear (high or same) | $C = 0$ |
| BCS(BLO) | Carry set (low) | $C = 1$ |
| BNE | Not equal | $Z = 0$ |
| BEQ | Equal | $Z = 1$ |
| BVC | Overflow clear | $V = 0$ |
| BVS | Overflow set | $V = 1$ |
| BPL | Plus | $N = 0$ |
| BMI | Minus | $N = 1$ |
| BGE | Greater or equal | $N \oplus V = 0$ |
| BLT | Less than | $N \oplus V = 1$ |
| BGT | Greater than | $Z \vee (N \oplus V) = 0$ |
| BLE | Less or equal | $Z \vee (N \oplus V) = 1$ |

| | | |
|---|---|---|
| JMP | — | Branches unconditionally to a specified address. |
| BSR | — | Branches to a subroutine at a specified address. |
| JSR | — | Branches to a subroutine at a specified address. |
| RTS | — | Returns from a subroutine |

Note: * Bcc is the general name for conditional branch instructions.

**Table 2.8 System Control Instructions**

| Instruction | Size* | Function |
|---|---|---|
| RTE | — | Returns from an exception-handling routine. |
| SLEEP | — | Causes a transition to a power-down state. |
| LDC | B/W | (EAs) → CCR<br>Moves the source operand contents to the CCR. The CCR size is one byte, but in transfer from memory, data is read by word access. |
| STC | B/W | CCR → (EAd)<br>Transfers the CCR contents to a destination location. The condition code register size is one byte, but in transfer to memory, data is written by word access. |
| ANDC | B | CCR ∧ #IMM → CCR<br>Logically ANDs the CCR with immediate data. |
| ORC | B | CCR ∨ #IMM → CCR<br>Logically ORs the CCR with immediate data. |
| XORC | B | CCR ⊕ #IMM → CCR<br>Logically XORs the CCR with immediate data. |
| NOP | — | PC + 2 → PC<br>Only increments the program counter. |

Note: * Refers to the operand size.
B: Byte
W: Word

**Table 2.9 Block Data Transfer Instructions**

| Instruction | Size | Function |
|---|---|---|
| EEPMOV.B | — | if R4L ≠ 0 then<br>Repeat @ER5+ → @ER6+,<br>R4L–1 → R4L<br>Until R4L = 0<br>else next; |
| EEPMOV.W | — | if R4 ≠ 0 then<br>Repeat @ER5+ → @ER6+,<br>R4–1 → R4<br>Until R4 = 0<br>else next;<br>Transfers a data block. Starting from the address set in ER5, transfers data for the number of bytes set in R4L or R4 to the address location set in ER6.<br>Execution of the next instruction begins as soon as the transfer is completed. |

### 2.4.2 Basic Instruction Formats

H8/300H CPU instructions consist of 2-byte (1-word) units. An instruction consists of an operation field (op), a register field (r), an effective address extension (EA), and a condition field (cc).

Figure 2.7 shows examples of instruction formats.

#### (1) Operation Field

Indicates the function of the instruction, the addressing mode, and the operation to be carried out on the operand. The operation field always includes the first four bits of the instruction. Some instructions have two operation fields.

#### (2) Register Field

Specifies a general register. Address registers are specified by 3 bits, and data registers by 3 bits or 4 bits. Some instructions have two register fields. Some have no register field.

#### (3) Effective Address Extension

8, 16, or 32 bits specifying immediate data, an absolute address, or a displacement. A24-bit address or displacement is treated as a 32-bit data in which the first 8 bits are 0 (H'00).

#### (4) Condition Field

Specifies the branching condition of Bcc instructions.

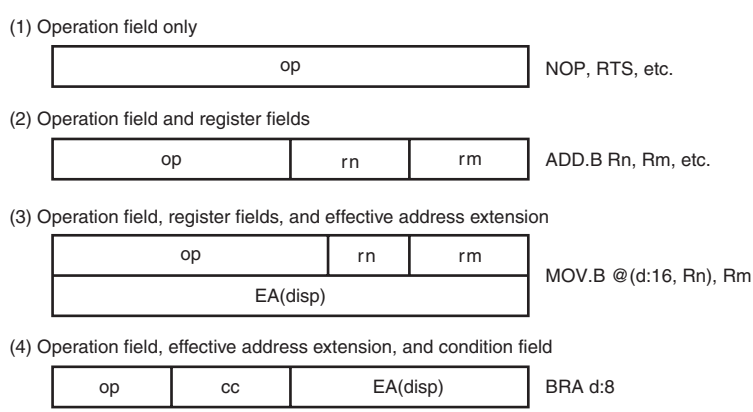



**Figure 2.7 Instruction Formats**

## 2.5 Addressing Modes and Effective Address Calculation

The following describes the H8/300H CPU. In this LSI, the upper eight bits are ignored in the generated 24-bit address, so the effective address is 16 bits.

### 2.5.1 Addressing Modes

The H8/300H CPU supports the eight addressing modes listed in table 2.10. Each instruction uses a subset of these addressing modes. Addressing modes that can be used differ depending on the instruction. For details, refer to appendix A.4, Combinations of Instructions and Addressing Modes.

Arithmetic and logic instructions can use the register direct and immediate modes. Data transfer instructions can use all addressing modes except program-counter relative and memory indirect. Bit-manipulation instructions use register direct, register indirect, or the absolute addressing mode (@aa:8) to specify an operand, and register direct (BSET, BCLR, BNOT, and BTST instructions) or immediate (3-bit) addressing mode to specify a bit number in the operand.

**Table 2.10 Addressing Modes**

| No. | Addressing Mode | Symbol |
|---|---|---|
| 1 | Register direct | Rn |
| 2 | Register indirect | @ERn |
| 3 | Register indirect with displacement | @(d:16,ERn)/@(d:24,ERn) |
| 4 | Register indirect with post-increment<br>Register indirect with pre-decrement | @ERn+<br>@–ERn |
| 5 | Absolute address | @aa:8/@aa:16/@aa:24 |
| 6 | Immediate | #xx:8/#xx:16/#xx:32 |
| 7 | Program-counter relative | @(d:8,PC)/@(d:16,PC) |
| 8 | Memory indirect | @@aa:8 |

#### (1) Register Direct—Rn

The register field of the instruction specifies an 8-, 16-, or 32-bit general register containing the operand. R0H to R7H and R0L to R7L can be specified as 8-bit registers. R0 to R7 and E0 to E7 can be specified as 16-bit registers. ER0 to ER7 can be specified as 32-bit registers.

### (2) Register Indirect—@ERn

The register field of the instruction code specifies an address register (ERn), the lower 24 bits of which contain the address of the operand on memory.

### (3) Register Indirect with Displacement—@(d:16, ERn) or @(d:24, ERn)

A 16-bit or 24-bit displacement contained in the instruction is added to an address register (ERn) specified by the register field of the instruction, and the lower 24 bits of the sum the address of a memory operand. A 16-bit displacement is sign-extended when added.

### (4) Register Indirect with Post-Increment or Pre-Decrement—@ERn+ or @-ERn

- Register indirect with post-increment—@ERn+

  The register field of the instruction code specifies an address register (ERn) the lower 24 bits of which contains the address of a memory operand. After the operand is accessed, 1, 2, or 4 is added to the address register contents (32 bits) and the sum is stored in the address register. The value added is 1 for byte access, 2 for word access, or 4 for longword access. For the word or longword access, the register value should be even.
- Register indirect with pre-decrement—@-ERn

  The value 1, 2, or 4 is subtracted from an address register (ERn) specified by the register field in the instruction code, and the lower 24 bits of the result is the address of a memory operand. The result is also stored in the address register. The value subtracted is 1 for byte access, 2 for word access, or 4 for longword access. For the word or longword access, the register value should be even.

### (5) Absolute Address—@aa:8, @aa:16, @aa:24

The instruction code contains the absolute address of a memory operand. The absolute address may be 8 bits long (@aa:8), 16 bits long (@aa:16), 24 bits long (@aa:24)

For an 8-bit absolute address, the upper 16 bits are all assumed to be 1 (H'FFFF). For a 16-bit absolute address the upper 8 bits are a sign extension. A 24-bit absolute address can access the entire address space.

The access ranges of absolute addresses for this LSI are those shown in table 2.11, because the upper 8 bits are ignored.

**Table 2.11 Absolute Address Access Ranges**

| Absolute Address | Access Range |
|---|---|
| 8 bits (@aa:8) | H'FF00 to H'FFFF |
| 16 bits (@aa:16) | H'0000 to H'FFFF |
| 24 bits (@aa:24) | H'0000 to H'FFFF |

### (6) Immediate—#xx:8, #xx:16, or #xx:32

The instruction contains 8-bit (#xx:8), 16-bit (#xx:16), or 32-bit (#xx:32) immediate data as an operand.

The ADDS, SUBS, INC, and DEC instructions contain immediate data implicitly. Some bit manipulation instructions contain 3-bit immediate data in the instruction code, specifying a bit number.

### (7) Program-Counter Relative—@(d:8, PC) or @(d:16, PC)

This mode is used in the BSR instruction. An 8-bit or 16-bit displacement contained in the instruction is sign-extended and added to the 24-bit PC contents to generate a branch address. The PC value to which the displacement is added is the address of the first byte of the next instruction, so the possible branching range is –126 to +128 bytes (–63 to +64 words) or –32766 to +32768 bytes (–16383 to +16384 words) from the branch instruction. The resulting value should be an even number.

### (8) Memory Indirect—@@aa:8

This mode can be used by the JMP and JSR instructions. The instruction code contains an 8-bit absolute address specifying a memory operand. This memory operand contains a branch address. The memory operand is accessed in words, generating a 16-bit branch address. Figure 2.8 shows how to specify branch address for in memory indirect mode. The upper bits of the absolute address are all assumed to be 0, so the address range is 0 to 255 (H'0000 to H'00FF).

Note that the first part of the address range is also the exception vector area.

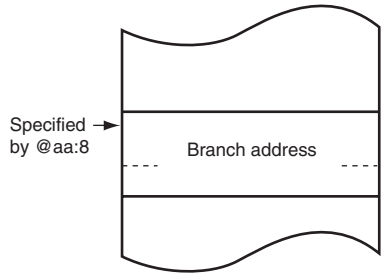



**Figure 2.8 Branch Address Specification in Memory Indirect Mode**

### 2.5.2 Effective Address Calculation

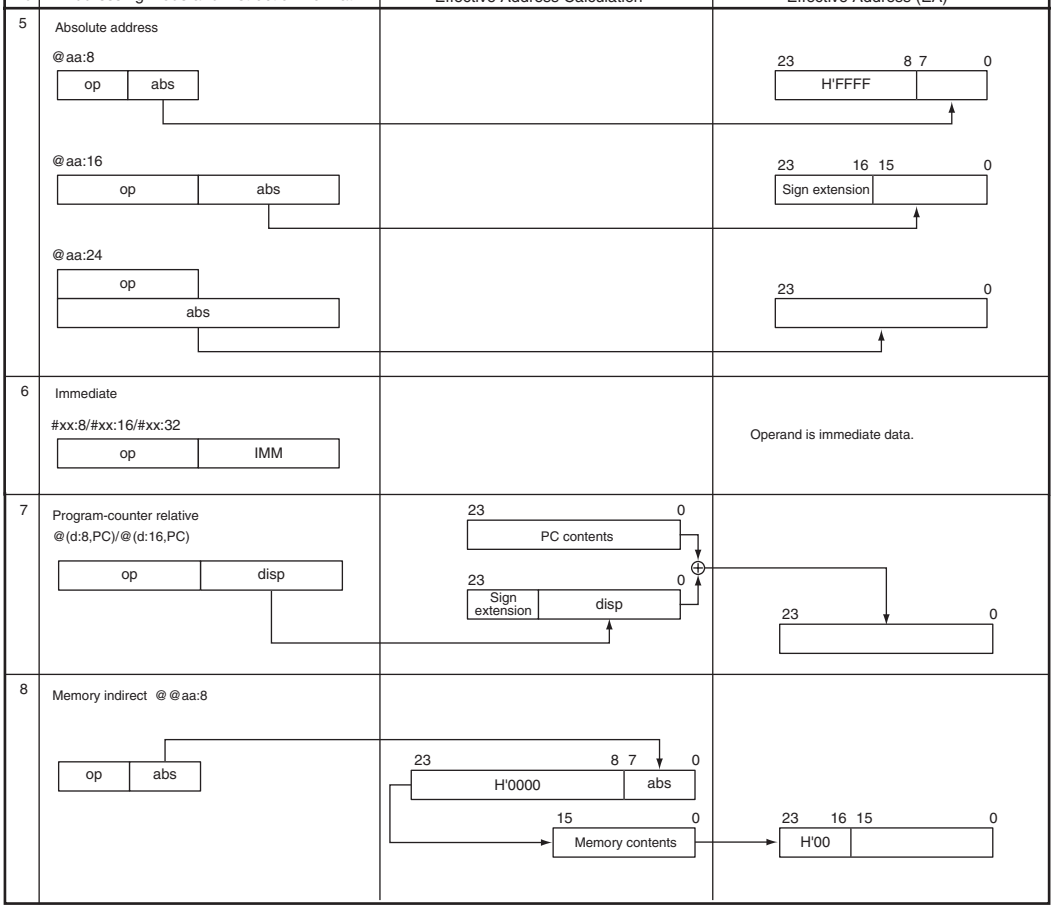
Table 2.12 indicates how effective addresses are calculated in each addressing mode. In this LSI the upper 8 bits of the effective address are ignored in order to generate a 16-bit effective address.

**Table 2.12 Effective Address Calculation (1)**

| No | Addressing Mode and Instruction Format | Effective Address Calculation | Effective Address (EA) |
|---|---|---|---|
| 1 | Register direct(Rn)<br>op \| rm \| rn | | Operand is general register contents. |
| 2 | Register indirect(@ERn)<br>op \| r | 31 … 0: General register contents | 23 … 0 |
| 3 | Register indirect with displacement<br>@(d:16,ERn) or @(d:24,ERn)<br>op \| r \| disp | 31 … 0: General register contents<br>31 … 0: Sign extension \| disp | 23 … 0 |
| 4 | Register indirect with post-increment or pre-decrement<br>•Register indirect with post-increment @ERn+<br>op \| r | 31 … 0: General register contents<br>1, 2, or 4 | 23 … 0 |
| | •Register indirect with pre-decrement @-ERn<br>op \| r | 31 … 0: General register contents<br>1, 2, or 4<br>The value to be added or subtracted is 1 when the operand is byte size, 2 for word size, and 4 for longword size. | 23 … 0 |

## Table 2.12 Effective Address Calculation (2)

| No | Addressing Mode and Instruction Format | Effective Address Calculation | Effective Address (EA) |
|---|---|---|---|
| 5 | Absolute address<br>@aa:8<br>op \| abs<br>@aa:16<br>op \| abs<br>@aa:24<br>op<br>abs | | 23 … 8 7 … 0: H'FFFF \|<br>23 … 16 15 … 0: Sign extension \|<br>23 … 0 |
| 6 | Immediate<br>#xx:8/#xx:16/#xx:32<br>op \| IMM | | Operand is immediate data. |
| 7 | Program-counter relative<br>@(d:8,PC)/@(d:16,PC)<br>op \| disp | 23 … 0: PC contents<br>⊕<br>23 … 0: Sign extension \| disp | 23 … 0 |
| 8 | Memory indirect @@aa:8<br>op \| abs | 23 … 8 7 … 0: H'0000 \| abs<br>15 … 0: Memory contents | 23 … 16 15 … 0: H'00 \| |



[Legend]
r, rm,rn : Register field
op : Operation field
disp : Displacement
IMM : Immediate data
abs : Absolute address

## 2.6 Basic Bus Cycle

CPU operation is synchronized by a system clock ($\phi$) or a subclock ($\phi_{SUB}$). The period from a rising edge of $\phi$ or $\phi_{SUB}$ to the next rising edge is called one state. A bus cycle consists of two states or three states. The cycle differs depending on whether access is to on-chip memory or to on-chip peripheral modules.

### 2.6.1 Access to On-Chip Memory (RAM, ROM)

Access to on-chip memory takes place in two states. The data bus width is 16 bits, allowing access in byte or word size. Figure 2.9 shows the on-chip memory access cycle.

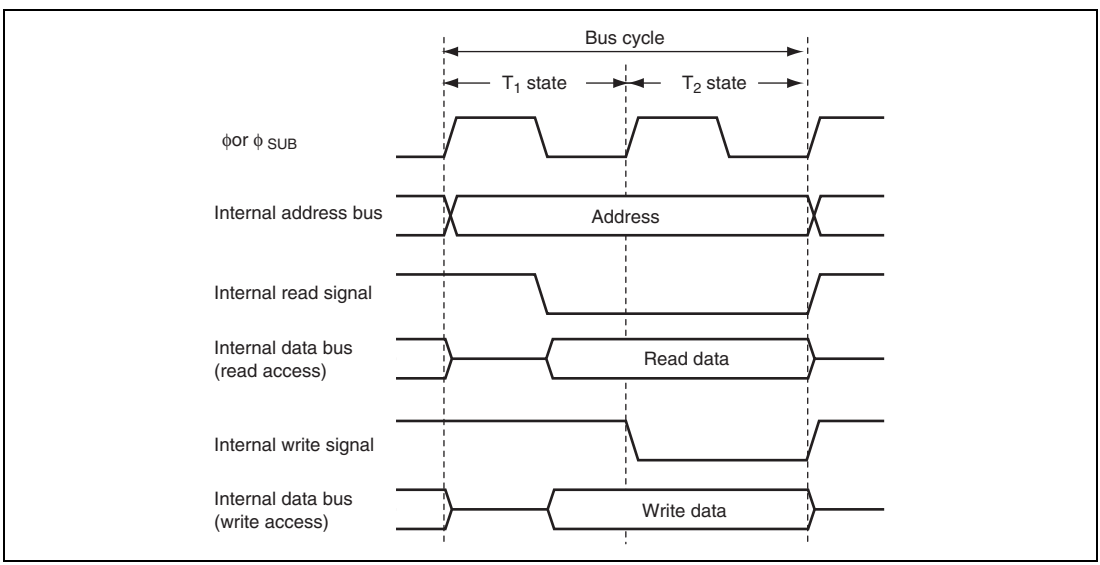



**Figure 2.9 On-Chip Memory Access Cycle**

### 2.6.2 On-Chip Peripheral Modules

On-chip peripheral modules are accessed in two states or three states. The data bus width is 8 bits or 16 bits depending on the register. For description on the data bus width and number of accessing states of each register, refer to section 15.1, Register Addresses (Address Order). Registers with 16-bit data bus width can be accessed by word size only. Registers with 8-bit data bus width can be accessed by byte or word size. When a register with 8-bit data bus width is accessed by word size, a bus cycle occurs twice. In two-state access, the operation timing is the same as that for on-chip memory.

Figure 2.10 shows the operation timing in the case of three-state access to an on-chip peripheral module.

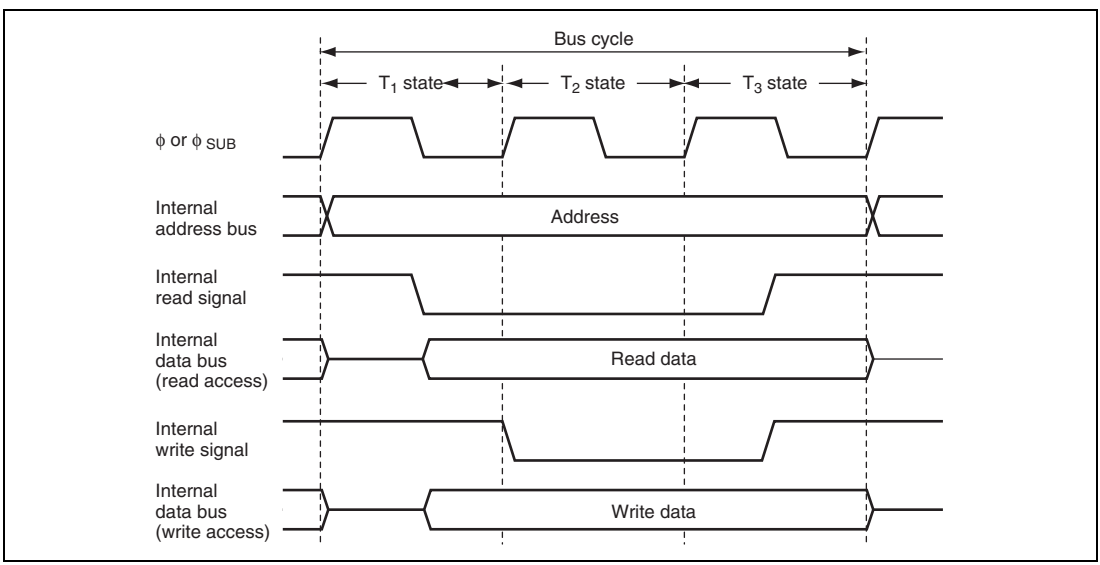



**Figure 2.10 On-Chip Peripheral Module Access Cycle (3-State Access)**

## 2.7 CPU States

There are four CPU states: the reset state, program execution state, program halt state, and exception-handling state. The program execution state includes active (high-speed or medium-speed) mode and subactive mode. For the program halt state, there are sleep (high-speed or medium-speed) mode, standby mode, watch mode, and subsleep mode. These states are shown in figure 2.11. Figure 2.12 shows the state transitions. For details on program execution state and program halt state, refer to section 5, Power-Down Modes. For details on exception handling, refer to section 3, Exception Handling.

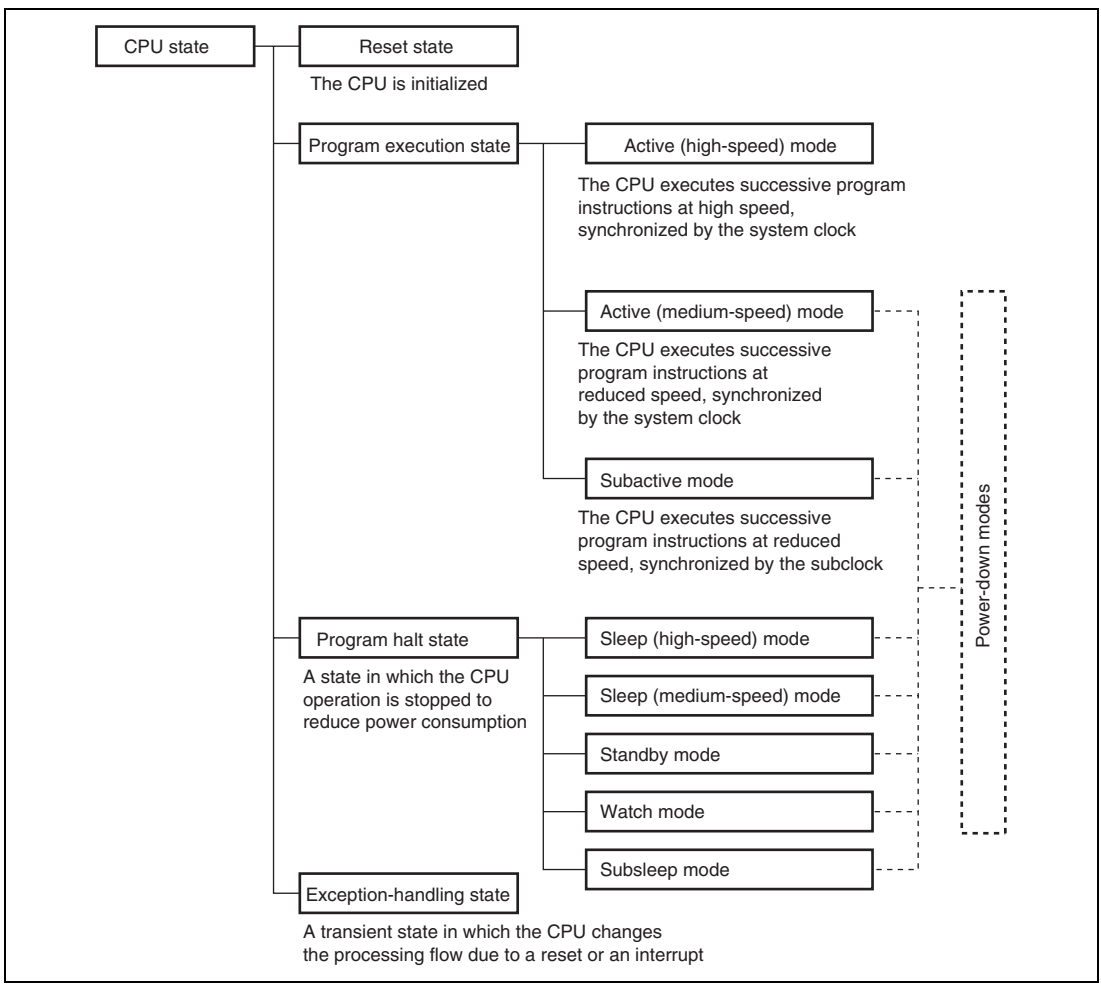



**Figure 2.11 CPU Operating States**

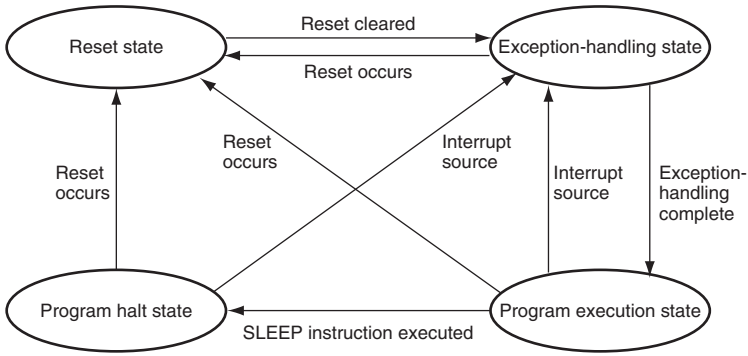



**Figure 2.12 State Transitions**

## 2.8 Usage Notes

### 2.8.1 Notes on Data Access to Empty Areas

The address space of this LSI includes empty areas in addition to the ROM, RAM, and on-chip I/O registers areas available to the user. When data is transferred from CPU to empty areas, the transferred data will be lost. This action may also cause the CPU to malfunction. When data is transferred from an empty area to CPU, the contents of the data cannot be guaranteed.

### 2.8.2 EEPMOV Instruction

EEPMOV is a block-transfer instruction and transfers the byte size of data indicated by R4L, which starts from the address indicated by R5, to the address indicated by R6. Set R4L and R6 so that the end address of the destination address (value of R6 + R4L) does not exceed H'FFFF (the value of R6 must not change from H'FFFF to H'0000 during execution).

### 2.8.3 Bit-Manipulation Instruction

The BSET, BCLR, BNOT, BST, and BIST instructions read data from the specified address in byte units, manipulate the data of the target bit, and write data to the same address again in byte units. Special care is required when using these instructions in cases where two registers are assigned to the same address, or when a bit is directly manipulated for a port or a register containing a write-only bit, because this may rewrite data of a bit other than the bit to be manipulated.

#### (1) Bit manipulation for two registers assigned to the same address

**Example 1: Bit manipulation for the timer load register and timer counter**

Figure 2.13 shows an example of a timer in which two timer registers are assigned to the same address. When a bit-manipulation instruction accesses the timer load register and timer counter of a reloadable timer, since these two registers share the same address, the following operations takes place.

1. Data is read in byte units.
2. The CPU sets or resets the bit to be manipulated with the bit-manipulation instruction.
3. The written data is written again in byte units to the timer load register.

The timer is counting, so the value read is not necessarily the same as the value in the timer load register. As a result, bits other than the intended bit in the timer counter may be modified and the modified value may be written to the timer load register.

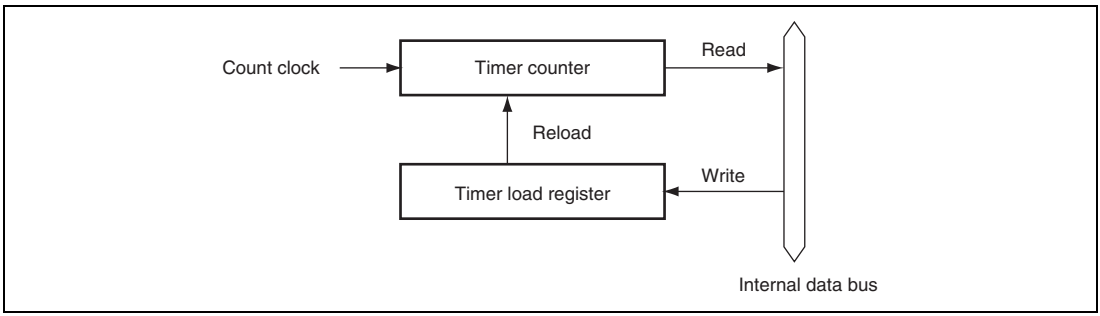



**Figure 2.13 Example of Timer Configuration with Two Registers Allocated to Same Address**

**Example 2: When the BSET instruction is executed for port 5**

P57 and P56 are input pins, with a low-level signal input at P57 and a high-level signal input at P56. P55 to P50 are output pins and output low-level signals. An example to output a high-level signal at P50 with a BSET instruction is shown below.

- Prior to executing BSET instruction

| | **P57** | **P56** | **P55** | **P54** | **P53** | **P52** | **P51** | **P50** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | Low level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- BSET instruction executed

```
BSET   #0,    @PDR5
```

The BSET instruction is executed for port 5.

- After executing BSET instruction

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | High level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

- Description on operation

1. When the BSET instruction is executed, first the CPU reads port 5.
   Since P57 and P56 are input pins, the CPU reads the pin states (low-level and high-level input).
   P55 to P50 are output pins, so the CPU reads the value in PDR5. In this example PDR5 has a value of H'80, but the value read by the CPU is H'40.
2. Next, the CPU sets bit 0 of the read data to 1, changing the PDR5 data to H'41.
3. Finally, the CPU writes H'41 to PDR5, completing execution of BSET instruction.
   As a result of the BSET instruction, bit 0 in PDR5 becomes 1, and P50 outputs a high-level signal. However, bits 7 and 6 of PDR5 end up with different values. To prevent this problem, store a copy of the PDR5 data in a work area in memory. Perform the bit manipulation on the data in the work area, then write this data to PDR5.

- Prior to executing BSET instruction

```
MOV.B   #H'80, R0L
MOV.B   R0L,   @RAM0
MOV.B   R0L,   @PDR5
```

The PDR5 value (H'80) is written to a work area in memory (RAM0) as well as to PDR5.

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | Low level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAM0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- BSET instruction executed

```
BSET    #0,    @RAM0
```

The BSET instruction is executed designating the PDR5 work area (RAM0).

- After executing BSET instruction

```
MOV.B   @RAM0, R0L
MOV.B   R0L,  @PDR5
```

The work area (RAM0) value is written to PDR5.

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | High level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| RAM0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### (2) Bit Manipulation in a Register Containing a Write-Only Bit

**Example 3: BCLR instruction executed designating port 5 control register PCR5**

P57 and P56 are input pins, with a low-level signal input at P57 and a high-level signal input at P56. P55 to P50 are output pins that output low-level signals. An example of setting the P50 pin as an input pin by the BCLR instruction is shown below. It is assumed that a high-level signal will be input to this input pin.

- Prior to executing BCLR instruction

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | Low level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- BCLR instruction executed

```
BCLR   #0,    @PCR5
```

The BCLR instruction is executed for PCR5.

- After executing BCLR instruction

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Output | Output | Output | Output | Output | Output | Output | Input |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | High level |
| PCR5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Description on operation

1. When the BCLR instruction is executed, first the CPU reads PCR5. Since PCR5 is a write-only register, the CPU reads a value of H'FF, even though the PCR5 value is actually H'3F.
2. Next, the CPU clears bit 0 in the read data to 0, changing the data to H'FE.
3. Finally, H'FE is written to PCR5 and BCLR instruction execution ends.

   As a result of this operation, bit 0 in PCR5 becomes 0, making P50 an input port. However, bits 7 and 6 in PCR5 change to 1, so that P57 and P56 change from input pins to output pins. To prevent this problem, store a copy of the PDR5 data in a work area in memory and manipulate data of the bit in the work area, then write this data to PDR5.

- Prior to executing BCLR instruction

```
MOV.B   #H'3F, R0L
MOV.B   R0L,   @RAM0
MOV.B   R0L,   @PCR5
```

The PCR5 value (H'3F) is written to a work area in memory (RAM0) as well as to PCR5.

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | Low level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAM0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

- BCLR instruction executed

```
BCLR   #0,    @RAM0
```

The BCLR instructions executed for the PCR5 work area (RAM0).

- After executing BCLR instruction

```
MOV.B   @RAM0, R0L
MOV.B   R0L,  @PCR5
```

The work area (RAM0) value is written to PCR5.

| | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low level | High level | Low level | Low level | Low level | Low level | Low level | High level |
| PCR5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| PDR5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAM0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Table 2.13 lists the pairs of registers that share identical addresses. Table 2.14 lists the registers that contain write-only bits.

**Table 2.13 Registers with Shared Addresses**

| Register Name | Abbr. | Address |
|---|---|---|
| Timer counter and timer load register C | TCC/TLC | H'FFB5 |
| Port data register 1* | PDR1 | H'FFD4 |
| Port data register 3* | PDR3 | H'FFD6 |
| Port data register 4* | PDR4 | H'FFD7 |
| Port data register 5* | PDR5 | H'FFD8 |
| Port data register 6* | PDR6 | H'FFD9 |
| Port data register 7* | PDR7 | H'FFDA |
| Port data register 8* | PDR8 | H'FFDB |
| Port data register A* | PDRA | H'FFDD |

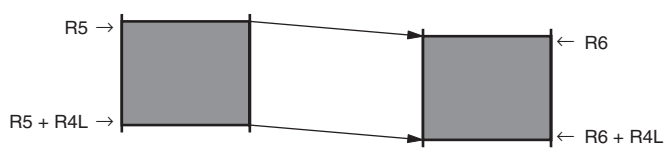
Note: * Port data registers have the same addresses as input pins.

**Table 2.14 Registers with Write-Only Bits**

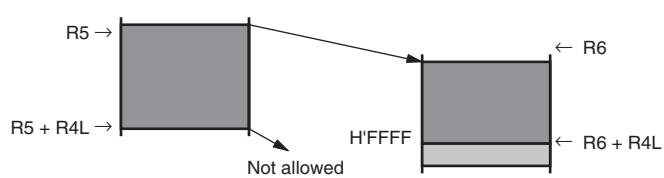
| Register Name | Abbr. | Address |
|---|---|---|
| Port control register 1 | PCR1 | H'FFE4 |
| Port control register 3 | PCR3 | H'FFE6 |
| Port control register 4 | PCR4 | H'FFE7 |
| Port control register 5 | PCR5 | H'FFE8 |
| Port control register 6 | PCR6 | H'FFE9 |
| Port control register 7 | PCR7 | H'FFEA |
| Port control register 8 | PCR8 | H'FFEB |
| Port control register A | PCRA | H'FFED |
| Timer control register F | TCRF | H'FFB6 |
| PWM control register | PWCR | H'FFD0 |
| PWM data register U | PWDRU | H'FFD1 |
| PWM data register L | PWDRL | H'FFD2 |

### 2.8.4 Notes on Use of the EEPMOV Instruction

- The EEPMOV instruction is a block data transfer instruction. It moves the number of bytes specified by R4L from the address specified by R5 to the address specified by R6.




- When setting R4L and R6, make sure that the final destination address (R6 + R4L) does not exceed H'FFFF. The value in R6 must not change from H'FFFF to H'0000 during execution of the instruction.



# Section 3 Exception Handling

## 3.1 Overview

Exception handling is performed in this LSI when a reset or interrupt occurs. Table 3.1 shows the priorities of these two types of exception handling.

**Table 3.1 Exception Handling Types and Priorities**

| Priority | Exception Source | Time of Start of Exception Handling |
|---|---|---|
| High | Reset | Exception handling starts as soon as the reset state is cleared |
| ↑ | Interrupt | When an interrupt is requested, exception handling starts after execution of the present instruction or the exception handling in progress is completed |
| Low | | |

## 3.2 Reset

### 3.2.1 Overview

A reset is the highest-priority exception. The internal state of the CPU and the registers of the on-chip peripheral modules are initialized.

### 3.2.2 Reset Sequence

As soon as the $\overline{\text{RES}}$ pin goes low, all processing is stopped and the chip enters the reset state.

To make sure the chip is reset properly, observe the following precautions.

- At power on: Hold the $\overline{\text{RES}}$ pin low until the clock pulse generator output stabilizes.
- Resetting during operation: Hold the $\overline{\text{RES}}$ pin low for at least 10 system clock cycles.

Reset exception handling takes place as follows.

- The CPU internal state and the registers of on-chip peripheral modules are initialized, with the I bit of the condition code register (CCR) set to 1.
- The PC is loaded from the reset exception handling vector address (H'0000 to H'0001), after which the program starts executing from the address indicated in PC.

When system power is turned on or off, the $\overline{\text{RES}}$ pin should be held low.

Figure 3.1 shows the reset sequence starting from $\overline{\text{RES}}$ input.

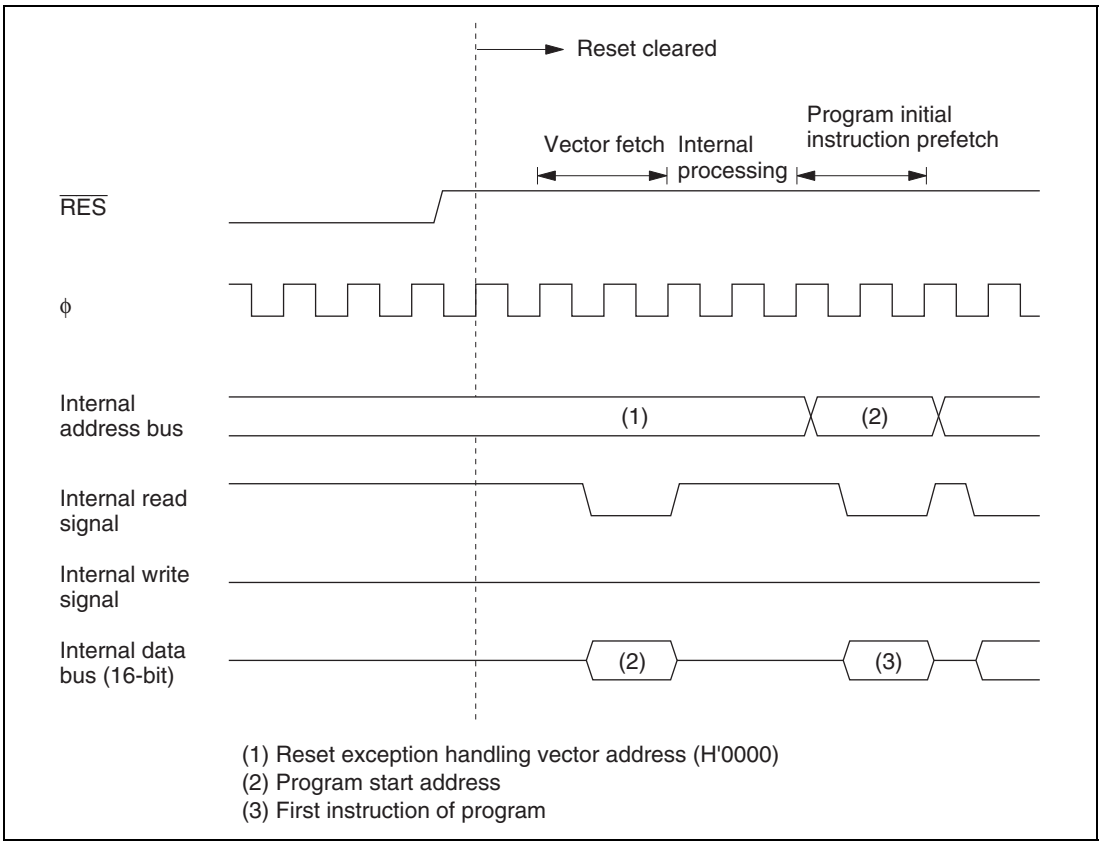



**Figure 3.1 Reset Sequence**

### 3.2.3 Interrupt Immediately after Reset

After a reset, if an interrupt were to be accepted before the stack pointer (SP: R7) was initialized, PC and CCR would not be pushed onto the stack correctly, resulting in program runaway. To prevent this, immediately after reset exception handling all interrupts are masked. For this reason, the initial program instruction is always executed immediately after a reset. This instruction should initialize the stack pointer (e.g. MOV.W #xx: 16, SP).

## 3.3 Interrupts

### 3.3.1 Overview

The interrupt sources include 13 external interrupts ($IRQ_4$ to $IRQ_0$, $WKP_7$ to $WKP_0$) and 23 internal interrupts from on-chip peripheral modules. Table 3.2 shows the interrupt sources, their priorities, and their vector addresses. When more than one interrupt is requested, the interrupt with the highest priority is processed.

The interrupts have the following features:

- Internal and external interrupts can be masked by the I bit in CCR. When the I bit is set to 1, interrupt request flags can be set but the interrupts are not accepted.
- $IRQ_4$ to $IRQ_0$ and $WKP_7$ to $WKP_0$ can be set to either rising edge sensing or falling edge sensing.

## Table 3.2 Interrupt Sources and Their Priorities

| Interrupt Source | Interrupt | Vector Number | Vector Address | Priority |
|---|---|---|---|---|
| $\overline{RES}$<br>Watchdog timer | Reset | 0 | H'0000 to H'0001 | High |
| $\overline{IRQ_0}$ | $IRQ_0$ | 4 | H'0008 to H'0009 | ↑ |
| $\overline{IRQ_1}$ | $IRQ_1$ | 5 | H'000A to H'000B | |
| $\overline{IRQ_2}$ | $IRQ_2$ | 6 | H'000C to H'000D | |
| $\overline{IRQ_3}$ | $IRQ_3$ | 7 | H'000E to H'000F | |
| $\overline{IRQ_4}$ | $IRQ_4$ | 8 | H'0010 to H'0011 | |
| $\overline{WKP_0}$<br>$\overline{WKP_1}$<br>$\overline{WKP_2}$<br>$\overline{WKP_3}$<br>$\overline{WKP_4}$<br>$\overline{WKP_5}$<br>$\overline{WKP_6}$<br>$\overline{WKP_7}$ | $WKP_0$<br>$WKP_1$<br>$WKP_2$<br>$WKP_3$<br>$WKP_4$<br>$WKP_5$<br>$WKP_6$<br>$WKP_7$ | 9 | H'0012 to H'0013 | |
| Timer A | Timer A overflow | 11 | H'0016 to H'0017 | |
| Asynchronous counter | Asynchronous counter overflow | 12 | H'0018 to H'0019 | |
| Timer C | Timer C overflow or underflow | 13 | H'001A to H'001B | |
| Timer FL | Timer FL compare match<br>Timer FL overflow | 14 | H'001C to H'001D | |
| Timer FH | Timer FH compare match<br>Timer FH overflow | 15 | H'001E to H'001F | |
| Timer G | Timer G input capture<br>Timer G overflow | 16 | H'0020 to H'0021 | |
| SCI3-1 | SCI3-1 transmit end<br>SCI3-1 transmit data empty<br>SCI3-1 receive data full<br>SCI3-1 overrrun error<br>SCI3-1 framing error<br>SCI3-1 parity error | 17 | H'0022 to H'0023 | |
| SCI3-2 | SCI3-2 transmit end<br>SCI3-2 transmit data empty<br>SCI3-2 receive data full<br>SCI3-2 overrun error<br>SCI3-2 framing error<br>SCI3-2 parity error | 18 | H'0024 to H'0025 | |
| A/D | A/D conversion end | 19 | H'0026 to H'0027 | ↓ |
| (SLEEP instruction executed) | Direct transfer | 20 | H'0028 to H'0029 | Low |

Note: Vector addresses H'0002 to H'0007 and H'0014 to H'0015 are reserved and cannot be used.

### 3.3.2 Interrupt Control Registers

Table 3.3 lists the registers that control interrupts.

**Table 3.3 Interrupt Control Registers**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| IRQ edge select register | IEGR | R/W | H'E0 | H'FFF2 |
| Interrupt enable register 1 | IENR1 | R/W | H'00 | H'FFF3 |
| Interrupt enable register 2 | IENR2 | R/W | H'00 | H'FFF4 |
| Interrupt request register 1 | IRR1 | R/W* | H'20 | H'FFF6 |
| Interrupt request register 2 | IRR2 | R/W* | H'00 | H'FFF7 |
| Wakeup interrupt request register | IWPR | R/W* | H'00 | H'FFF9 |
| Wakeup edge select register | WEGR | R/W | H'00 | H'FF90 |

Note: * Write is enabled only for writing of 0 to clear a flag.

#### (1) IRQ Edge Select Register (IEGR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | IEG4 | IEG3 | IEG2 | IEG1 | IEG0 |
| Initial value | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | — | R/W | R/W | R/W | R/W | R/W |

IEGR is an 8-bit read/write register used to designate whether pins $\overline{IRQ_4}$ to $\overline{IRQ_0}$ are set to rising edge sensing or falling edge sensing.

**Bits 7 to 5:** Reserved bits

Bits 7 to 5 are reserved: they are always read as 1 and cannot be modified.

**Bit 4:** $IRQ_4$ edge select (IEG4)

Bit 4 selects the input sensing of the $\overline{IRQ_4}$ pin and $\overline{ADTRG}$ pin.

| Bit 4 IEG4 | Description | |
|---|---|---|
| 0 | Falling edge of $\overline{IRQ_4}$ and $\overline{ADTRG}$ pin input is detected | (initial value) |
| 1 | Rising edge of $\overline{IRQ_4}$ and $\overline{ADTRG}$ pin input is detected | |

**Bit 3:** $IRQ_3$ edge select (IEG3)

Bit 3 selects the input sensing of the $\overline{IRQ_3}$ pin and TMIF pin.

| Bit 3<br>IEG3 | Description | |
|---|---|---|
| 0 | Falling edge of $\overline{IRQ_3}$ and TMIF pin input is detected | (initial value) |
| 1 | Rising edge of $\overline{IRQ_3}$ and TMIF pin input is detected | |

**Bit 2:** $IRQ_2$ edge select (IEG2)

Bit 2 selects the input sensing of pin $\overline{IRQ_2}$.

| Bit 2<br>IEG2 | Description | |
|---|---|---|
| 0 | Falling edge of $\overline{IRQ_2}$ pin input is detected | (initial value) |
| 1 | Rising edge of $\overline{IRQ_2}$ pin input is detected | |

**Bit 1:** $IRQ_1$ edge select (IEG1)

Bit 1 selects the input sensing of the $\overline{IRQ_1}$ pin and TMIC pin.

| Bit 1<br>IEG1 | Description | |
|---|---|---|
| 0 | Falling edge of $\overline{IRQ_1}$ and TMIC pin input is detected | (initial value) |
| 1 | Rising edge of $\overline{IRQ_1}$ and TMIC pin input is detected | |

**Bit 0:** $IRQ_0$ edge select (IEG0)

Bit 0 selects the input sensing of pin $\overline{IRQ_0}$.

| Bit 0<br>IEG0 | Description | |
|---|---|---|
| 0 | Falling edge of $\overline{IRQ_0}$ pin input is detected | (initial value) |
| 1 | Rising edge of $\overline{IRQ_0}$ pin input is detected | |

### (2) Interrupt Enable Register 1 (IENR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IENTA | — | IENWP | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

IENR1 is an 8-bit read/write register that enables or disables interrupt requests.

**Bit 7:** Timer A interrupt enable (IENTA)

Bit 7 enables or disables timer A overflow interrupt requests.

| Bit 7<br>IENTA | Description | |
|---|---|---|
| 0 | Disables timer A interrupt requests | (initial value) |
| 1 | Enables timer A interrupt requests | |

**Bit 6:** Reserved bit

Bit 6 is a readable/writable reserved bit. It is initialized to 0 by a reset.

**Bit 5:** Wakeup interrupt enable (IENWP)

Bit 5 enables or disables $WKP_7$ to $WKP_0$ interrupt requests.

| Bit 5<br>IENWP | Description | |
|---|---|---|
| 0 | Disables $\overline{WKP}_7$ to $\overline{WKP}_0$ interrupt requests | (initial value) |
| 1 | Enables $\overline{WKP}_7$ to $\overline{WKP}_0$ interrupt requests | |

**Bits 4 to 0:** $IRQ_4$ to $IRQ_0$ interrupt enable (IEN4 to IEN0)

Bits 4 to 0 enable or disable $IRQ_4$ to $IRQ_0$ interrupt requests.

| Bit n<br>IENn | Description | |
|---|---|---|
| 0 | Disables interrupt requests from pin $\overline{IRQ}$n | (initial value) |
| 1 | Enables interrupt requests from pin $\overline{IRQ}$n | |

(n = 4 to 0)

### (3) Interrupt Enable Register 2 (IENR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IENDT | IENAD | — | IENTG | IENTFH | IENTFL | IENTC | IENEC |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

IENR2 is an 8-bit read/write register that enables or disables interrupt requests.

**Bit 7:** Direct transfer interrupt enable (IENDT)

Bit 7 enables or disables direct transfer interrupt requests.

| Bit 7<br>IENDT | Description | |
|---|---|---|
| 0 | Disables direct transfer interrupt requests | (initial value) |
| 1 | Enables direct transfer interrupt requests | |

**Bit 6:** A/D converter interrupt enable (IENAD)

Bit 6 enables or disables A/D converter interrupt requests.

| Bit 6<br>IENAD | Description | |
|---|---|---|
| 0 | Disables A/D converter interrupt requests | (initial value) |
| 1 | Enables A/D converter interrupt requests | |

**Bit 5:** Reserved bit

Bit 5 is a readable/writable reserved bit. It is initialized to 0 by a reset.

**Bit 4:** Timer G interrupt enable (IENTG)

Bit 4 enables or disables timer G input capture or overflow interrupt requests.

| Bit 4<br>IENTG | Description | |
|---|---|---|
| 0 | Disables timer G interrupt requests | (initial value) |
| 1 | Enables timer G interrupt requests | |

**Bit 3:** Timer FH interrupt enable (IENTFH)

Bit 3 enables or disables timer FH compare match and overflow interrupt requests.

| Bit 3<br>IENTFH | Description | |
|---|---|---|
| 0 | Disables timer FH interrupt requests | (initial value) |
| 1 | Enables timer FH interrupt requests | |

**Bit 2:** Timer FL interrupt enable (IENTFL)

Bit 2 enables or disables timer FL compare match and overflow interrupt requests.

| Bit 2<br>IENTFL | Description | |
|---|---|---|
| 0 | Disables timer FL interrupt requests | (initial value) |
| 1 | Enables timer FL interrupt requests | |

**Bit 1:** Timer C interrupt enable (IENTC)

Bit 1 enables or disables timer C overflow and underflow interrupt requests.

| Bit 1<br>IENTC | Description | |
|---|---|---|
| 0 | Disables timer C interrupt requests | (initial value) |
| 1 | Enables timer C interrupt requests | |

**Bit 0:** Asynchronous event counter interrupt enable (IENEC)

Bit 0 enables or disables asynchronous event counter interrupt requests.

| Bit 0<br>IENEC | Description | |
|---|---|---|
| 0 | Disables asynchronous event counter interrupt requests | (initial value) |
| 1 | Enables asynchronous event counter interrupt requests | |

For details of SCI3-1 and SCI3-2 interrupt control, see section 10.2.6, Serial Control Register 3 (SCR3).

### (4) Interrupt Request Register 1 (IRR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IRRTA | — | — | IRRI4 | IRRI3 | IRRI2 | IRRI1 | IRRI0 |
| Initial value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W* | R/W* | — | R/W* | R/W* | R/W* | R/W* | R/W* |

Note: * Only a write of 0 for flag clearing is possible

IRR1 is an 8-bit read/write register, in which a corresponding flag is set to 1 when a timer A or $IRQ_4$ to $IRQ_0$ interrupt is requested. The flags are not cleared automatically when an interrupt is accepted. It is necessary to write 0 to clear each flag.

**Bit 7:** Timer A interrupt request flag (IRRTA)

| Bit 7<br>IRRTA | Description | |
|---|---|---|
| 0 | Clearing condition:<br>When IRRTA = 1, it is cleared by writing 0 | (initial value) |
| 1 | Setting condition:<br>When the timer A counter value overflows from H'FF to H'00 | |

**Bit 6:** Reserved bit

Bit 6 is a readable/writable reserved bit. It is initialized to 0 by a reset.

**Bit 5:** Reserved bit

Bit 5 is reserved; it is always read as 1 and cannot be modified.

**Bits 4 to 0:** $IRQ_4$ to $IRQ_0$ interrupt request flags (IRRI4 to IRRI0)

| Bit n<br>IRRIn | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IRRIn = 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When pin $\overline{IRQ}$n is designated for interrupt input and the designated signal edge is input |

(n = 4 to 0)

### (5) Interrupt Request Register 2 (IRR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IRRDT | IRRAD | — | IRRTG | IRRTFH | IRRTFL | IRRTC | IRREC |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W* | R/W* | R/W | R/W* | R/W* | R/W* | R/W* | R/W* |

Note: * Only a write of 0 for flag clearing is possible

IRR2 is an 8-bit read/write register, in which a corresponding flag is set to 1 when a direct transfer, A/D converter, Timer G, Timer FH, Timer FC, or Timer C interrupt is requested. The flags are not cleared automatically when an interrupt is accepted. It is necessary to write 0 to clear each flag.

**Bit 7:** Direct transfer interrupt request flag (IRRDT)

| Bit 7<br>IRRDT | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IRRDT = 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When a direct transfer is made by executing a SLEEP instruction while DTON = 1 in SYSCR2 |

**Bit 6:** A/D converter interrupt request flag (IRRAD)

| Bit 6<br>IRRAD | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IRRAD = 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When A/D conversion is completed and ADSF is cleared to 0 in ADSR |

**Bit 5:** Reserved bit

Bit 5 is a readable/writable reserved bit. It is initialized to 0 by a reset.

**Bit 4:** Timer G interrupt request flag (IRRTG)

| Bit 4<br>IRRTG | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IRRTG = 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When the TMIG pin is designated for TMIG input and the designated signal edge is input, or when TCG overflows while OVIE is set to 1 in TMG |

**Bit 3:** Timer FH interrupt request flag (IRRTFH)

| Bit 3<br>IRRTFH | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IRRTFH = 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When TCFH and OCRFH match in 8-bit timer mode, or when TCF (TCFL, TCFH) and OCRF (OCRFL, OCRFH) match in 16-bit timer mode |

**Bit 2:** Timer FL interrupt request flag (IRRTFL)

| Bit 2<br>IRRTFL | Description | |
|---|---|---|
| 0 | Clearing condition:<br>When IRRTFL= 1, it is cleared by writing 0 | (initial value) |
| 1 | Setting condition:<br>When TCFL and OCRFL match in 8-bit timer mode | |

**Bit 1:** Timer C interrupt request flag (IRRTC)

| Bit 1<br>IRRTC | Description | |
|---|---|---|
| 0 | Clearing condition:<br>When IRRTC= 1, it is cleared by writing 0 | (initial value) |
| 1 | Setting condition:<br>When the timer C counter value overflows (from H'FF to H'00) or underflows (from H'00 to H'FF) | |

**Bit 0:** Asynchronous event counter interrupt request flag (IRREC)

| Bit 0<br>IRREC | Description | |
|---|---|---|
| 0 | Clearing condition:<br>When IRREC = 1, it is cleared by writing 0 | (initial value) |
| 1 | Setting condition:<br>When ECH overflows in 16-bit counter mode, or ECH or ECL overflows in 8-bit counter mode | |

### (6) Wakeup Interrupt Request Register (IWPR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IWPF7 | IWPF6 | IWPF5 | IWPF4 | IWPF3 | IWPF2 | IWPF1 | IWPF0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* |

Note: * Only a write of 0 for flag clearing is possible

IWPR is an 8-bit read/write register containing wakeup interrupt request flags. When one of pins $\overline{WKP}_7$ to $\overline{WKP}_0$ is designated for wakeup input and a rising or falling edge is input at that pin, the corresponding flag in IWPR is set to 1. A flag is not cleared automatically when the corresponding interrupt is accepted. Flags must be cleared by writing 0.

**Bits 7 to 0:** Wakeup interrupt request flags (IWPF7 to IWPF0)

| Bit n<br>IWPFn | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>When IWPFn= 1, it is cleared by writing 0 |
| 1 | Setting condition:<br>When pin $\overline{WKP}_n$ is designated for wakeup input and a rising or falling edge is input at that pin |

(n = 7 to 0)

### (7) Wakeup Edge Select Register (WEGR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WKEGS7 | WKEGS6 | WKEGS5 | WKEGS4 | WKEGS3 | WKEGS2 | WKEGS1 | WKEGS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

WEGR is an 8-bit read/write register that specifies rising or falling edge sensing for pins $\overline{WKP}$n.

WEGR is initialized to H'00 by a reset.

**Bit n:** $\overline{\text{WKP}}$n edge select (WKEGSn)

Bit n selects $\overline{\text{WKP}}$n pin input sensing.

| Bit n<br>WKEGSn | Description | |
|---|---|---|
| 0 | $\overline{\text{WKP}}$n pin falling edge detected | (initial value) |
| 1 | $\overline{\text{WKP}}$n pin rising edge detected | |

(n = 7 to 0)

### 3.3.3 External Interrupts

There are 13 external interrupts: $IRQ_4$ to $IRQ_0$ and $WKP_7$ to $WKP_0$.

#### (1) Interrupts $WKP_7$ to $WKP_0$

Interrupts $WKP_7$ to $WKP_0$ are requested by either rising or falling edge input to pins $\overline{WKP}_7$ to $\overline{WKP}_0$. When these pins are designated as pins $\overline{WKP}_7$ to $\overline{WKP}_0$ in port mode register 5 and a rising or falling edge is input, the corresponding bit in IWPR is set to 1, requesting an interrupt. Recognition of wakeup interrupt requests can be disabled by clearing the IENWP bit to 0 in IENR1. These interrupts can all be masked by setting the I bit to 1 in CCR.

When $WKP_7$ to $WKP_0$ interrupt exception handling is initiated, the I bit is set to 1 in CCR. Vector number 9 is assigned to interrupts $WKP_7$ to $WKP_0$. All eight interrupt sources have the same vector number, so the interrupt-handling routine must discriminate the interrupt source.

#### (2) Interrupts $IRQ_4$ to $IRQ_0$

Interrupts $IRQ_4$ to $IRQ_0$ are requested by input signals to pins $\overline{IRQ}_4$ to $\overline{IRQ}_0$. These interrupts are detected by either rising edge sensing or falling edge sensing, depending on the settings of bits IEG4 to IEG0 in IEGR.

When these pins are designated as pins $\overline{IRQ}_4$ to $\overline{IRQ}_0$ in port mode register 3 and 1 and the designated edge is input, the corresponding bit in IRR1 is set to 1, requesting an interrupt. Recognition of these interrupt requests can be disabled individually by clearing bits IEN4 to IEN0 to 0 in IENR1. These interrupts can all be masked by setting the I bit to 1 in CCR.

When $IRQ_4$ to $IRQ_0$ interrupt exception handling is initiated, the I bit is set to 1 in CCR. Vector numbers 8 to 4 are assigned to interrupts $IRQ_4$ to $IRQ_0$. The order of priority is from $IRQ_0$ (high) to $IRQ_4$ (low). Table 3.2 gives details.

### 3.3.4 Internal Interrupts

There are 23 internal interrupts that can be requested by the on-chip peripheral modules. When a peripheral module requests an interrupt, the corresponding bit in IRR1 or IRR2 is set to 1. Recognition of individual interrupt requests can be disabled by clearing the corresponding bit in IENR1 or IENR2. All these interrupts can be masked by setting the I bit to 1 in CCR. When internal interrupt handling is initiated, the I bit is set to 1 in CCR. Vector numbers from 20 to 11 are assigned to these interrupts. Table 3.2 shows the order of priority of interrupts from on-chip peripheral modules.

### 3.3.5 Interrupt Operations

Interrupts are controlled by an interrupt controller. Figure 3.2 shows a block diagram of the interrupt controller. Figure 3.3 shows the flow up to interrupt acceptance.

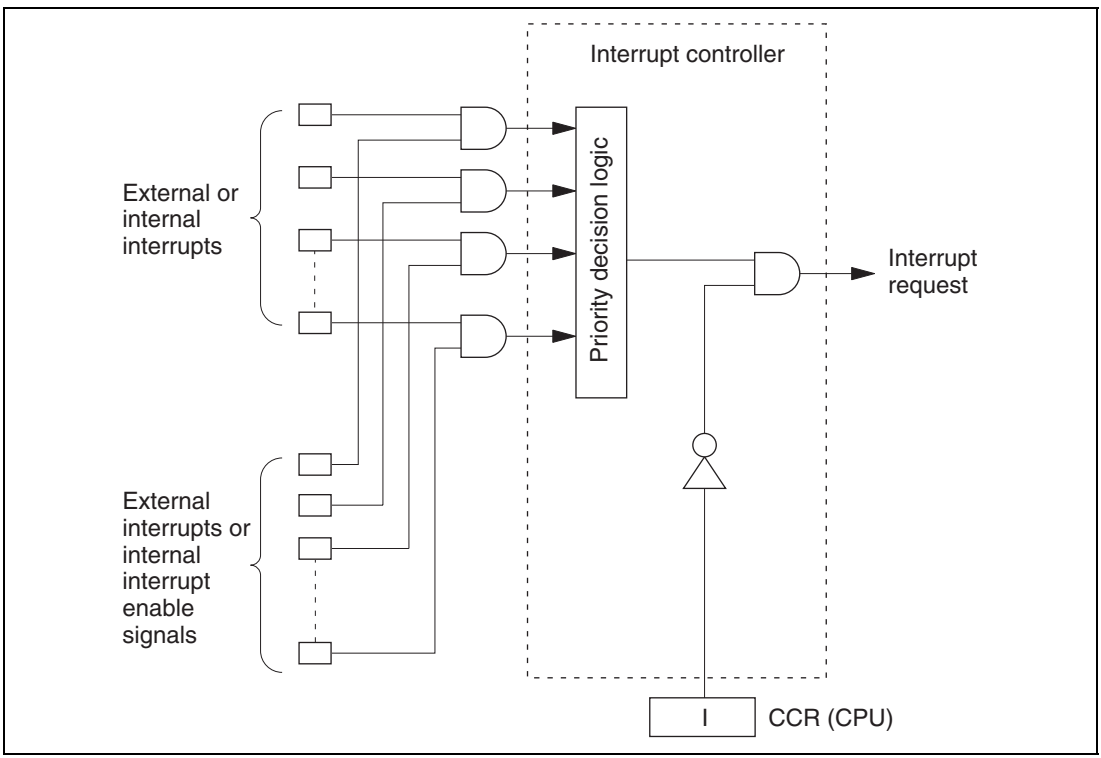


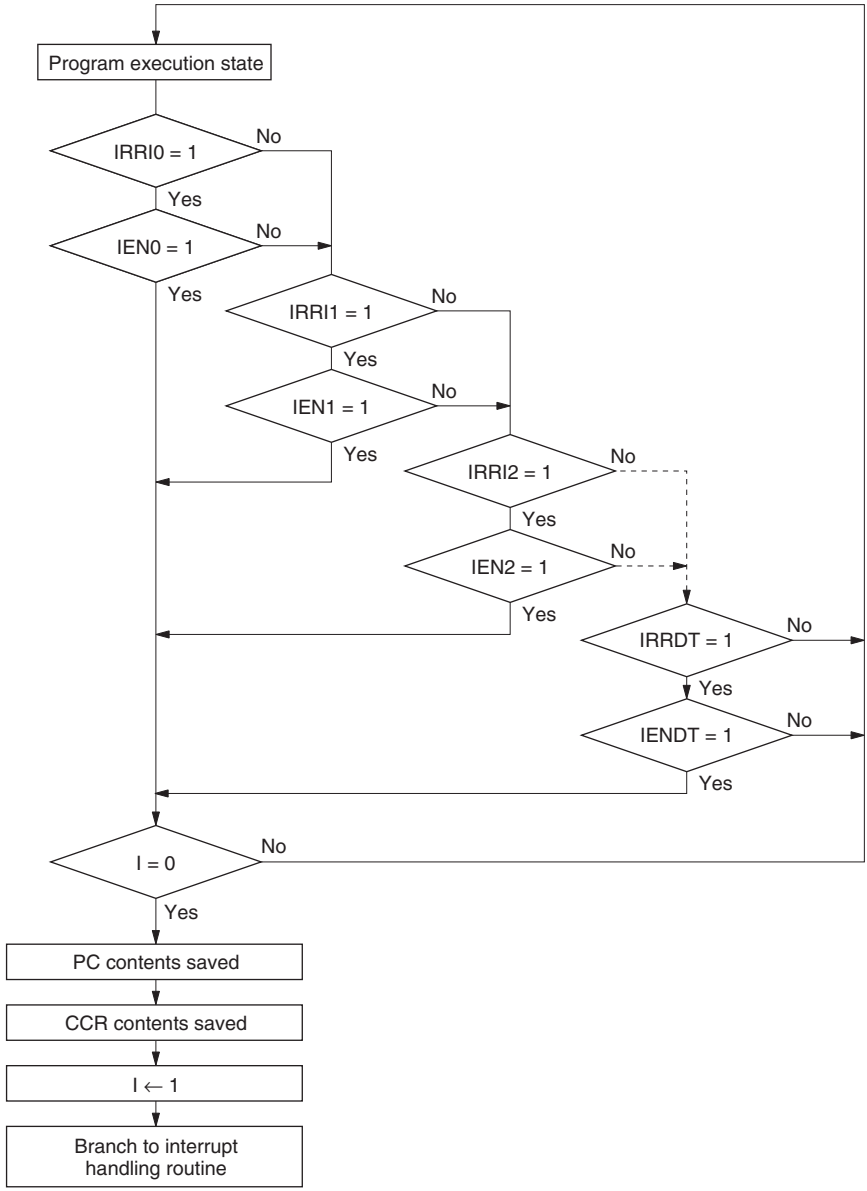

**Figure 3.2 Block Diagram of Interrupt Controller**

Interrupt operation is described as follows.

- When an interrupt condition is met while the interrupt enable register bit is set to 1, an interrupt request signal is sent to the interrupt controller.
- When the interrupt controller receives an interrupt request, it sets the interrupt request flag.
- From among the interrupts with interrupt request flags set to 1, the interrupt controller selects the interrupt request with the highest priority and holds the others pending. (Refer to table 3.2 for a list of interrupt priorities.)
- The interrupt controller checks the I bit of CCR. If the I bit is 0, the selected interrupt request is accepted; if the I bit is 1, the interrupt request is held pending.
- If the interrupt is accepted, after processing of the current instruction is completed, both PC and CCR are pushed onto the stack. The state of the stack at this time is shown in figure 3.4. The PC value pushed onto the stack is the address of the first instruction to be executed upon return from interrupt handling.
- The I bit of CCR is set to 1, masking further interrupts.
- The vector address corresponding to the accepted interrupt is generated, and the interrupt handling routine located at the address indicated by the contents of the vector address is executed.

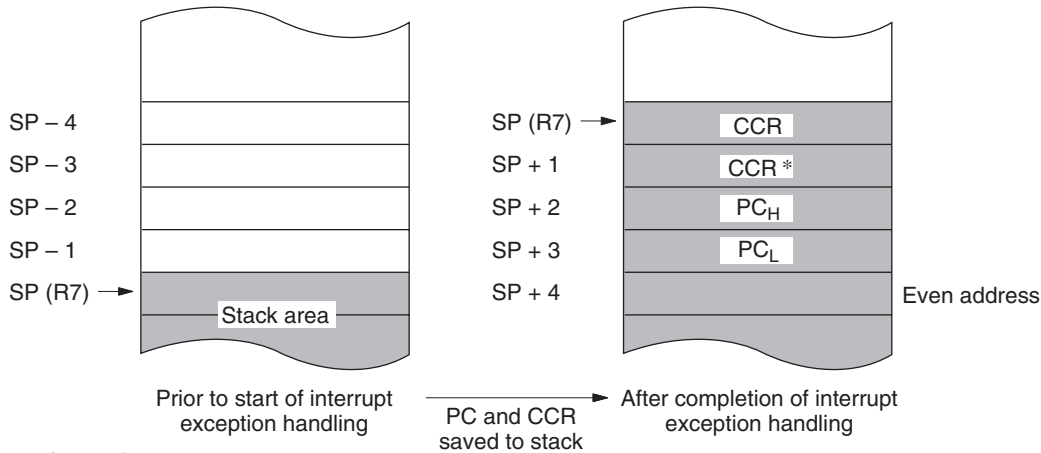
Notes: 1. When disabling interrupts by clearing bits in an interrupt enable register, or when clearing bits in an interrupt request register, always do so while interrupts are masked (I = 1).

2. If the above clear operations are performed while I = 0, and as a result a conflict arises between the clear instruction and an interrupt request, exception processing for the interrupt will be executed after the clear instruction has been executed.




Legend:
PC: Program counter
CCR: Condition code register
I: I bit of CCR

**Figure 3.3 Flow Up to Interrupt Acceptance**




Legend:
$PC_H$: Upper 8 bits of program counter (PC)
$PC_L$: Lower 8 bits of program counter (PC)
CCR: Condition code register
SP: Stack pointer

Notes: 1. PC shows the address of the first instruction to be executed upon return from the interrupt handling routine.
2. Register contents must always be saved and restored by word access, starting from an even-numbered address.
* Ignored on return.

**Figure 3.4 Stack State after Completion of Interrupt Exception Handling**

Figure 3.5 shows a typical interrupt sequence.

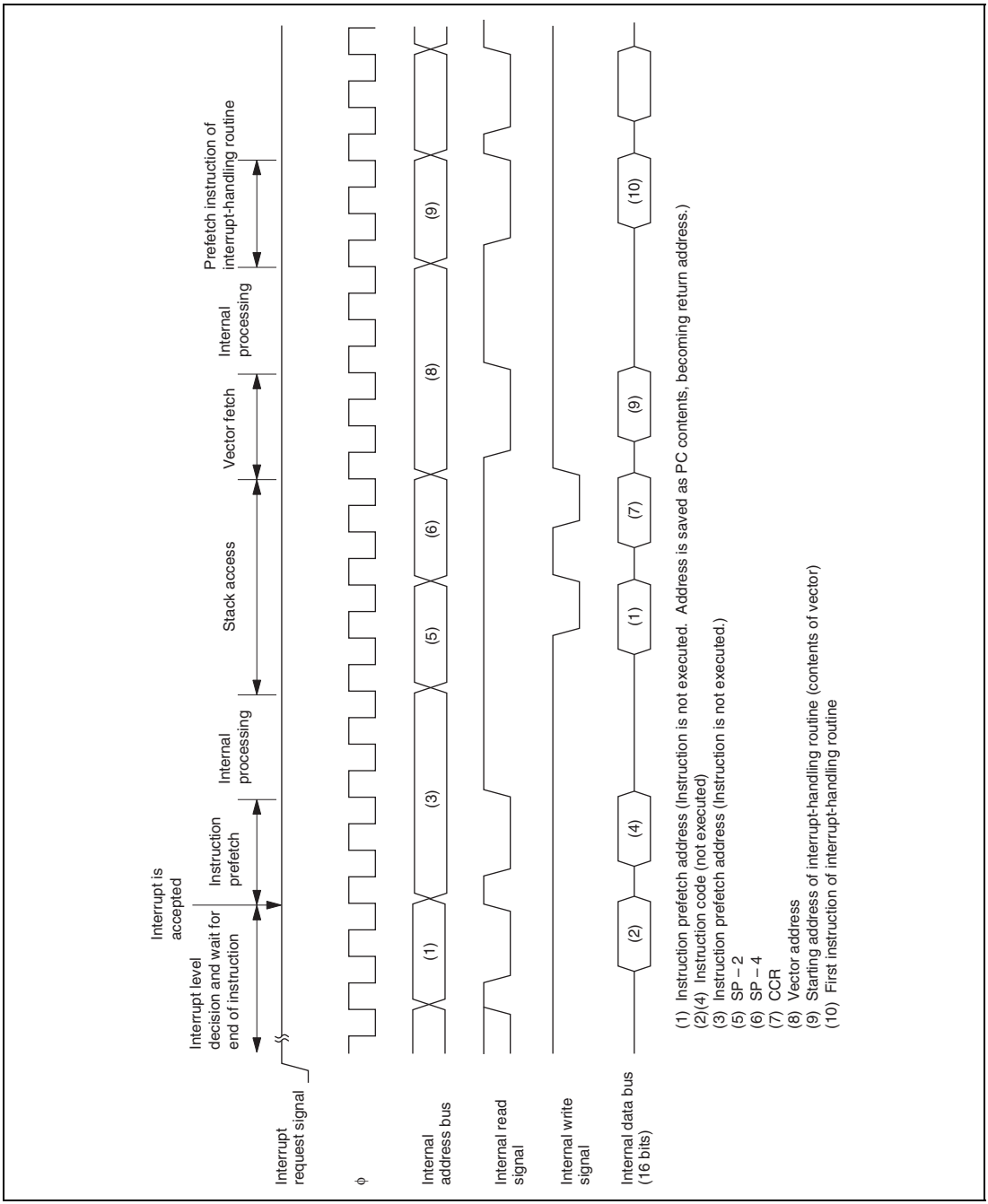



**Figure 3.5 Interrupt Sequence**

### 3.3.6 Interrupt Response Time

Table 3.4 shows the number of wait states after an interrupt request flag is set until the first instruction of the interrupt handler is executed.

**Table 3.4 Interrupt Wait States**

| Item | States | Total |
|---|---|---|
| Waiting time for completion of executing instruction* | 1 to 13 | 15 to 27 |
| Saving of PC and CCR to stack | 4 | |
| Vector fetch | 2 | |
| Instruction fetch | 4 | |
| Internal processing | 4 | |

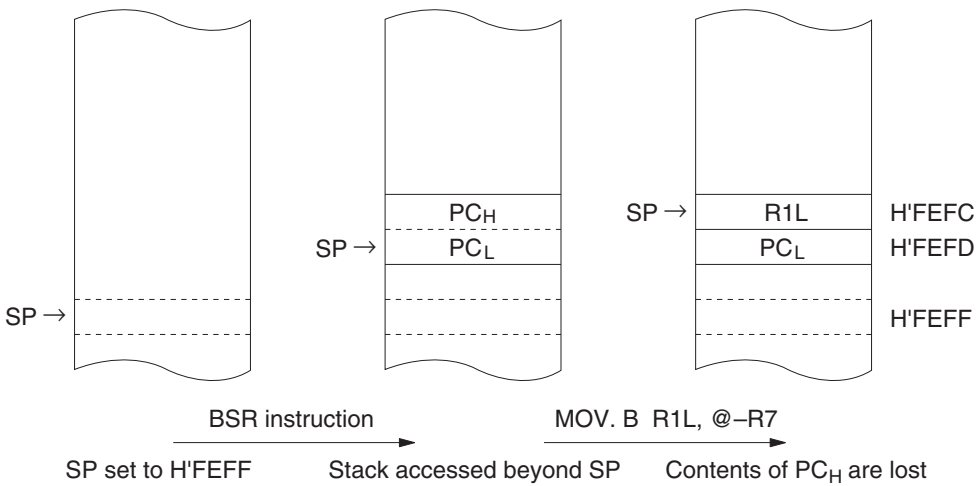
Note: * Not including EEPMOV instruction.

## 3.4 Application Notes

### 3.4.1 Notes on Stack Area Use

When word data is accessed in the LSI, the least significant bit of the address is regarded as 0. Access to the stack always takes place in word size, so the stack pointer (SP: R7) should never indicate an odd address. Use PUSH Rn (MOV.W Rn, @–SP) or POP Rn (MOV.W @SP+, Rn) to save or restore register values.

Setting an odd address in SP may cause a program to crash. An example is shown in figure 3.6.




Legend:
$PC_H$: Upper byte of program counter
$PC_L$: Lower byte of program counter
R1L: General register R1L
SP: Stack pointer

**Figure 3.6 Operation when Odd Address is Set in SP**

When CCR contents are saved to the stack during interrupt exception handling or restored when RTE is executed, this also takes place in word size. Both the upper and lower bytes of word data are saved to the stack; on return, the even address contents are restored to CCR while the odd address contents are ignored.

### 3.4.2 Notes on Rewriting Port Mode Registers

When a port mode register is rewritten to switch the functions of external interrupt pins, the following points should be observed.

When an external interrupt pin function is switched by rewriting the port mode register that controls pins $\overline{IRQ}_4$ to $\overline{IRQ}_0$, $\overline{WKP}_7$ to $\overline{WKP}_0$, the interrupt request flag may be set to 1 at the time the pin function is switched, even if no valid interrupt is input at the pin. Be sure to clear the interrupt request flag to 0 after switching pin functions. Table 3.5 shows the conditions under which interrupt request flags are set to 1 in this way.

**Table 3.5 Conditions Under which Interrupt Request Flag is Set to 1**

| Interrupt Request Flags Set to 1 | | Conditions |
|---|---|---|
| IRR1 | IRRI4 | When PMR1 bit IRQ4 is changed from 0 to 1 while pin $\overline{IRQ}_4$ is low and IEGR bit IEG4 = 0. |
| | | When PMR1 bit IRQ4 is changed from 1 to 0 while pin $\overline{IRQ}_4$ is low and IEGR bit IEG4 = 1. |
| | IRRI3 | When PMR1 bit IRQ3 is changed from 0 to 1 while pin $\overline{IRQ}_3$ is low and IEGR bit IEG3 = 0. |
| | | When PMR1 bit IRQ3 is changed from 1 to 0 while pin $\overline{IRQ}_3$ is low and IEGR bit IEG3 = 1. |
| | IRRI2 | When PMR1 bit IRQ2 is changed from 0 to 1 while pin $\overline{IRQ}_2$ is low and IEGR bit IEG2 = 0. |
| | | When PMR1 bit IRQ2 is changed from 1 to 0 while pin $\overline{IRQ}_2$ is low and IEGR bit IEG2 = 1. |
| | IRRI1 | When PMR1 bit IRQ1 is changed from 0 to 1 while pin $\overline{IRQ}_1$ is low and IEGR bit IEG1 = 0. |
| | | When PMR1 bit IRQ1 is changed from 1 to 0 while pin $\overline{IRQ}_1$ is low and IEGR bit IEG1 = 1. |
| | IRRI0 | When PMR3 bit IRQ0 is changed from 0 to 1 while pin $\overline{IRQ}_0$ is low and IEGR bit IEG0 = 0. |
| | | When PMR3 bit IRQ0 is changed from 1 to 0 while pin $\overline{IRQ}_0$ is low and IEGR bit IEG0 = 1. |
| IWPR | IWPF7 | When PMR5 bit WKP7 is changed from 0 to 1 while pin $\overline{WKP}_7$ is low. |
| | IWPF6 | When PMR5 bit WKP6 is changed from 0 to 1 while pin $\overline{WKP}_6$ is low. |
| | IWPF5 | When PMR5 bit WKP5 is changed from 0 to 1 while pin $\overline{WKP}_5$ is low. |
| | IWPF4 | When PMR5 bit WKP4 is changed from 0 to 1 while pin $\overline{WKP}_4$ is low. |
| | IWPF3 | When PMR5 bit WKP3 is changed from 0 to 1 while pin $\overline{WKP}_3$ is low. |
| | IWPF2 | When PMR5 bit WKP2 is changed from 0 to 1 while pin $\overline{WKP}_2$ is low. |
| | IWPF1 | When PMR5 bit WKP1 is changed from 0 to 1 while pin $\overline{WKP}_1$ is low. |
| | IWPF0 | When PMR5 bit WKP0 is changed from 0 to 1 while pin $\overline{WKP}_0$ is low. |

Figure 3.7 shows the procedure for setting a bit in a port mode register and clearing the interrupt request flag.

When switching a pin function, mask the interrupt before setting the bit in the port mode register. After accessing the port mode register, execute at least one instruction (e.g., NOP), then clear the interrupt request flag from 1 to 0. If the instruction to clear the flag is executed immediately after the port mode register access without executing an intervening instruction, the flag will not be cleared.

An alternative method is to avoid the setting of interrupt request flags when pin functions are switched by keeping the pins at the high level so that the conditions in table 3.5 do not occur.

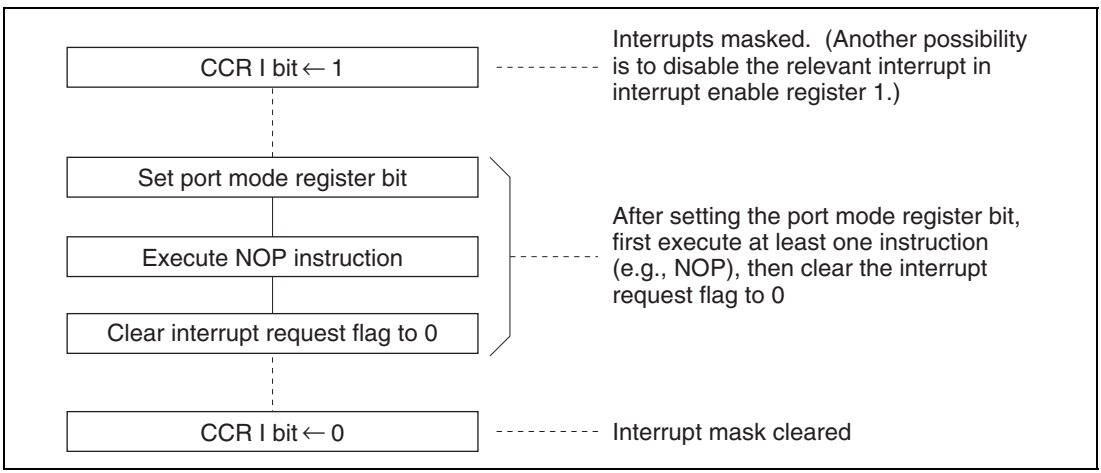



**Figure 3.7 Port Mode Register Setting and Interrupt Request Flag Clearing Procedure**

### 3.4.3 Method for Clearing Interrupt Request Flags

Use the recommended method, given below when clearing the flags of interrupt request registers (IRR1, IRR2, IWPR).

- Recommended method

  Use a single instruction to clear flags. The bit control instruction and byte-size data transfer instruction can be used. Two examples of program code for clearing IRRI1 (bit 1 of IRR1) are given below.

```
BCLR  #1,  @IRR1:8
MOV.B R1L, @IRR1:8 (set the value of R1L to B'11111101)
```

- Example of a malfunction

  When flags are cleared with multiple instructions, other flags might be cleared during execution of the instructions, even though they are currently set, and this will cause a malfunction.

  Here is an example in which IRRI0 is cleared and disabled in the process of clearing IRRI1 (bit 1 of IRR1).

```
MOV.B @IRR1:8,R1L ......... IRRI0 = 0 at this time
AND.B #B'11111101,R1L ..... Here, IRRI0 = 1
MOV.B R1L,@IRR1:8 ......... IRRI0 is cleared to 0
```

  In the above example, it is assumed that an IRQ0 interrupt is generated while the AND.B instruction is executing.

  The IRQ0 interrupt is disabled because, although the original objective is clearing IRRI1, IRRI0 is also cleared.

# Section 4 Clock Pulse Generators

## 4.1 Overview

Clock oscillator circuitry (CPG: clock pulse generator) is provided on-chip, including both a system clock pulse generator and a subclock pulse generator. The system clock pulse generator consists of a system clock oscillator and system clock dividers. The subclock pulse generator consists of a subclock oscillator circuit and a subclock divider.

### 4.1.1 Block Diagram

Figure 4.1 shows a block diagram of the clock pulse generators.

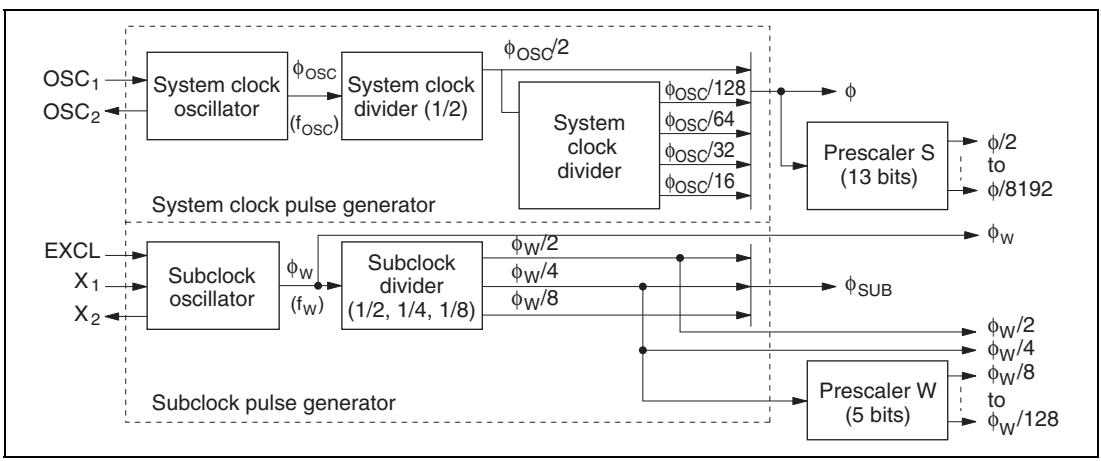



**Figure 4.1 Block Diagram of Clock Pulse Generators**

### 4.1.2 System Clock and Subclock

The basic clock signals that drive the CPU and on-chip peripheral modules are $\phi$ and $\phi_{SUB}$. Four of the clock signals have names: $\phi$ is the system clock, $\phi_{SUB}$ is the subclock, $\phi_{OSC}$ is the oscillator clock, and $\phi_w$ is the watch clock.

The clock signals available for use by peripheral modules are $\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, $\phi/32$, $\phi/64$, $\phi/128$, $\phi/256$, $\phi/512$, $\phi/1024$, $\phi/2048$, $\phi/4096$, $\phi/8192$, $\phi_w$, $\phi_w/2$, $\phi_w/4$, $\phi_w/8$, $\phi_w/16$, $\phi_w/32$, $\phi_w/64$, and $\phi_w/128$. The clock requirements differ from one module to another.

## 4.2 System Clock Generator

Clock pulses can be supplied to the system clock divider either by connecting a crystal or ceramic oscillator, or by providing external clock input.

### (1) Connecting a Crystal Oscillator

Figure 4.2 shows a typical method of connecting a crystal oscillator. For information on recommended resonators, see the product AC characteristics listed in section 16, Electrical Characteristics. Please consult with the resonator manufacturer when selecting a resonator model.

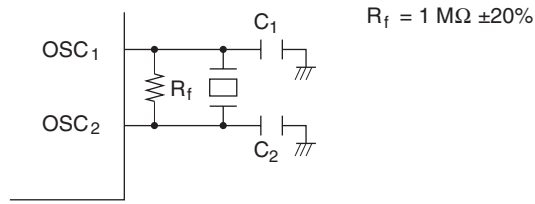



**Figure 4.2 Typical Connection to Crystal Oscillator**

### (2) Connecting a Ceramic Oscillator

Figure 4.3 shows a typical method of connecting a ceramic oscillator. For information on recommended resonators, see the product AC characteristics listed in section 16, Electrical Characteristics. Please consult with the resonator manufacturer when selecting a resonator model.

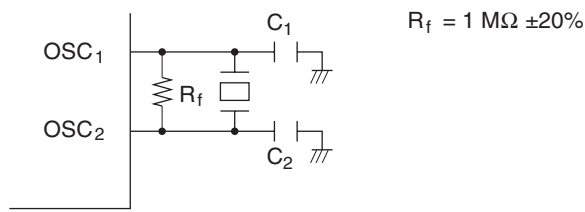



**Figure 4.3 Typical Connection to Ceramic Oscillator**

### (3) Notes on Board Design

When generating clock pulses by connecting a crystal or ceramic oscillator, pay careful attention to the following points.

Avoid running signal lines close to the oscillator circuit, since the oscillator may be adversely affected by induction currents. (See figure 4.4.)

The board should be designed so that the oscillator and load capacitors are located as close as possible to pins $OSC_1$ and $OSC_2$.

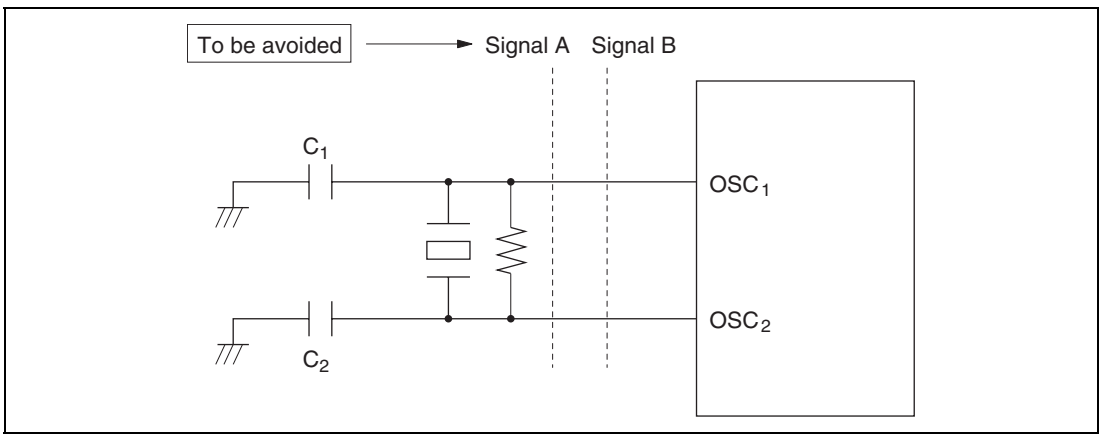



**Figure 4.4 Board Design of Oscillator Circuit**

### (4) External Clock Input Method

Connect an external clock signal to pin $OSC_1$, and leave pin $OSC_2$ open. Figure 4.5 shows a typical connection.

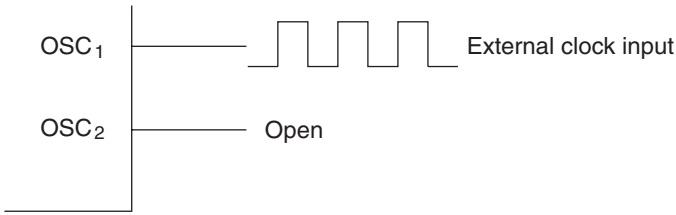



**Figure 4.5 External Clock Input (Example)**

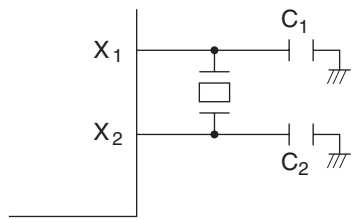
| Frequency | Oscillator Clock ($\phi_{osc}$) |
|---|---|
| Duty cycle | 45% to 55% |

Note: The circuit parameters above are recommended by the crystal or ceramic oscillator manufacturer.
The circuit parameters are affected by the crystal or ceramic oscillator and floating capacitance when designing the board. When using the oscillator, consult with the crystal or ceramic oscillator manufacturer to determine the circuit parameters.

## 4.3 Subclock Generator

### (1) Connecting a 32.768 kHz/38.4 kHz Crystal Oscillator

Clock pulses can be supplied to the subclock divider by connecting a 32.768 kHz/38.4 kHz crystal oscillator, as shown in figure 4.6. Follow the same precautions as noted under 3. notes on board design for the system clock in 4.2.



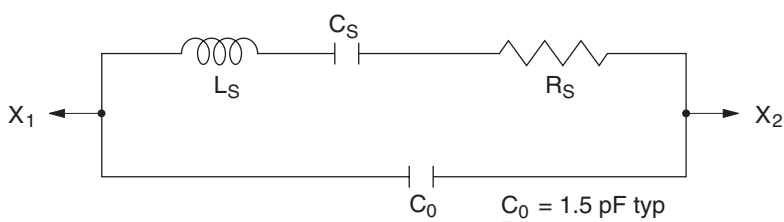

$C_1 = C_2 = 15$ pF (typ.)

Note: Circuit constants should be determined in consultation with the resonator manufacturer.

| Oscillation frequency | Manufacturer | Products Name |
|---|---|---|
| 38.4 kHz | Seiko Instrument Inc. | VTC-200 |
| 32.768 kHz | Nihon Denpa Kogyo | MX73P |

**Figure 4.6 Typical Connection to 32.768 kHz/38.4 kHz Crystal Oscillator (Subclock)**

Figure 4.7 shows the equivalent circuit of the 32.768 kHz/38.4 kHz crystal oscillator.




$C_0 = 1.5$ pF typ
$R_S = 14$ kΩ typ
$f_W = 32.768$ kHz/38.4kHz

**Figure 4.7 Equivalent Circuit of 32.768 kHz/38.4 kHz Crystal Oscillator**

### (2) Pin Connection when Not Using Subclock

When the subclock is not used, connect pin $X_1$ to GND and leave pin $X_2$ open, as shown in figure 4.8.

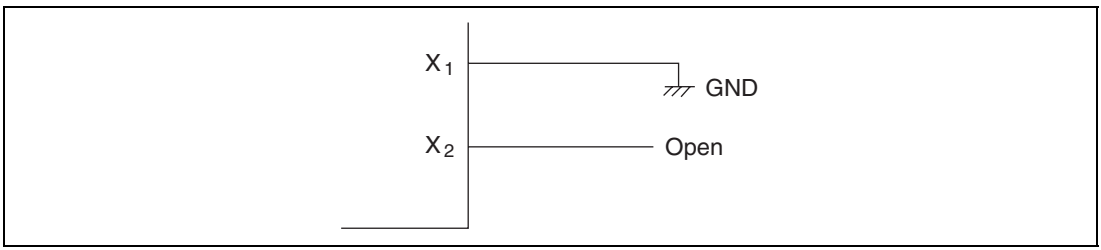



**Figure 4.8 Pin Connection when not Using Subclock**

### (3) External Clock Input

Connect pin $X_1$ to GND and leave pin $X_2$ open. Input an external clock to pin EXCL. Set bit EXCL in register PMR2 to 1 to supply the external clock to the internal components of the device. A connection example is shown in figure 4.9.

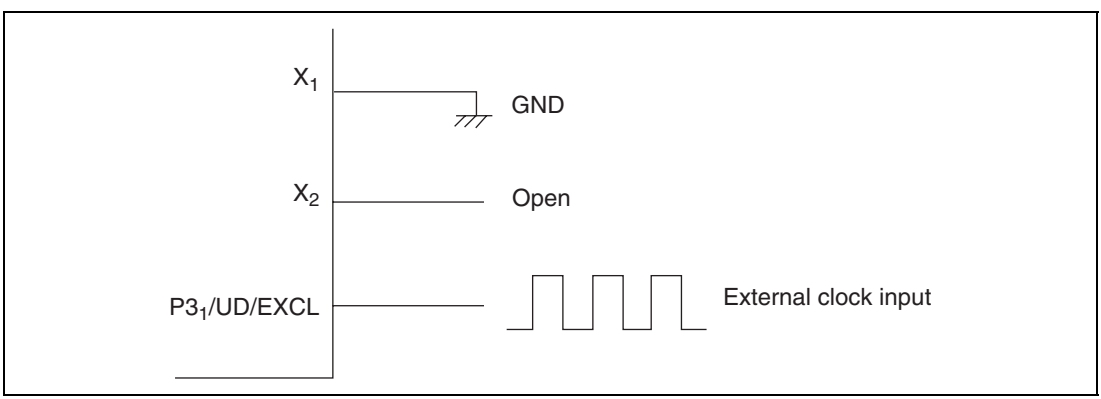



**Figure 4.9 Pin Connection when Inputting External Clock**

| Frequency | Subclock (ϕw) |
|---|---|
| Duty | 45% to 55% |

## 4.4 Prescalers

This LSI is equipped with two on-chip prescalers having different input clocks (prescaler S and prescaler W). Prescaler S is a 13-bit counter using the system clock (ϕ) as its input clock. Its prescaled outputs provide internal clock signals for on-chip peripheral modules. Prescaler W is a 5-bit counter using a 32.768 kHz or 38.4 kHz signal divided by 4 ($\phi_w/4$) as its input clock. Its prescaled outputs are used by timer A as a time base for timekeeping.

### (1) Prescaler S (PSS)

Prescaler S is a 13-bit counter using the system clock (ϕ) as its input clock. It is incremented once per clock period.

Prescaler S is initialized to H'0000 by a reset, and starts counting on exit from the reset state.

In standby mode, watch mode, subactive mode, and subsleep mode, the system clock pulse generator stops. Prescaler S also stops and is initialized to H'0000.

The CPU cannot read or write prescaler S.

The output from prescaler S is shared by timer A, timer C, timer F, timer G, SCI3-1, SC3-2, the A/D converter, the LCD controller, the watchdog timer, and the 14-bit PWM. The divider ratio can be set separately for each on-chip peripheral function.

In active (medium-speed) mode the clock input to prescaler S is ϕosc/16, ϕosc/32, ϕosc/64, or ϕosc/128.

### (2) Prescaler W (PSW)

Prescaler W is a 5-bit counter using a 32.768 kHz/38.4 kHz signal divided by 4 ($\phi_w/4$) as its input clock.

Prescaler W is initialized to H'00 by a reset, and starts counting on exit from the reset state.

Even in standby mode, watch mode, subactive mode, or subsleep mode, prescaler W continues functioning so long as clock signals are supplied to pins X1 and X2.

Prescaler W can be reset by setting 1s in bits TMA3 and TMA2 of timer mode register A (TMA).

Output from prescaler W can be used to drive timer A, in which case timer A functions as a time base for timekeeping.

## 4.5 Note on Oscillators

Oscillator characteristics are closely related to board design and should be carefully evaluated by the user, referring to the examples shown in this section. Oscillator circuit constants will differ depending on the oscillator element, stray capacitance in its interconnecting circuit, and other factors. Suitable constants should be determined in consultation with the oscillator element manufacturer. Design the circuit so that the oscillator element never receives voltages exceeding its maximum rating.

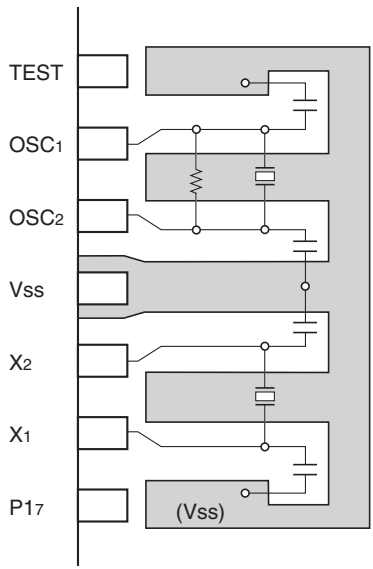



**Figure 4.10 Example of Crystal and Ceramic Oscillator Element Arrangement**

Figure 4.11 (1) shows an example measuring circuit with the negative resistance suggested by the oscillator manufacturer. Note that if the negative resistance of the circuit is less than that suggested by the oscillator manufacturer, it may be difficult to start the main oscillator.

If it is determined that oscillation is not occurring because the negative resistance is lower than the level suggested by the oscillator manufacturer, the circuit may be modified as shown in figure 4.11 (2) through (4). Which of the modification suggestions to use and the capacitor capacitance should be decided based upon an evaluation of factors such as the negative resistance and the frequency deviation.

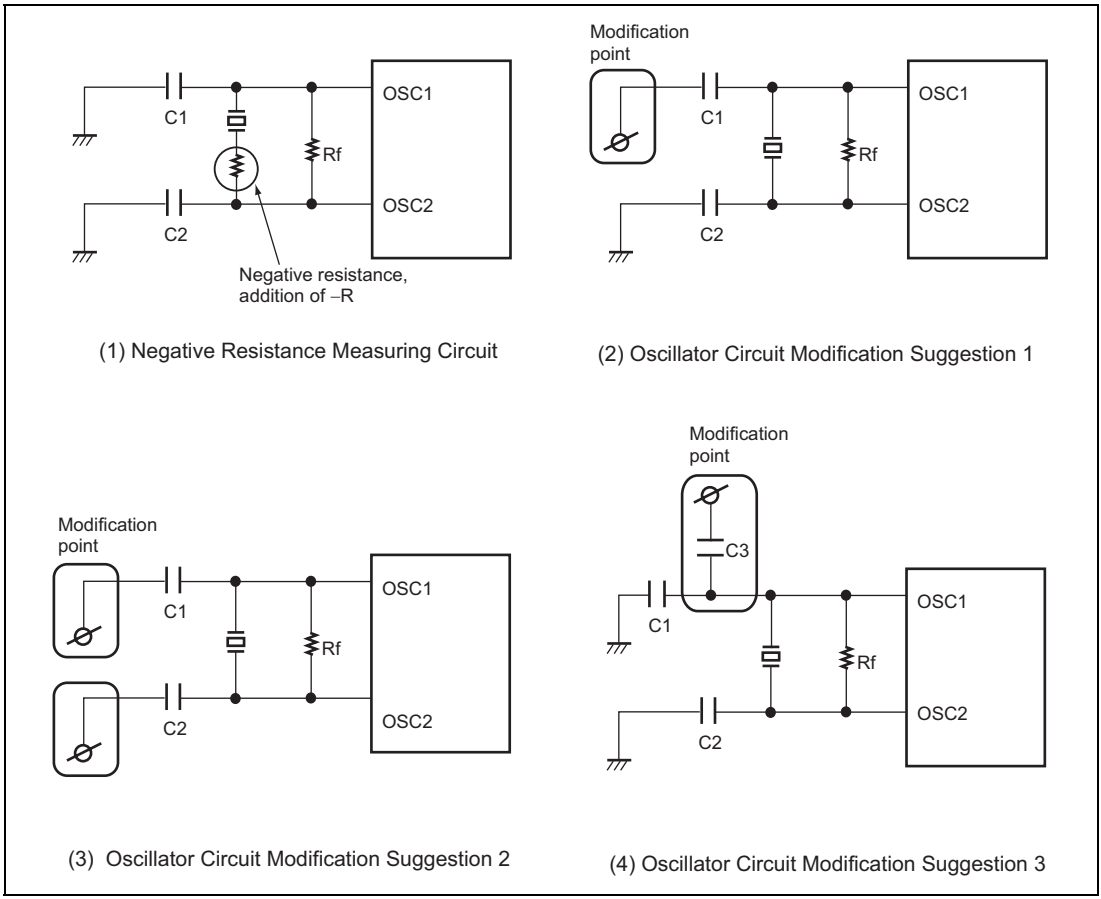



**Figure 4.11 Negative Resistance Measurement and Circuit Modification Suggestions**

### 4.5.1 Definition of Oscillation Stabilization Wait Time

Figure 4.12 shows the oscillation waveform ($OSC_2$), system clock (ϕ), and microcomputer operating mode when a transition is made from standby mode, watch mode, or subactive mode, to active (high-speed/medium-speed) mode, with an oscillator element connected to the system clock oscillator.

As shown in figure 4.12, as the system clock oscillator is halted in standby mode, watch mode, and subactive mode, when a transition is made to active (high-speed/medium-speed) mode, the sum of the following two times (oscillation stabilization time and wait time) is required.

### (1) Oscillation Stabilization Time ($t_{rc}$)

The time from the point at which the system clock oscillator oscillation waveform starts to change when an interrupt is generated, until the amplitude of the oscillation waveform increases and the oscillation frequency stabilizes.

### (2) Wait Time

The time required for the CPU and peripheral functions to begin operating after the oscillation waveform frequency and system clock have stabilized.

The wait time setting is selected with standby timer select bits 2 to 0 (STS2 to STS0) (bits 6 to 4 in system control register 1 (SYSCR1)).

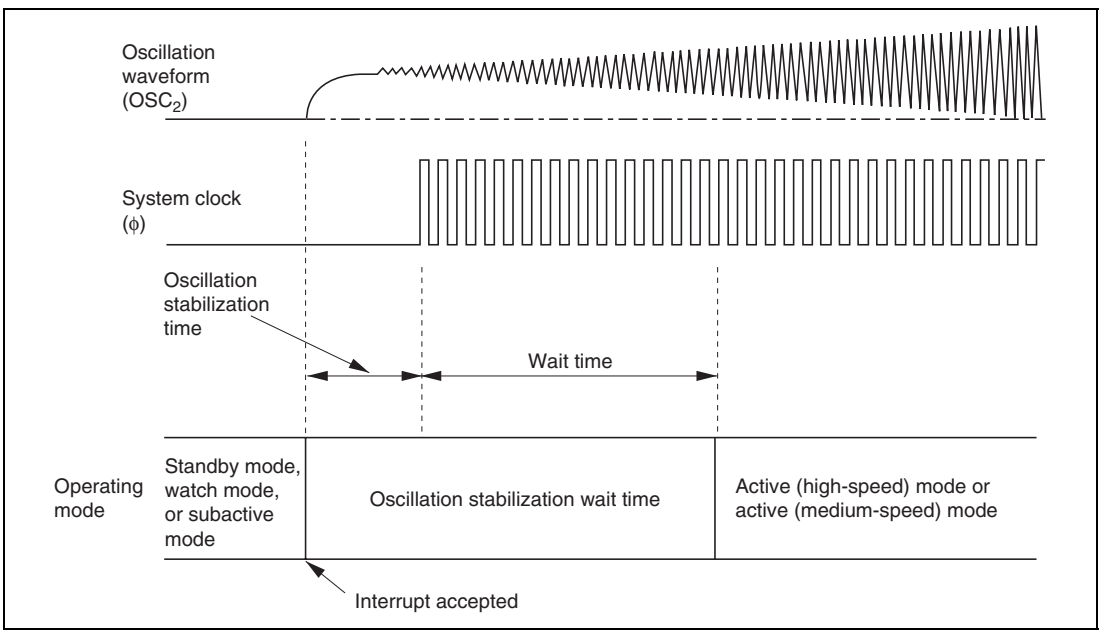



**Figure 4.12 Oscillation Stabilization Wait Time**

When standby mode, watch mode, or subactive mode is cleared by an interrupt or reset, and a transition is made to active (high-speed/medium-speed) mode, the oscillation waveform begins to change at the point at which the interrupt is accepted. Therefore, when an oscillator element is connected in standby mode, watch mode, or subactive mode, since the system clock oscillator is halted, the time from the point at which this oscillation waveform starts to change until the amplitude of the oscillation waveform increases and the oscillation frequency stabilizes—that is, the oscillation stabilization time—is required.

The oscillation stabilization time in the case of these state transitions is the same as the oscillation stabilization time at power-on (the time from the point at which the power supply voltage reaches the prescribed level until the oscillation stabilizes), specified by "oscillation stabilization time $t_{rc}$" in the AC characteristics.

Meanwhile, once the system clock has halted, a wait time of at least 8 states is necessary in order for the CPU and peripheral functions to operate normally.

Thus, the time required from interrupt generation until operation of the CPU and peripheral functions is the sum of the above described oscillation stabilization time and wait time. This total time is called the oscillation stabilization wait time, and is expressed by equation (1) below.

Oscillation stabilization wait time = oscillation stabilization time + wait time
= $t_{rc}$ + (8 to 131,072 states) ..... (1)

Therefore, when a transition is made from standby mode, watch mode, or subactive mode, to active (high-speed/medium-speed) mode, with an oscillator element connected to the system clock oscillator, careful evaluation must be carried out on the installation circuit before deciding on the oscillation stabilization wait time. In particular, since the oscillation stabilization time is affected by installation circuit constants, stray capacitance, and so forth, suitable constants should be determined in consultation with the oscillator element manufacturer.

### 4.5.2 Notes on Use of Crystal Oscillator Element (Excluding Ceramic Oscillator Element)

When a microcomputer operates, the internal power supply potential fluctuates slightly in synchronization with the system clock. Depending on the individual crystal oscillator element characteristics, the oscillation waveform amplitude may not be sufficiently large immediately after the oscillation stabilization wait time, making the oscillation waveform susceptible to influence by fluctuations in the power supply potential. In this state, the oscillation waveform may be disrupted, leading to an unstable system clock and erroneous operation of the microcomputer.

If erroneous operation occurs, change the setting of standby timer select bits 2 to 0 (STS2 to STS0) (bits 6 to 4 in system control register 1 (SYSCR1)) to give a longer wait time.

For example, if erroneous operation occurs with a wait time setting of 16 states, check the operation with a wait time setting of 8,192 states or more.

If the same kind of erroneous operation occurs after a reset as after a state transition, hold the $\overline{RES}$ pin low for a longer period.

# Section 5 Power-Down Modes

## 5.1 Overview

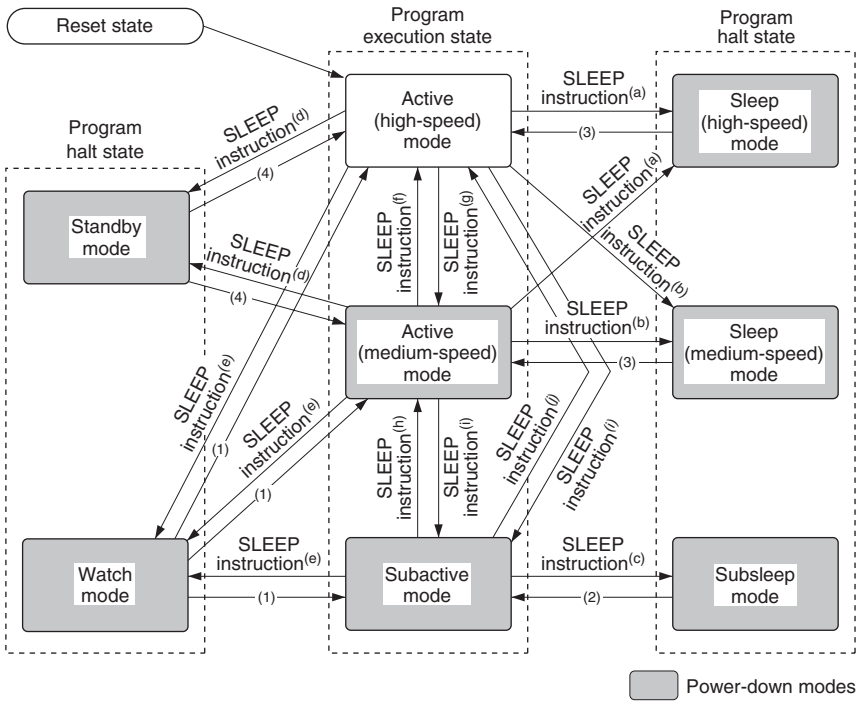
The LSI has nine modes of operation after a reset. These include eight power-down modes, in which power dissipation is significantly reduced. Table 5.1 gives a summary of the nine operating modes.

**Table 5.1 Operating Modes**

| Operating Mode | Description |
|---|---|
| Active (high-speed) mode | The CPU and all on-chip peripheral functions are operable on the system clock in high-speed operation |
| Active (medium-speed) mode | The CPU and all on-chip peripheral functions are operable on the system clock in low-speed operation |
| Subactive mode | The CPU is operable on the subclock in low-speed operation |
| Sleep (high-speed) mode | The CPU halts. On-chip peripheral functions are operable on the system clock |
| Sleep (medium-speed) mode | The CPU halts. On-chip peripheral functions operate at a frequency of 1/64, 1/32, 1/16, or 1/8 of the system clock frequency |
| Subsleep mode | The CPU halts. The time-base function of timer A, timer C, timer G, timer F,WDT, SCI3-1, SCI3-2, AEC, and LCD controller/driver are operable on the subclock |
| Watch mode | The CPU halts. The time-base function of timer A, timer F, timer G, AEC, and LCD controller/driver are operable on the subclock |
| Standby mode | The CPU and all on-chip peripheral functions halt |
| Module standby mode | Individual on-chip peripheral functions specified by software enter standby mode and halt |

Of these nine operating modes, all but the active (high-speed) mode are power-down modes. In this section the two active modes (high-speed and medium speed) will be referred to collectively as active mode.

Figure 5.1 shows the transitions among these operation modes. Table 5.2 indicates the internal states in each mode.




**Mode Transition Conditions (1)**

| | LSON | MSON | SSBY | TMA3 | DTON |
|---|---|---|---|---|---|
| (a) | 0 | 0 | 0 | * | 0 |
| (b) | 0 | 1 | 0 | * | 0 |
| (c) | 1 | * | 0 | 1 | 0 |
| (d) | 0 | * | 1 | 0 | 0 |
| (e) | * | * | 1 | 1 | 0 |
| (f) | 0 | 0 | 0 | * | 1 |
| (g) | 0 | 1 | 0 | * | 1 |
| (h) | 0 | 1 | 1 | 1 | 1 |
| (i) | 1 | * | 1 | 1 | 1 |
| (J) | 0 | 0 | 1 | 1 | 1 |

* : Don't care

**Mode Transition Conditions (2)**

| | Interrupt Sources |
|---|---|
| (1) | Timer A, Timer F, Timer G interrupt, $IRQ_0$ interrupt, $WKP_7$ to $WKP_0$ interrupts |
| (2) | Timer A, Timer C, Timer F, Timer G, SCI3-1, SCI3-2 interrupt, $IRQ_4$ to $IRQ_0$ interrupts, $WKP_7$ to $WKP_0$ interrupts, AEC |
| (3) | All interrupts |
| (4) | $IRQ_1$ or $IRQ_0$ interrupt, $WKP_7$ to $WKP_0$ interrupts |

Notes: 1. A transition between different modes cannot be made to occur simply because an interrupt request is generated. Make sure that interrupt handling is performed after the interrupt is accepted.
2. Details on the mode transition conditions are given in the explanations of each mode, in sections 5.2 to 5.9.

**Figure 5.1 Mode Transition Diagram**

## Table 5.2 Internal State in Each Operating Mode

| Function | | Active Mode | | Sleep Mode | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | High-Speed | Medium-Speed | High-Speed | Medium-Speed | Watch Mode | Subactive Mode | Subsleep Mode | Standby Mode |
| System clock oscillator | | Functions | Functions | Functions | Functions | Halted | Halted | Halted | Halted |
| Subclock oscillator | | Functions | Functions | Functions | Functions | Functions | Functions | Functions | Functions |
| CPU operations | Instructions | Functions | Functions | Halted | Halted | Halted | Functions | Halted | Halted |
| | RAM | | | Retained | Retained | Retained | | Retained | Retained |
| | Registers | | | | | | | | |
| | I/O ports | | | | | | | | Retained*1 |
| External interrupts | $IRQ_0$ | Functions | Functions | Functions | Functions | Functions | Functions | Functions | Functions |
| | $IRQ_1$ | | | | | Retained*6 | | | |
| | $IRQ_2$ | | | | | | | | Retained*6 |
| | $IRQ_3$ | | | | | | | | |
| | $IRQ_4$ | | | | | | | | |
| | $WKP_0$ | Functions | Functions | Functions | Functions | Functions | Functions | Functions | Functions |
| | $WKP_1$ | | | | | | | | |
| | $WKP_2$ | | | | | | | | |
| | $WKP_3$ | | | | | | | | |
| | $WKP_4$ | | | | | | | | |
| | $WKP_5$ | | | | | | | | |
| | $WKP_6$ | | | | | | | | |
| | $WKP_7$ | | | | | | | | |
| Peripheral functions | Timer A | Functions | Functions | Functions | Functions | Functions*5 | Functions*5 | Functions*5 | Retained |
| | Asynchronous counter | | | | | Functions*8 | Functions | Functions | Functions*8 |
| | Timer C | | | | | Retained | Functions/ Retained*2 | Functions/ Retained*2 | Retained |
| | WDT | | | | | | Functions/ Retained*7 | Retained | |
| | Timer G, Timer F | | | | | Functions/ Retained*9 | Functions/ Retained*2 | Functions/ Retained*2 | |
| | SCI3-1 | | | | | Reset | Functions/ Retained*3 | Functions/ Retained*3 | Reset |
| | SCI3-2 | | | | | | | | |
| | PWM | | | | | Retained | Retained | Retained | Retained |
| | A/D converter | | | | | Retained | Retained | Retained | Retained |
| | LCD | | | | | Functions/ Retained*4 | Functions/ Retained*4 | Functions/ Retained*4 | Retained |

Notes:
1. Register contents are retained, but output is high-impedance state.
2. Functions if an external clock or the $\phi_W/4$ internal clock is selected; otherwise halted and retained.
3. Functions if $\phi_W/2$ is selected as the internal clock; otherwise halted and retained.
4. Functions if $\phi_W$, $\phi_W/2$ or $\phi_W/4$ is selected as the operating clock; otherwise halted and retained.
5. Functions if the timekeeping time-base function is selected.
6. External interrupt requests are ignored. Interrupt request register contents are not altered.
7. Functions if $\phi_W/32$ is selected as the internal clock; otherwise halted and retained.
8. Incrementing is possible, but interrupt generation is not.
9. Functions if the $\phi_W/4$ internal clock is selected; otherwise halted and retained.

### 5.1.1 System Control Registers

The operation mode is selected using the system control registers described in table 5.3.

**Table 5.3 System Control Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| System control register 1 | SYSCR1 | R/W | H'07 | H'FFF0 |
| System control register 2 | SYSCR2 | R/W | H'F0 | H'FFF1 |

#### (1) System Control Register 1 (SYSCR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | LSON | — | MA1 | MA0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

SYSCR1 is an 8-bit read/write register for control of the power-down modes.

Upon reset, SYSCR1 is initialized to H'07.

**Bit 7:** Software standby (SSBY)

This bit designates transition to standby mode or watch mode.

| Bit 7<br>SSBY | Description |
|---|---|
| 0 | • When a SLEEP instruction is executed in active mode, a transition is made to sleep mode (initial value)<br>• When a SLEEP instruction is executed in subactive mode, a transition is made to subsleep mode |
| 1 | • When a SLEEP instruction is executed in active mode, a transition is made to standby mode or watch mode<br>• When a SLEEP instruction is executed in subactive mode, a transition is made to watch mode |

**Bits 6 to 4:** Standby timer select 2 to 0 (STS2 to STS0)

These bits designate the time the CPU and peripheral modules wait for stable clock operation after exiting from standby mode or watch mode to active mode due to an interrupt. The designation should be made according to the operating frequency so that the waiting time is at least equal to the oscillation settling time.

| Bit 6<br>STS2 | Bit 5<br>STS1 | Bit 4<br>STS0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | Wait time = 8,192 states | (initial value) |
| 0 | 0 | 1 | Wait time = 16,384 states | |
| 0 | 1 | 0 | Wait time = 32,768 states | |
| 0 | 1 | 1 | Wait time = 65,536 states | |
| 1 | 0 | 0 | Wait time = 131,072 states | |
| 1 | 0 | 1 | Wait time = 2 states | (External clock mode) |
| 1 | 1 | 0 | Wait time = 8 states | |
| 1 | 1 | 1 | Wait time = 16 states | |

Note: In the case that external clock is input, set up the “Standby timer select” selection to External clock mode before Mode Transition. Also, do not set up to external clock mode, in the case that it does not use external clock.

**Bit 3:** Low speed on flag (LSON)

This bit chooses the system clock ($\phi$) or subclock ($\phi_{SUB}$) as the CPU operating clock when watch mode is cleared. The resulting operation mode depends on the combination of other control bits and interrupt input.

| Bit 3<br>LSON | Description | |
|---|---|---|
| 0 | The CPU operates on the system clock ($\phi$) | (initial value) |
| 1 | The CPU operates on the subclock ($\phi_{SUB}$) | |

**Bit 2:** Reserved bit

Bit 2 is reserved: it is always read as 1 and cannot be modified.

**Bits 1 and 0:** Active (medium-speed) mode clock select (MA1, MA0)

Bits 1 and 0 choose $\phi_{osc}/128$, $\phi_{osc}/64$, $\phi_{osc}/32$, or $\phi_{osc}/16$ as the operating clock in active (medium-speed) mode and sleep (medium-speed) mode. MA1 and MA0 should be written in active (high-speed) mode or subactive mode.

| Bit 1 MA1 | Bit 0 MA0 | Description |
|---|---|---|
| 0 | 0 | $\phi_{osc}/16$ |
| 0 | 1 | $\phi_{osc}/32$ |
| 1 | 0 | $\phi_{osc}/64$ |
| 1 | 1 | $\phi_{osc}/128$ (initial value) |

### (2) System Control Register 2 (SYSCR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | NESEL | DTON | MSON | SA1 | SA0 |
| Initial value | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | — | R/W | R/W | R/W | R/W | R/W |

SYSCR2 is an 8-bit read/write register for power-down mode control.

**Bits 7 to 5:** Reserved bits

These bits are reserved; they are always read as 1, and cannot be modified.

**Bit 4:** Noise elimination sampling frequency select (NESEL)

This bit selects the frequency at which the watch clock signal ($\phi_w$) generated by the subclock pulse generator is sampled, in relation to the oscillator clock ($\phi_{osc}$) generated by the system clock pulse generator. When $\phi_{osc}$ = 2 to 16 MHz, clear NESEL to 0.

| Bit 4 NESEL | Description |
|---|---|
| 0 | Sampling rate is $\phi_{osc}/16$ |
| 1 | Sampling rate is $\phi_{osc}/4$ (initial value) |

**Bit 3:** Direct transfer on flag (DTON)

This bit designates whether or not to make direct transitions among active (high-speed), active (medium-speed) and subactive mode when a SLEEP instruction is executed. The mode to which the transition is made after the SLEEP instruction is executed depends on a combination of this and other control bits.

| Bit 3<br>DTON | Description |
|---|---|
| 0 | • When a SLEEP instruction is executed in active mode, a transition is made to standby mode, watch mode, or sleep mode (initial value)<br>• When a SLEEP instruction is executed in subactive mode, a transition is made to watch mode or subsleep mode |
| 1 | • When a SLEEP instruction is executed in active (high-speed) mode, a direct transition is made to active (medium-speed) mode if SSBY = 0, MSON = 1, and LSON = 0, or to subactive mode if SSBY = 1, TMA3 = 1, and LSON = 1<br>• When a SLEEP instruction is executed in active (medium-speed) mode, a direct transition is made to active (high-speed) mode if SSBY = 0, MSON = 0, and LSON = 0, or to subactive mode if SSBY = 1, TMA3 = 1, and LSON = 1<br>• When a SLEEP instruction is executed in subactive mode, a direct transition is made to active (high-speed) mode if SSBY = 1, TMA3 = 1, LSON = 0, and MSON = 0, or to active (medium-speed) mode if SSBY = 1, TMA3 = 1, LSON = 0, and MSON = 1 |

**Bit 2:** Medium speed on flag (MSON)

After standby, watch, or sleep mode is cleared, this bit selects active (high-speed) or active (medium-speed) mode.

| Bit 2<br>MSON | Description |
|---|---|
| 0 | Operation in active (high-speed) mode (initial value) |
| 1 | Operation in active (medium-speed) mode |

**Bits 1 and 0:** Subactive mode clock select (SA1 and SA0)

These bits select the CPU clock rate ($\phi_w/2$, $\phi_w/4$, or $\phi_w/8$) in subactive mode. SA1 and SA0 cannot be modified in subactive mode.

| Bit 1<br>SA1 | Bit 0<br>SA0 | Description |
|---|---|---|
| 0 | 0 | $\phi_w/8$ (initial value) |
| 0 | 1 | $\phi_w/4$ |
| 1 | * | $\phi_w/2$ |

* : Don't care

## 5.2 Sleep Mode

### 5.2.1 Transition to Sleep Mode

#### (1) Transition to Sleep (High-Speed) Mode

The system goes from active mode to sleep (high-speed) mode when a SLEEP instruction is executed while the SSBY and LSON bits in SYSCR1 are cleared to 0, the MSON and DTON bits in SYSCR2 are cleared to 0. In sleep mode CPU operation is halted but the on-chip peripheral functions. CPU register contents are retained.

#### (2) Transition to Sleep (Medium-Speed) Mode

The system goes from active mode to sleep (medium-speed) mode when a SLEEP instruction is executed while the SSBY and LSON bits in SYSCR1 are cleared to 0, the MSON bit in SYSCR2 is set to 1, and the DTON bit in SYSCR2 is cleared to 0. In sleep (medium-speed) mode, as in sleep (high-speed) mode, CPU operation is halted but the on-chip peripheral functions are operational. The clock frequency in sleep (medium-speed) mode is determined by the MA1 and MA0 bits in SYSCR1. CPU register contents are retained.
Furthermore, it sometimes acts with half state early timing at the time of transition to sleep (medium-speed) mode.

### 5.2.2 Clearing Sleep Mode

Sleep mode is cleared by any interrupt (timer A, timer C, timer F, timer G, asynchronous counter, $IRQ_4$ to $IRQ_0$, $WKP_7$ to $WKP_0$, SCI3-1, SCI3-2, A/D converter, or), or by input at the $\overline{RES}$ pin.

- Clearing by interrupt
  When an interrupt is requested, sleep mode is cleared and interrupt exception handling starts. A transition is made from sleep (high-speed) mode to active (high-speed) mode, or from sleep (medium-speed) mode to active (medium-speed) mode. Sleep mode is not cleared if the I bit of the condition code register (CCR) is set to 1 or the particular interrupt is disabled in the interrupt enable register.
  Interrupt signal and system clock are mutually asynchronous. Synchronization error time in a maximum is 2/ϕ (s).

- Clearing by $\overline{RES}$ input
  When the $\overline{RES}$ pin goes low, the CPU goes into the reset state and sleep mode is cleared.

### 5.2.3 Clock Frequency in Sleep (Medium-Speed) Mode

Operation in sleep (medium-speed) mode is clocked at the frequency designated by the MA1 and MA0 bits in SYSCR1.

## 5.3 Standby Mode

### 5.3.1 Transition to Standby Mode

The system goes from active mode to standby mode when a SLEEP instruction is executed while the SSBY bit in SYSCR1 is set to 1, the LSON bit in SYSCR1 is cleared to 0, and bit TMA3 in TMA is cleared to 0. In standby mode the clock pulse generator stops, so the CPU and on-chip peripheral modules stop functioning, but as long as the rated voltage is supplied, the contents of CPU registers, on-chip RAM, and some on-chip peripheral module registers are retained. On-chip RAM contents will be further retained down to a minimum RAM data retention voltage. The I/O ports go to the high-impedance state.

### 5.3.2 Clearing Standby Mode

Standby mode is cleared by an interrupt ($IRQ_1$ or $IRQ_0$), $WKP_7$ to $WKP_0$ or by input at the $\overline{RES}$ pin.

- Clearing by interrupt
  When an interrupt is requested, the system clock pulse generator starts. After the time set in bits STS2 to STS0 in SYSCR1 has elapsed, a stable system clock signal is supplied to the entire chip, standby mode is cleared, and interrupt exception handling starts. Operation resumes in active (high-speed) mode if MSON = 0 in SYSCR2, or active (medium-speed) mode if MSON = 1. Standby mode is not cleared if the I bit of CCR is set to 1 or the particular interrupt is disabled in the interrupt enable register.

- Clearing by $\overline{RES}$ input
  When the $\overline{RES}$ pin goes low, the system clock pulse generator starts. After the pulse generator output has stabilized, if the $\overline{RES}$ pin is driven high, the CPU starts reset exception handling. Since system clock signals are supplied to the entire chip as soon as the system clock pulse generator starts functioning, the $\overline{RES}$ pin should be kept at the low level until the pulse generator output stabilizes.

### 5.3.3 Oscillator Settling Time after Standby Mode is Cleared

Bits STS2 to STS0 in SYSCR1 should be set as follows.

- When a crystal oscillator is used

  The table below gives settings for various operating frequencies. Set bits STS2 to STS0 for a waiting time at least as long as the oscillation settling time.

**Table 5.4 Clock Frequency and Settling Time (Times are in ms)**

| STS2 | STS1 | STS0 | Waiting Time | 2 MHz | 1 MHz |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 8,192 states | 4.1 | 8.2 |
| 0 | 0 | 1 | 16,384 states | 8.2 | 16.4 |
| 0 | 1 | 0 | 32,768 states | 16.4 | 32.8 |
| 0 | 1 | 1 | 65,536 states | 32.8 | 65.5 |
| 1 | 0 | 0 | 131,072 states | 65.5 | 131.1 |
| 1 | 0 | 1 | 2 states (Use prohibited) | 0.001 | 0.002 |
| 1 | 1 | 0 | 8 states | 0.004 | 0.008 |
| 1 | 1 | 1 | 16 states | 0.008 | 0.016 |

- When an external clock is used

  STS2 = 1, STS1 = 0, and STS0 = 1 should be set. Other values possible use, but CPU sometimes will start operation before waiting time completion.

### 5.3.4 Standby Mode Transition and Pin States

When a SLEEP instruction is executed in active (high-speed) mode or active (medium-speed) mode while bit SSBY is set to 1 and bit LSON is cleared to 0 in SYSCR1, and bit TMA3 is cleared to 0 in TMA, a transition is made to standby mode. At the same time, pins go to the high-impedance state (except pins for which the pull-up MOS is designated as on). Figure 5.2 shows the timing in this case.

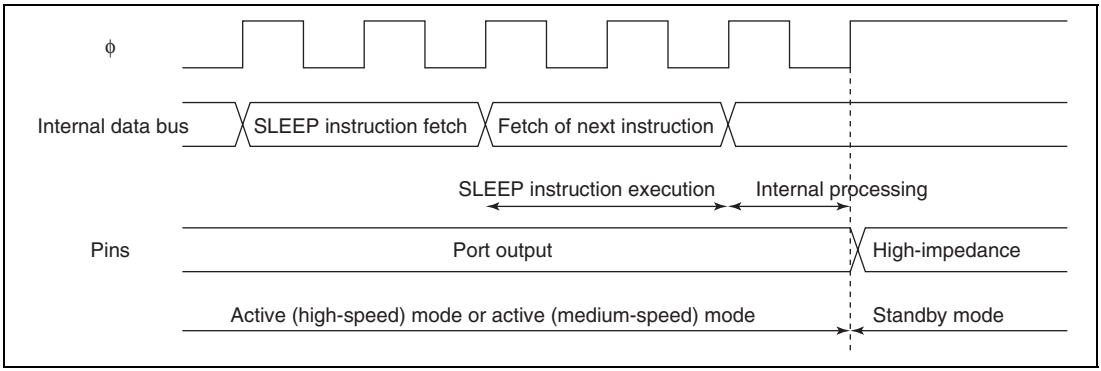



**Figure 5.2 Standby Mode Transition and Pin States**

### 5.3.5 Notes on External Input Signal Changes before/after Standby Mode

1. When external input signal changes before/after standby mode or watch mode

   When an external input signal such as $\overline{IRQ}$ or $\overline{WKP}$ is input, both the high- and low-level widths of the signal must be at least two cycles of system clock ϕ or subclock $\phi_{SUB}$ (referred to together in this section as the internal clock). As the internal clock stops in standby mode and watch mode, the width of external input signals requires careful attention when a transition is made via these operating modes. Ensure that external input signals conform to the conditions stated in 3, Recommended timing of external input signals, below.

2. When external input signals cannot be captured because internal clock stops

   The case of falling edge capture is illustrated in figure 5.3

   As shown in the case marked "Capture not possible," when an external input signal falls immediately after a transition to active (high-speed or medium-speed) mode or subactive mode, after oscillation is started by an interrupt via a different signal, the external input signal cannot be captured if the high-level width at that point is less than 2 $t_{cyc}$ or 2 $t_{subcyc}$.

3. Recommended timing of external input signals

   To ensure dependable capture of an external input signal, high- and low-level signal widths of at least 2 $t_{cyc}$ or 2 $t_{subcyc}$ are necessary before a transition is made to standby mode or watch mode, as shown in "Capture possible: case 1."

   External input signal capture is also possible with the timing shown in "Capture possible: case 2" and "Capture possible: case 3," in which a 2 $t_{cyc}$ or 2 $t_{subcyc}$ level width is secured.

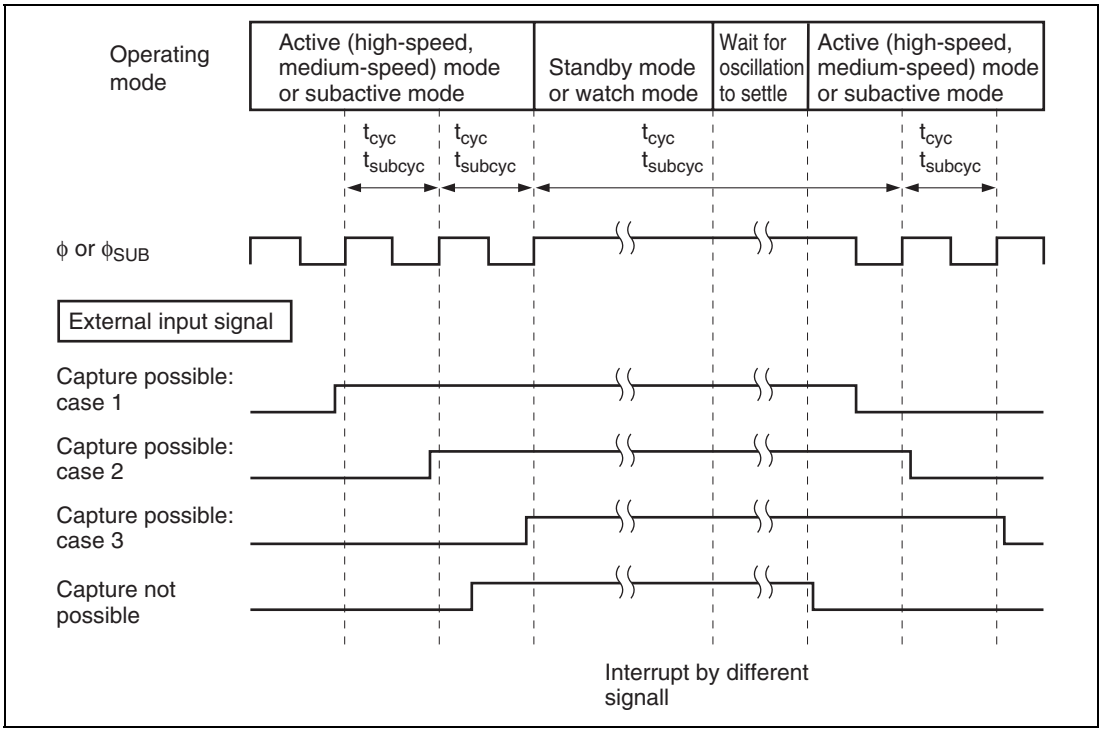



**Figure 5.3 External Input Signal Capture when Signal Changes before/after Standby Mode or Watch Mode**

4. Input pins to which these notes apply:
   $\overline{IRQ_4}$ to $\overline{IRQ_0}$, $\overline{WKP_7}$ to $\overline{WKP_0}$, $\overline{ADTRG}$, TMIC, TMIF, TMIG

## 5.4 Watch Mode

### 5.4.1 Transition to Watch Mode

The system goes from active or subactive mode to watch mode when a SLEEP instruction is executed while the SSBY bit in SYSCR1 is set to 1 and bit TMA3 in TMA is set to 1.

In watch mode, operation of on-chip peripheral modules is halted except for timer A, timer F, timer G, AEC and the LCD controller/driver (for which operation or halting can be set) is halted. As long as a minimum required voltage is applied, the contents of CPU registers, the on-chip RAM and some registers of the on-chip peripheral modules, are retained. I/O ports keep the same states as before the transition.

### 5.4.2 Clearing Watch Mode

Watch mode is cleared by an interrupt (timer A, timer F, timer G, $IRQ_0$, or $WKP_7$ to $WKP_0$) or by input at the $\overline{RES}$ pin.

- Clearing by interrupt
  When watch mode is cleared by interrupt, the mode to which a transition is made depends on the settings of LSON in SYSCR1 and MSON in SYSCR2. If both LSON and MSON are cleared to 0, transition is to active (high-speed) mode; if LSON = 0 and MSON = 1, transition is to active (medium-speed) mode; if LSON = 1, transition is to subactive mode. When the transition is to active mode, after the time set in SYSCR1 bits STS2 to STS0 has elapsed, a stable clock signal is supplied to the entire chip, watch mode is cleared, and interrupt exception handling starts. Watch mode is not cleared if the I bit of CCR is set to 1 or the particular interrupt is disabled in the interrupt enable register.

- Clearing by $\overline{RES}$ input
  Clearing by $\overline{RES}$ pin is the same as for standby mode; see Clearing by $\overline{RES}$ pin in section 5.3.2, Clearing Standby Mode.

### 5.4.3 Oscillator Settling Time after Watch Mode is Cleared

The waiting time is the same as for standby mode; see section 5.3.3, Oscillator Settling Time after Standby Mode is Cleared.

### 5.4.4 Notes on External Input Signal Changes before/after Watch Mode

See section 5.3.5, Notes on External Input Signal Changes before/after Standby Mode.

## 5.5 Subsleep Mode

### 5.5.1 Transition to Subsleep Mode

The system goes from subactive mode to subsleep mode when a SLEEP instruction is executed while the SSBY bit in SYSCR1 is cleared to 0, LSON bit in SYSCR1 is set to 1, and TMA3 bit in TMA is set to 1. In subsleep mode, operation of on-chip peripheral modules other than the A/D converter WDT and PWM is halted. As long as a minimum required voltage is applied, the contents of CPU registers, the on-chip RAM and some registers of the on-chip peripheral modules are retained. I/O ports keep the same states as before the transition.

### 5.5.2 Clearing Subsleep Mode

Subsleep mode is cleared by an interrupt (timer A, timer C, timer F, timer G, asynchronous counter, SCI3-2, SCI3-1, $IRQ_4$ to $IRQ_0$, $WKP_7$ to $WKP_0$) or by a low input at the $\overline{RES}$ pin.

- Clearing by interrupt

  When an interrupt is requested, subsleep mode is cleared and interrupt exception handling starts. Subsleep mode is not cleared if the I bit of CCR is set to 1 or the particular interrupt is disabled in the interrupt enable register.

  Interrupt signal and system clock are mutually asynchronous. Synchronization error time in a maximum is $2/\phi_{SUB}$ (s).

- Clearing by $\overline{RES}$ input

  Clearing by $\overline{RES}$ pin is the same as for standby mode; see Clearing by $\overline{RES}$ pin in section 5.3.2, Clearing Standby Mode.

## 5.6 Subactive Mode

### 5.6.1 Transition to Subactive Mode

Subactive mode is entered from watch mode if a timer A, timer F, timer G, $IRQ_0$, or $WKP_7$ to $WKP_0$ interrupt is requested while the LSON bit in SYSCR1 is set to 1. From subsleep mode, subactive mode is entered if a timer A, timer C, timer F, timer G, asynchronous counter, SCI3-1, SCI3-2, $IRQ_4$ to $IRQ_0$, or $WKP_7$ to $WKP_0$ interrupt is requested. A transition to subactive mode does not take place if the I bit of CCR is set to 1 or the particular interrupt is disabled in the interrupt enable register.

### 5.6.2 Clearing Subactive Mode

Subactive mode is cleared by a SLEEP instruction or by a low input at the $\overline{RES}$ pin.

- Clearing by SLEEP instruction

  If a SLEEP instruction is executed while the SSBY bit in SYSCR1 is set to 1 and TMA3 bit in TMA is set to 1, subactive mode is cleared and watch mode is entered. If a SLEEP instruction is executed while SSBY = 0 and LSON = 1 in SYSCR1 and TMA3 = 1 in TMA, subsleep mode is entered. Direct transfer to active mode is also possible; see section 5.8, Direct Transfer, below.

- Clearing by $\overline{RES}$ pin

  Clearing by $\overline{RES}$ pin is the same as for standby mode; see Clearing by $\overline{RES}$ pin in section 5.3.2, Clearing Standby Mode.

### 5.6.3 Operating Frequency in Subactive Mode

The operating frequency in subactive mode is set in bits SA1 and SA0 in SYSCR2. The choices are $\phi_w/2$, $\phi_w/4$, and $\phi_w/8$.

## 5.7 Active (Medium-Speed) Mode

### 5.7.1 Transition to Active (Medium-Speed) Mode

If the $\overline{\text{RES}}$ pin is driven low, active (medium-speed) mode is entered. If the LSON bit in SYSCR2 is set to 1 while the LSON bit in SYSCR1 is cleared to 0, a transition to active (medium-speed) mode results from $IRQ_0$, $IRQ_1$, or $WKP_7$ to $WKP_0$ interrupts in standby mode, timer A, timer F, timer G, $IRQ_0$, or $WKP_7$ to $WKP_0$ interrupts in watch mode, or any interrupt in sleep mode. A transition to active (medium-speed) mode does not take place if the I bit of CCR is set to 1 or the particular interrupt is disabled in the interrupt enable register.
Furthermore, it sometimes acts with half state early timing at the time of transition to active (medium-speed) mode.

### 5.7.2 Clearing Active (Medium-Speed) Mode

Active (medium-speed) mode is cleared by a SLEEP instruction.

- Clearing by SLEEP instruction

  A transition to standby mode takes place if the SLEEP instruction is executed while the SSBY bit in SYSCR1 is set to 1, the LSON bit in SYSCR1 is cleared to 0, and the TMA3 bit in TMA is cleared to 0. The system goes to watch mode if the SSBY bit in SYSCR1 is set to 1 and bit TMA3 in TMA is set to 1 when a SLEEP instruction is executed.

  When both SSBY and LSON are cleared to 0 in SYSCR1 and a SLEEP instruction is executed, sleep mode is entered. Direct transfer to active (high-speed) mode or to subactive mode is also possible. See section 5.8, Direct Transfer, below for details.

- Clearing by $\overline{\text{RES}}$ pin

  When the $\overline{\text{RES}}$ pin is driven low, a transition is made to the reset state and active (medium-speed) mode is cleared.

### 5.7.3 Operating Frequency in Active (Medium-Speed) Mode

Operation in active (medium-speed) mode is clocked at the frequency designated by the MA1 and MA0 bits in SYSCR1.

## 5.8 Direct Transfer

### 5.8.1 Overview of Direct Transfer

The CPU can execute programs in three modes: active (high-speed) mode, active (medium-speed) mode, and subactive mode. A direct transfer is a transition among these three modes without the stopping of program execution. A direct transfer can be made by executing a SLEEP instruction while the DTON bit in SYSCR2 is set to 1. After the mode transition, direct transfer interrupt exception handling starts.

If the direct transfer interrupt is disabled in interrupt enable register 2, a transition is made instead to sleep mode or watch mode. Note that if a direct transition is attempted while the I bit in CCR is set to 1, sleep mode or watch mode will be entered, and it will be impossible to clear the resulting mode by means of an interrupt.

- Direct transfer from active (high-speed) mode to active (medium-speed) mode
  When a SLEEP instruction is executed in active (high-speed) mode while the SSBY and LSON bits in SYSCR1 are cleared to 0, the MSON bit in SYSCR2 is set to 1, and the DTON bit in SYSCR2 is set to 1, a transition is made to active (medium-speed) mode via sleep mode.

- Direct transfer from active (medium-speed) mode to active (high-speed) mode
  When a SLEEP instruction is executed in active (medium-speed) mode while the SSBY and LSON bits in SYSCR1 are cleared to 0, the MSON bit in SYSCR2 is cleared to 0, and the DTON bit in SYSCR2 is set to 1, a transition is made to active (high-speed) mode via sleep mode.

- Direct transfer from active (high-speed) mode to subactive mode
  When a SLEEP instruction is executed in active (high-speed) mode while the SSBY and LSON bits in SYSCR1 are set to 1, the DTON bit in SYSCR2 is set to 1, and the TMA3 bit in TMA is set to 1, a transition is made to subactive mode via watch mode.

- Direct transfer from subactive mode to active (high-speed) mode
  When a SLEEP instruction is executed in subactive mode while the SSBY bit in SYSCR1 is set to 1, the LSON bit in SYSCR1 is cleared to 0, the MSON bit in SYSCR2 is cleared to 0, the DTON bit in SYSCR2 is set to 1, and the TMA3 bit in TMA is set to 1, a transition is made directly to active (high-speed) mode via watch mode after the waiting time set in SYSCR1 bits STS2 to STS0 has elapsed.

- Direct transfer from active (medium-speed) mode to subactive mode

  When a SLEEP instruction is executed in active (medium-speed) while the SSBY and LSON bits in SYSCR1 are set to 1, the DTON bit in SYSCR2 is set to 1, and the TMA3 bit in TMA is set to 1, a transition is made to subactive mode via watch mode.

- Direct transfer from subactive mode to active (medium-speed) mode

  When a SLEEP instruction is executed in subactive mode while the SSBY bit in SYSCR1 is set to 1, the LSON bit in SYSCR1 is cleared to 0, the MSON bit in SYSCR2 is set to 1, the DTON bit in SYSCR2 is set to 1, and the TMA3 bit in TMA is set to 1, a transition is made directly to active (medium-speed) mode via watch mode after the waiting time set in SYSCR1 bits STS2 to STS0 has elapsed.

### 5.8.2 Direct Transition Times

#### (1) Time for Direct Transition from Active (High-Speed) Mode to Active (Medium-Speed) Mode

A direct transition from active (high-speed) mode to active (medium-speed) mode is performed by executing a SLEEP instruction in active (high-speed) mode while bits SSBY and LSON are both cleared to 0 in SYSCR1, and bits MSON and DTON are both set to 1 in SYSCR2. The time from execution of the SLEEP instruction to the end of interrupt exception handling (the direct transition time) is given by equation (1) below.

Direct transition time = { (Number of SLEEP instruction execution states) + (number of internal processing states) } × (tcyc before transition) + (number of interrupt exception handling execution states) × (tcyc after transition) ..... (1)

Example: Direct transition time = (2 + 1) × 2tosc + 14 × 16tosc = 230tosc (when ϕ/8 is selected as the CPU operating clock)

Notation:
tosc: OSC clock cycle time
tcyc: System clock (ϕ) cycle time

#### (2) Time for Direct Transition from Active (Medium-Speed) Mode to Active (High-Speed) Mode

A direct transition from active (medium-speed) mode to active (high-speed) mode is performed by executing a SLEEP instruction in active (medium-speed) mode while bits SSBY and LSON are both cleared to 0 in SYSCR1, and bit MSON is cleared to 0 and bit DTON is set to 1 in SYSCR2.

The time from execution of the SLEEP instruction to the end of interrupt exception handling (the direct transition time) is given by equation (2) below.

Direct transition time = { (Number of SLEEP instruction execution states) + (number of internal processing states) } × (tcyc before transition) + (number of interrupt exception handling execution states) × (tcyc after transition) ..... (2)

Example: Direct transition time = (2 + 1) × 16tosc + 14 × 2tosc = 76tosc (when ϕ/8 is selected as the CPU operating clock)

Notation:
tosc: OSC clock cycle time
tcyc: System clock (ϕ) cycle time

**(3) Time for Direct Transition from Subactive Mode to Active (High-Speed) Mode**

A direct transition from subactive mode to active (high-speed) mode is performed by executing a SLEEP instruction in subactive mode while bit SSBY is set to 1 and bit LSON is cleared to 0 in SYSCR1, bit MSON is cleared to 0 and bit DTON is set to 1 in SYSCR2, and bit TMA3 is set to 1 in TMA. The time from execution of the SLEEP instruction to the end of interrupt exception handling (the direct transition time) is given by equation (3) below.

Direct transition time = { (Number of SLEEP instruction execution states) + (number of internal processing states) } × (tsubcyc before transition) + { (wait time set in STS2 to STS0) + (number of interrupt exception handling execution states) } × (tcyc after transition) ..... (3)

Example: Direct transition time = (2 + 1) × 8tw + (8192 + 14) × 2tosc = 24tw + 16412tosc (when ϕw/8 is selected as the CPU operating clock, and wait time = 8192 states)

Notation:
tosc: OSC clock cycle time
tw: Watch clock cycle time
tcyc: System clock (ϕ) cycle time
tsubcyc: Subclock ($\phi_{SUB}$) cycle time

### (4) Time for Direct Transition from Subactive Mode to Active (Medium-Speed) Mode

A direct transition from subactive mode to active (medium-speed) mode is performed by executing a SLEEP instruction in subactive mode while bit SSBY is set to 1 and bit LSON is cleared to 0 in SYSCR1, bits MSON and DTON are both set to 1 in SYSCR2, and bit TMA3 is set to 1 in TMA. The time from execution of the SLEEP instruction to the end of interrupt exception handling (the direct transition time) is given by equation (4) below.

Direct transition time = { (Number of SLEEP instruction execution states) + (number of internal processing states) } × (tsubcyc before transition) + { (wait time set in STS2 to STS0) + (number of interrupt exception handling execution states) } × (tcyc after transition) ..... (4)

Example: Direct transition time = (2 + 1) × 8tw + (8192 + 14) × 16tosc = 24tw + 131296tosc
(when ϕw/8 or ϕ8 is selected as the CPU operating clock, and wait time = 8192 states)

Notation:
tosc: OSC clock cycle time
tw: Watch clock cycle time
tcyc: System clock (ϕ) cycle time
tsubcyc: Subclock ($\phi_{SUB}$) cycle time

### 5.8.3 Notes on External Input Signal Changes before/after Direct Transition

1. Direct transition from active (high-speed) mode to subactive mode
   Since the mode transition is performed via watch mode, see section 5.3.5, Notes on External Input Signal Changes before/after Standby Mode.

2. Direct transition from active (medium-speed) mode to subactive mode
   Since the mode transition is performed via watch mode, see section 5.3.5, Notes on External Input Signal Changes before/after Standby Mode.

3. Direct transition from subactive mode to active (high-speed) mode
   Since the mode transition is performed via watch mode, see section 5.3.5, Notes on External Input Signal Changes before/after Standby Mode.

4. Direct transition from subactive mode to active (medium-speed) mode
   Since the mode transition is performed via watch mode, see section 5.3.5, Notes on External Input Signal Changes before/after Standby Mode.

## 5.9 Module Standby Mode

### 5.9.1 Setting Module Standby Mode

Module standby mode is set for individual peripheral functions. All the on-chip peripheral modules can be placed in module standby mode. When a module enters module standby mode, the system clock supply to the module is stopped and operation of the module halts. This state is identical to standby mode.

Module standby mode is set for a particular module by setting the corresponding bit to 0 in clock stop register 1 (CKSTPR1) or clock stop register 2 (CKSTPR2). (See table 5.5.)

### 5.9.2 Clearing Module Standby Mode

Module standby mode is cleared for a particular module by setting the corresponding bit to 1 in clock stop register 1 (CKSTPR1) or clock stop register 2 (CKSTPR2). (See table 5.5.)

Following a reset, clock stop register 1 (CKSTPR1) and clock stop register 2 (CKSTPR2) are both initialized to H'FF.

**Table 5.5 Setting and Clearing Module Standby Mode by Clock Stop Register**

| Register Name | Bit Name | | Operation |
|---|---|---|---|
| CKSTPR1 | TACKSTP | 1 | Timer A module standby mode is cleared |
| | | 0 | Timer A is set to module standby mode |
| | TCCKSTP | 1 | Timer C module standby mode is cleared |
| | | 0 | Timer C is set to module standby mode |
| | TFCKSTP | 1 | Timer F module standby mode is cleared |
| | | 0 | Timer F is set to module standby mode |
| | TGCKSTP | 1 | Timer G module standby mode is cleared |
| | | 0 | Timer G is set to module standby mode |
| | ADCKSTP | 1 | A/D converter module standby mode is cleared |
| | | 0 | A/D converter is set to module standby mode |
| | S32CKSTP | 1 | SCI3-2 module standby mode is cleared |
| | | 0 | SCI3-2 is set to module standby mode |
| | S31CKSTP | 1 | SCI3-1 module standby mode is cleared |
| | | 0 | SCI3-1 is set to module standby mode |
| CKSTPR2 | LDCKSTP | 1 | LCD module standby mode is cleared |
| | | 0 | LCD is set to module standby mode |
| | PWCKSTP | 1 | PWM module standby mode is cleared |
| | | 0 | PWM is set to module standby mode |
| | WDCKSTP | 1 | Watchdog timer module standby mode is cleared |
| | | 0 | Watchdog timer is set to module standby mode |
| | AECKSTP | 1 | Asynchronous event counter module standby mode is cleared |
| | | 0 | Asynchronous event counter is set to module standby mode |

Note: For details of module operation, see the sections on the individual modules.

### 5.9.3 Usage Note

If, due to the timing with which a peripheral module issues interrupt requests, the module in question is set to module standby mode before an interrupt is processed, the module will stop with the interrupt request still pending. In this situation, interrupt processing will be repeated indefinitely unless interrupts are prohibited.

It is therefore necessary to ensure that no interrupts are generated when a module is set to module standby mode. The surest way to do this is to specify the module standby mode setting only when interrupts are prohibited (interrupts prohibited using the interrupt enable register or interrupts masked using bit CCR-I).

# Section 6 ROM

## 6.1 Overview

The H8/38532 has 16 Kbytes of on-chip mask ROM, the H8/38533 has 24 Kbytes, the H8/38534 has 32 Kbytes, the H8/38535 has 40 Kbytes, the H8/38536 has 48 Kbytes, and the H8/38537 has 60 Kbytes. The ROM is connected to the CPU by a 16-bit data bus, allowing high-speed two-state access for both byte data and word data.

The flash memory version of the H8/38537 is equipped with 60 Kbytes of flash memory. The flash memory version of the H8/38534 is equipped with 32 Kbytes of flash memory.

## 6.2 Flash Memory Overview

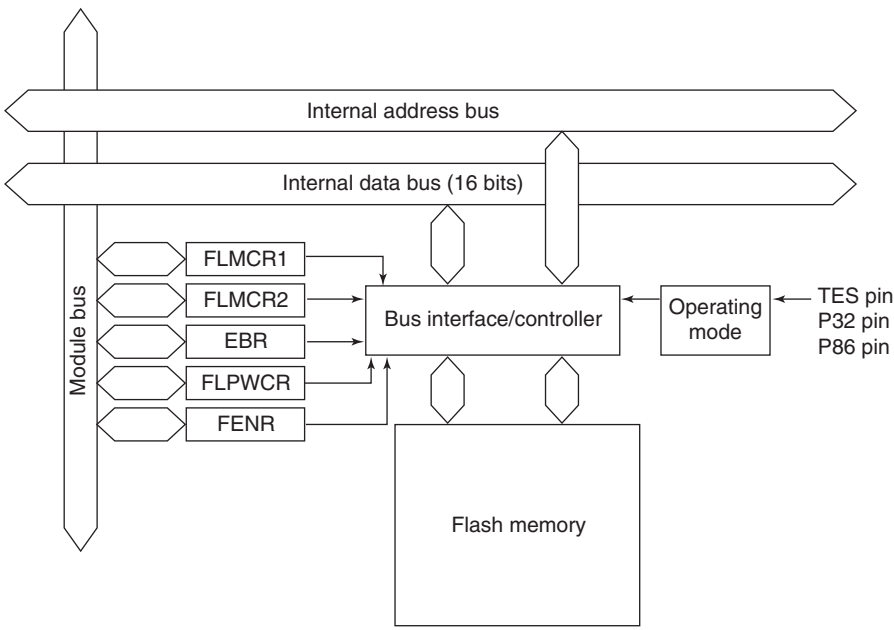
### 6.2.1 Features

The features of the 60 Kbytes or 32 Kbytes of flash memory built into the flash memory versions are summarized below.

- Programming/erase methods
  - The 60-Kbyte flash memory is programmed 128 bytes at a time. Erase is performed in single-block units. The flash memory is configured as follows: 1 Kbyte × 4 blocks, 28 Kbytes × 1 block, 16 Kbytes × 1 block, 8 Kbytes × 1 block, 4 Kbytes × 1 block. The 32-Kbyte flash memory is configured as follows: 1 Kbyte × 4 blocks, 28 Kbytes × 1 block. To erase the entire flash memory, each block must be erased in turn.
- Reprogramming capability
  - The flash memory can be reprogrammed up to 1,000 times.
- On-board programming
  - On-board programming/erasing can be done in boot mode, in which the boot program built into the chip is started to erase or program of the entire flash memory. In normal user program mode, individual blocks can be erased or programmed.
- Programmer mode
  - Flash memory can be programmed/erased in programmer mode using a PROM programmer, as well as in on-board programming mode.
- Automatic bit rate adjustment
  - For data transfer in boot mode, this LSI's bit rate can be automatically adjusted to match the transfer bit rate of the host.

- Programming/erasing protection
  — Sets software protection against flash memory programming/erasing.
- Power-down mode
  — The power supply circuit is partly halted in the subactive mode and can be read in the power-down mode.

### 6.2.2 Block Diagram




Legend:
FLMCR1: Flash memory control register 1
FLMCR2: Flash memory control register 2
EBR: Erase block register
FLPWCR: Flash memory power control register
FENR: Flash memory enable register

**Figure 6.1 Block Diagram of Flash Memory**

### 6.2.3 Block Configuration

Figure 6.2 shows the block configuration of flash memory. The thick lines indicate erasing units, the narrow lines indicate programming units, and the values are addresses. The flash memory is divided into 1 Kbyte × 4 blocks, 28 Kbytes × 1 block, 16 Kbytes × 1 block, 8 Kbytes × 1 block, and 4 Kbytes × 1 block. Erasing is performed in these units. Programming is performed in 128-byte units starting from an address with lower eight bits H'00 or H'80.

| | | | | | |
|---|---|---|---|---|---|
| Erase unit 1 Kbyte | H'0000 | H'0001 | H'0002 | ← Programming unit: 128 bytes → | H'007F |
| | H'0080 | H'0081 | H'0082 | | H'00FF |
| | | | | | |
| | H'0380 | H'0381 | H'0382 | | H'03FF |
| Erase unit 1 Kbyte | H'0400 | H'0401 | H'0402 | ← Programming unit: 128 bytes → | H'047F |
| | H'0480 | H'0481 | H'0482 | | H'04FF |
| | | | | | |
| | H'0780 | H'0781 | H'0782 | | H'07FF |
| Erase unit 1 Kbyte | H'0800 | H'0801 | H'0802 | ← Programming unit: 128 bytes → | H'087F |
| | H'0880 | H'0881 | H'0882 | | H'080F |
| | | | | | |
| | H'0B80 | H'0B81 | H'0B82 | | H'0BFF |
| Erase unit 1 Kbyte | H'0C00 | H'0C01 | H'0C02 | ← Programming unit: 128 bytes → | H'0C7F |
| | H'0C80 | H'0C81 | H'0C82 | | H'0CFF |
| | | | | | |
| | H'0F80 | H'0F81 | H'0F82 | | H'0FFF |
| Erase unit 28 Kbytes | H'1000 | H'1001 | H'1002 | ← Programming unit: 128 bytes → | H'107F |
| | H'1080 | H'1081 | H'1082 | | H'10FF |
| | | | | | |
| | H'7F80 | H'7F81 | H'7F82 | | H'7FFF |
| Erase unit 16 Kbytes | H'8000 | H'8001 | H'8002 | ← Programming unit: 128 bytes → | H'807F |
| | H'8080 | H'8081 | H'8082 | | H'8CFF |
| | | | | | |
| | H'BF80 | H'BF81 | H'BF82 | | H'BFFF |
| Erase unit 8 Kbytes | H'C000 | H'C001 | H'C002 | ← Programming unit: 128 bytes → | H'C07F |
| | H'C080 | H'C081 | H'C082 | | H'CCFF |
| | | | | | |
| | H'DF80 | H'DF81 | H'DF82 | | H'DFFF |
| Erase unit 4 Kbytes | H'E000 | H'E001 | H'E002 | ← Programming unit: 128 bytes → | H'E07F |
| | H'E080 | H'E081 | H'E082 | | H'ECFF |
| | | | | | |
| | H'EF80 | H'EF81 | H'EF82 | | H'EFFF |

**Figure 6.2 Flash Memory Block Configuration**

### 6.2.4 Register Configuration

Table 6.1 lists the register configuration to control the flash memory when the built in flash memory is effective.

**Table 6.1 Register Configuration**

| Register Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| Flash memory control register 1 | FLMCR1 | R/W | H'00 | H'F020 |
| Flash memory control register 2 | FLMCR2 | R | H'00 | H'F021 |
| Flash memory power control register | FLPWCR | R/W | H'00 | H'F022 |
| Erase block register | EBR | R/W | H'00 | H'F023 |
| Flash memory enable register | FENR | R/W | H'00 | H'F02B |

Note: FLMCR1, FLMCR2, FLPWCR, EBR, and FENR are 8 bit registers. Only byte access is enabled which are two-state access. These registers are dedicated to the product in which flash memory is included. The product in which mask ROM is included does not have these registers. When the corresponding address is read in these products, the value is undefined. A write is disabled.

## 6.3 Descriptions of Registers of the Flash Memory

### 6.3.1 Flash Memory Control Register 1 (FLMCR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SWE | ESU | PSU | EV | PV | E | P |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

FLMCR1 is a register that makes the flash memory change to program mode, program-verify mode, erase mode, or erase-verify mode. For details on register setting, refer to section 6.5, Flash Memory Programming/Erasing. By setting this register, the flash memory enters program mode, erase mode, program-verify mode, or erase-verify mode. Read the data in the state that bits 6 to 0 of this register are cleared when using flash memory as normal built-in ROM.

**Bit 7—Reserved**

This bit is always read as 0 and cannot be modified.

**Bit 6—Software Write Enable (SWE)**

This bit is to set enabling/disabling of programming/enabling of flash memory (set when bits 5 to 0 and the EBR register are to be set).

| Bit 6<br>SWE | Description |
|---|---|
| 0 | Programming/erasing is disabled. Other FLMCR1 register bits and all EBR bits cannot be set. (initial value) |
| 1 | Flash memory programming/erasing is enabled. |

### Bit 5—Erase Setup (ESU)

This bit is to prepare for changing to erase mode. Set this bit to 1 before setting the E bit to 1 in FLMCR1 (do not set SWE, PSU, EV, PV, E, and P bits at the same time).

| Bit 5<br>ESU | Description | |
|---|---|---|
| 0 | The erase setup state is cancelled | (initial value) |
| 1 | The flash memory changes to the erase setup state. Set this bit to 1 before setting the E bit to 1 in FLMCR1. | |

### Bit 4—Program Setup (PSU)

This bit is to prepare for changing to program mode. Set this bit to 1 before setting the P bit to 1 in FLMCR1 (do not set SWE, ESU, EV, PV, E, and P bits at the same time).

| Bit 4<br>PSU | Description | |
|---|---|---|
| 0 | The program setup state is cancelled | (initial value) |
| 1 | The flash memory changes to the program setup state. Set this bit to 1 before setting the P bit to 1 in FLMCR1. | |

### Bit 3—Erase-Verify (EV)

This bit is to set changing to or cancelling erase-verify mode (do not set SWE, ESU, PSU, PV, E, and P bits at the same time).

| Bit 3<br>EV | Description | |
|---|---|---|
| 0 | Erase-verify mode is cancelled | (initial value) |
| 1 | The flash memory changes to erase-verify mode | |

### Bit 2—Program-Verify (PV)

This bit is to set changing to or cancelling program-verify mode (do not set SWE, ESU, PSU, EV, E, and P bits at the same time).

| Bit 2<br>PV | Description | |
|---|---|---|
| 0 | Program-verify mode is cancelled | (initial value) |
| 1 | The flash memory changes to program-verify mode | |

### Bit 1—Erase (E)

This bit is to set changing to or cancelling erase mode (do not set SWE, ESU, PSU, EV, PV, and P bits at the same time).

| Bit 1<br>E | Description | |
|---|---|---|
| 0 | Erase mode is cancelled | (initial value) |
| 1 | When this bit is set to 1, while the SWE = 1 and ESU = 1, the flash memory changes to erase mode. | |

### Bit 0—Program (P)

This bit is to set changing to or cancelling program mode (do not set SWE, ESU, PSU, EV, PV, and E bits at the same time).

| Bit 0<br>P | Description | |
|---|---|---|
| 0 | Program mode is cancelled | (initial value) |
| 1 | When this bit is set to 1, while the SWE = 1 and PSU = 1, the flash memory changes to program mode. | |

### 6.3.2 Flash Memory Control Register 2 (FLMCR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FLER | — | — | — | — | — | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | — | — | — | — | — | — | — |

FLMCR2 is a register that displays the state of flash memory programming/erasing. FLMCR2 is a read-only register, and should not be written to.

**Bit 7—Flash Memory Error (FLER)**

This bit is set when the flash memory detects an error and goes to the error-protection state during programming or erasing to the flash memory. See section 6.6.3, Error Protection, for details.

| Bit 7<br>FLER | Description |
|---|---|
| 0 | The flash memory operates normally. (initial value) |
| 1 | Indicates that an error has occurred during an operation on flash memory (programming or erasing). |

**Bits 6 to 0—Reserved**

These bits are always read as 0 and cannot be modified.

### 6.3.3 Erase Block Register (EBR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EB7 | EB6 | EB5 | EB4 | EB3 | EB2 | EB1 | EB0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

EBR specifies the flash memory erase area block. EBR is initialized to H'00 when the SWE bit in FLMCR1 is 0. Do not set more than one bit at a time, as this will cause all the bits in EBR to be automatically cleared to 0. When each bit is set to 1 in EBR, the corresponding block can be erased. Other blocks change to the erase-protection state. See table 6.2 for the method of dividing blocks of the flash memory. When the whole bits are to be erased, erase them in turn in unit of a block.

**Table 6.2 Division of Blocks to Be Erased**

| EBR | Bit Name | Block (Size) | Address |
|---|---|---|---|
| 0 | EB0 | EB0 (1 Kbyte) | H'0000 to H'03FF |
| 1 | EB1 | EB1 (1 Kbyte) | H'0400 to H'07FF |
| 2 | EB2 | EB2 (1 Kbyte) | H'0800 to H'0BFF |
| 3 | EB3 | EB3 (1 Kbyte) | H'0C00 to H'0FFF |
| 4 | EB4 | EB4 (28 Kbytes) | H'1000 to H'7FFF |
| 5 | EB5 | EB5 (16 Kbytes) | H'8000 to H'BFFF |
| 6 | EB6 | EB6 (8 Kbytes) | H'C000 to H'DFFF |
| 7 | EB7 | EB7 (4 Kbytes) | H'E000 to H'EFFF |

### 6.3.4 Flash Memory Power Control Register (FLPWCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PDWND | — | — | — | — | — | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | — | — | — | — | — | — | — |

FLPWCR enables or disables a transition to the flash memory power-down mode when the LSI switches to subactive mode. The power supply circuit can be read in the subactive mode, although it is partly halted in the power-down mode.

**Bit 7—Power-down Disable (PDWND)**

This bit selects the power-down mode of the flash memory when a transition to the subactive mode is made.

| Bit 7<br>PDWND | Description |
|---|---|
| 0 | When this bit is 0 and a transition is made to the subactive mode, the flash memory enters the power-down mode. (initial value) |
| 1 | When this bit is 1, the flash memory remains in the normal mode even after a transition is made to the subactive mode. |

**Bits 6 to 0—Reserved**

These bits are always read as 0 and cannot be modified.

### 6.3.5 Flash Memory Enable Register (FENR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FLSHE | — | — | — | — | — | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | — | — | — | — | — | — | — |

FENR controls CPU access to the flash memory control registers, FLMCR1, FLMCR2, EBR, and FLPWCR.

**Bit 7—Flash Memory Control Register Enable (FLSHE)**

This bit controls access to the flash memory control registers.

| Bit 7<br>FLSHE | Description | |
|---|---|---|
| 0 | Flash memory control registers cannot be accessed | (initial value) |
| 1 | Flash memory control registers can be accessed | |

**Bits 6 to 0—Reserved**

These bits are always read as 0 and cannot be modified.

## 6.4 On-Board Programming Modes

There are two modes for programming/erasing of the flash memory; boot mode, which enables on-board programming/erasing, and programmer mode, in which programming/erasing is performed with a PROM programmer. On-board programming/erasing can also be performed in user program mode. At reset-start in reset mode, this LSI changes to a mode depending on the TEST pin settings, P32 pin settings, and input level of each port, as shown in table 6.3. The input level of each pin must be defined four states before the reset ends.

When changing to boot mode, the boot program built into this LSI is initiated. The boot program transfers the programming control program from the externally-connected host to on-chip RAM via SCI32. After erasing the entire flash memory, the programming control program is executed. This can be used for programming initial values in the on-board state or for a forcible return when programming/erasing can no longer be done in user program mode. In user program mode, individual blocks can be erased and programmed by branching to the user program/erase control program prepared by the user.

**Table 6.3 Setting Programming Modes**

| TEST | P32 | P86 | PB0 | PB1 | PB2 | LSI State after Reset End |
|---|---|---|---|---|---|---|
| 0 | 1 | X | X | X | X | User Mode |
| 0 | 0 | 1 | X | X | X | Boot Mode |
| 1 | X | X | 0 | 0 | 0 | Programmer Mode |

X: Don’t care

### 6.4.1 Boot Mode

Table 6.4 shows the boot mode operations between reset end and branching to the programming control program. The device uses SCI32 in the boot mode.

1. When boot mode is used, the flash memory programming control program must be prepared in the host beforehand. Prepare a programming control program in accordance with the description in section 6.5, Flash Memory Programming/Erasing.
2. SCI3 should be set to asynchronous mode, and the transfer format as follows: 8-bit data, 1 stop bit, and no parity. The inversion function of TXD and RXD pins by the SPCR register is set to “Not to be inverted,” so do not put the circuit for inverting a value between the host and this LSI.

3. When the boot program is initiated, the chip measures the low-level period of asynchronous SCI communication data (H'00) transmitted continuously from the host. The chip then calculates the bit rate of transmission from the host, and adjusts the SCI3 bit rate to match that of the host. The reset should end with the RXD pin high. The RXD and TXD pins should be pulled up on the board if necessary. After the reset is complete, it takes approximately 100 states before the chip is ready to measure the low-level period.
4. After matching the bit rates, the chip transmits one H'00 byte to the host to indicate the completion of bit rate adjustment. The host should confirm that this adjustment end indication (H'00) has been received normally, and transmit one H'55 byte to the chip. If reception could not be performed normally, initiate boot mode again by a reset. Depending on the host's transfer bit rate and system clock frequency of this LSI, there will be a discrepancy between the bit rates of the host and the chip. To operate the SCI properly, set the host's transfer bit rate and system clock frequency of this LSI within the ranges listed in table 6.5.
5. In boot mode, a part of the on-chip RAM area is used by the boot program. The area H'F780 to H'FEEF is the area to which the programming control program is transferred from the host. The boot program area cannot be used until the execution state in boot mode switches to the programming control program.
6. Before branching to the programming control program, the chip terminates transfer operations by SCI3 (by clearing the RE and TE bits in SCR to 0), however the adjusted bit rate value remains set in BRR. Therefore, the programming control program can still use it for transfer of write data or verify data with the host. The TXD pin is high (PCR42 = 1, P42 = 1). The contents of the CPU general registers are undefined immediately after branching to the programming control program. These registers must be initialized at the beginning of the programming control program, as the stack pointer (SP), in particular, is used implicitly in subroutine calls, etc.
7. Boot mode can be cleared by a reset. End the reset after driving the reset pin low, waiting at least 20 states, and then setting the TEST pin and P32 pin. Boot mode is also cleared when a WDT overflow occurs.
8. Do not change the TEST pin and P32 pin input levels in boot mode.

## Table 6.4 Boot Mode Operation

| Item | Host Operation<br>Processing Contents | LSI Operation<br>Processing Contents |
|---|---|---|
| | | Branches to boot program at reset-start. |
| Bit rate adjustment | Continuously transmits data H'00 at specified bit rate. | • Measures low-level period of receive data H'00.<br>• Calculates bit rate and sets it in BRR of SCI3.<br>• Transmits data H'00 to the host to indicate that the adjustment has ended. |
| Flash memory erase | Transmits data H'55 when data H'00 is received and no error occurs. | Checks flash memory data, erases all flash memory blocks in case of written data existing, and transmits data H'AA to host. (If erase could not be done, transmits data H'FF to host and aborts operation.) |
| Transfer of programming control program | Transmits number of bytes (N) of programming control program to be transferred as 2-byte data (low-order byte following high-order byte) | Echobacks the 2-byte received data to host. |
| Transfer of programming control program (repeated for N times) | Transmits 1-byte of programming control program | Echobacks received data to host and also transfers it to RAM. |
| Execution of Programming control program | | Transmits 1-byte data H'AA to host.<br>Branches to programming control program transferred to on-chip RAM and starts execution. |

## Table 6.5 Oscillating Frequencies ($f_{osc}$) for which Automatic Adjustment of LSI Bit Rate Is Possible

| Product Group | Host Bit Rate | Oscillating Frequencies ($f_{osc}$) Range of LSI |
|---|---|---|
| H8/38537<br>H8/38534<br>Flash memory version | 19,200 bps | 16 MHz |
| | 9,600 bps | 8 to 16 MHz |
| | 4,800 bps | 6 to 16 MHz |
| | 2,400 bps | 2 to 16 MHz |
| | 1,200 bps | 2 to 16 MHz |

### 6.4.2 Programming/Erasing in User Program Mode

The term user mode refers to the status when a user program is being executed. On-board programming/erasing of an individual flash memory block can also be performed in user program mode by branching to a user program/erase control program. The user must set branching conditions and provide on-board means of supplying programming data. The flash memory must contain the user program/erase control program or a program that provides the user program/erase control program from external memory. As the flash memory itself cannot be read during programming/erasing, transfer the user program/erase control program to on-chip RAM, as in boot mode. Figure 6.3 shows a sample procedure for programming/erasing in user program mode. Prepare a user program/erase control program in accordance with the description in section 6.5, Flash Memory Programming/Erasing.

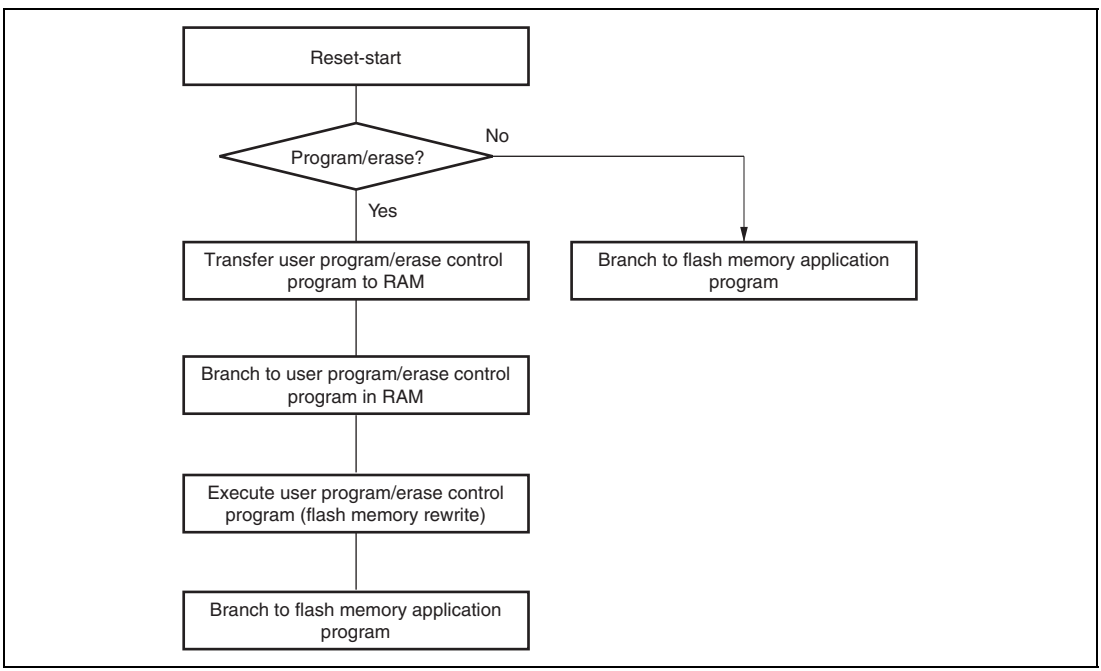



**Figure 6.3 Programming/Erasing Flowchart Example in User Program Mode**

## 6.5 Flash Memory Programming/Erasing

A software method using the CPU is employed to program and erase flash memory in the on-board programming modes. Depending on the FLMCR1 setting, the flash memory operates in one of the following four modes: Program mode, program-verify mode, erase mode, and erase-verify mode. The programming control program in boot mode and the user program/erase control program in user program mode use these operating modes in combination to perform programming/erasing. Flash memory programming and erasing should be performed in accordance with the descriptions in section 6.5.1, Program/Program-Verify and section 6.5.2, Erase/Erase-Verify, respectively.

### 6.5.1 Program/Program-Verify

When writing data or programs to the flash memory, the program/program-verify flowchart shown in figure 6.4 should be followed. Performing programming operations according to this flowchart will enable data or programs to be written to the flash memory without subjecting the chip to voltage stress or sacrificing program data reliability.

1. Programming must be done to an empty address. Do not reprogram an address to which programming has already been performed.
2. Programming should be carried out 128 bytes at a time. A 128-byte data transfer must be performed even if writing fewer than 128 bytes. In this case, H'FF data must be written to the extra addresses.
3. Prepare the following data storage areas in RAM: A 128-byte programming data area, a 128-byte reprogramming data area, and a 128-byte additional-programming data area. Perform reprogramming data computation according to table 6.6, and additional programming data computation according to table 6.7.
4. Consecutively transfer 128 bytes of data in byte units from the reprogramming data area or additional-programming data area to the flash memory. The program address and 128-byte data are latched in the flash memory. The lower 8 bits of the start address in the flash memory destination area must be H'00 or H'80.
5. The time during which the P bit is set to 1 is the programming time. Figure 6.8 shows the allowable programming times.
6. The watchdog timer (WDT) is set to prevent overprogramming due to program runaway, etc. An overflow cycle of approximately 6.6 ms is allowed.
7. For a dummy write to a verify address, write 1-byte data H'FF to an address whose lower 1 bit is b'0. Verify data can be read in word size from the address to which a dummy write was performed.
8. The maximum number of repetitions of the program/program-verify sequence of the same bit is 1,000.

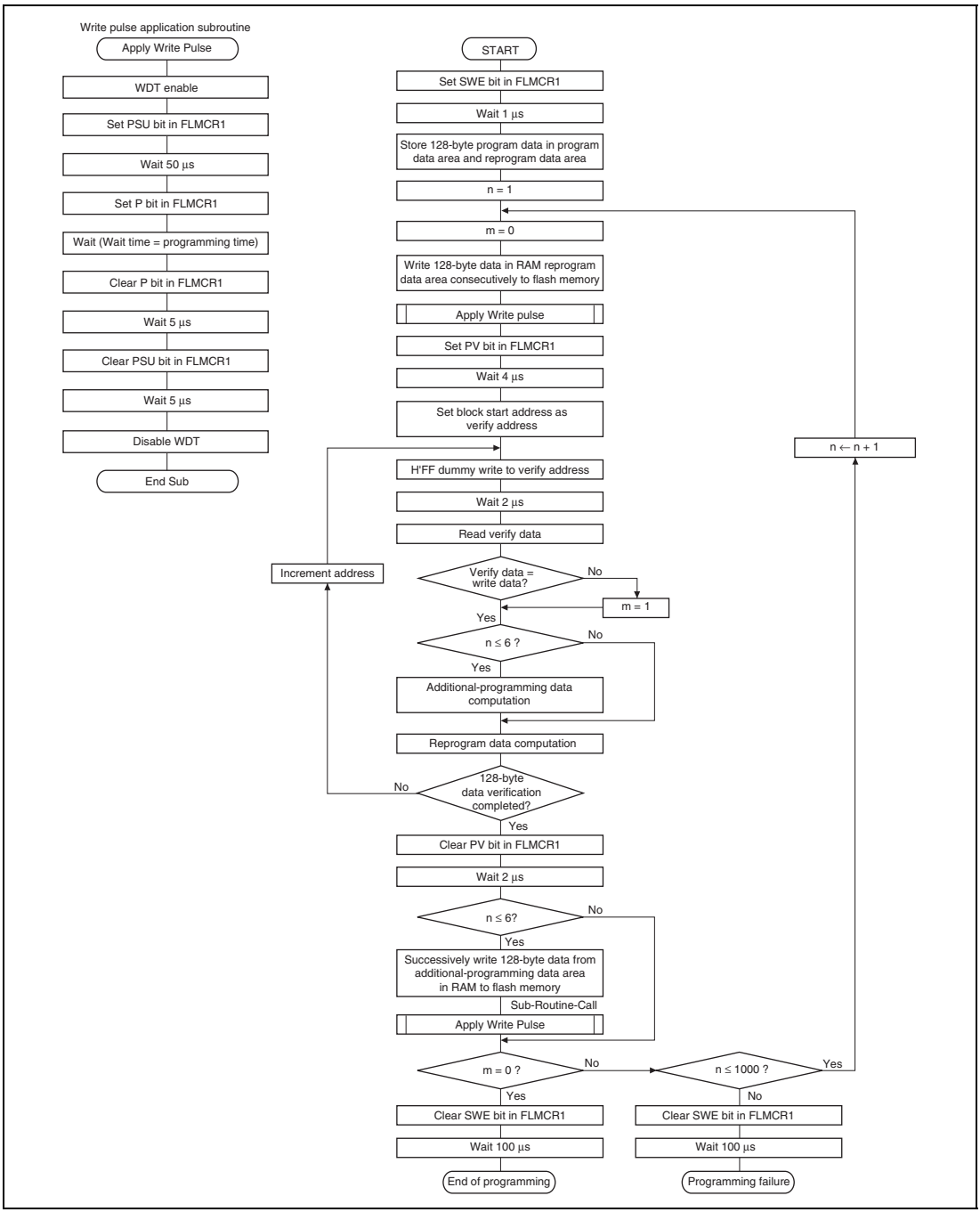



**Figure 6.4 Program/Program-Verify Flowchart**

**Table 6.6 Reprogram Data Computation Table**

| Program Data | Verify Data | Reprogram Data | Comments |
|---|---|---|---|
| 0 | 0 | 1 | Programming completed |
| 0 | 1 | 0 | Reprogram bit |
| 1 | 0 | 1 | — |
| 1 | 1 | 1 | Remains in erased state |

**Table 6.7 Additional-Program Data Computation Table**

| Reprogram Data | Verify Data | Additional-Program Data | Comments |
|---|---|---|---|
| 0 | 0 | 0 | Additional-program bit |
| 0 | 1 | 1 | No additional programming |
| 1 | 0 | 1 | No additional programming |
| 1 | 1 | 1 | No additional programming |

**Table 6.8 Programming Time**

| n (Number of Writes) | Programming Time | In Additional Programming | Comments |
|---|---|---|---|
| 1 to 6 | 30 | 10 | |
| 7 to 1,000 | 200 | — | |

Note: Time shown in μs.

### 6.5.2 Erase/Erase-Verify

When erasing flash memory, the erase/erase-verify flowchart shown in figure 6.5 should be followed.

1. Prewriting (setting erase block data to all 0s) is not necessary.
2. Erasing is performed in block units. Make only a single-bit specification in the erase block register (EBR). To erase multiple blocks, each block must be erased in turn.
3. The time during which the E bit is set to 1 is the flash memory erase time.
4. The watchdog timer (WDT) is set to prevent overerasing due to program runaway, etc. An overflow cycle of approximately 19.8 ms is allowed.
5. For a dummy write to a verify address, write 1-byte data H'FF to an address whose lower 1 bit is b'0. Verify data can be read in word size from the address to which a dummy write was performed.
6. If the read data is not erased successfully, set erase mode again, and repeat the erase/erase-verify sequence as before. The maximum number of repetitions of the erase/erase-verify sequence is 100.

### 6.5.3 Interrupt Handling when Programming/Erasing Flash Memory

All interrupts, are disabled while flash memory is being programmed or erased, or while the boot program is executing, for the following three reasons:

1. Interrupt during programming/erasing may cause a violation of the programming or erasing algorithm, with the result that normal operation cannot be assured.
2. If interrupt exception handling starts before the vector address is written or during programming/erasing, a correct vector cannot be fetched and the CPU malfunctions.
3. If an interrupt occurs during boot program execution, normal boot mode sequence cannot be carried out.

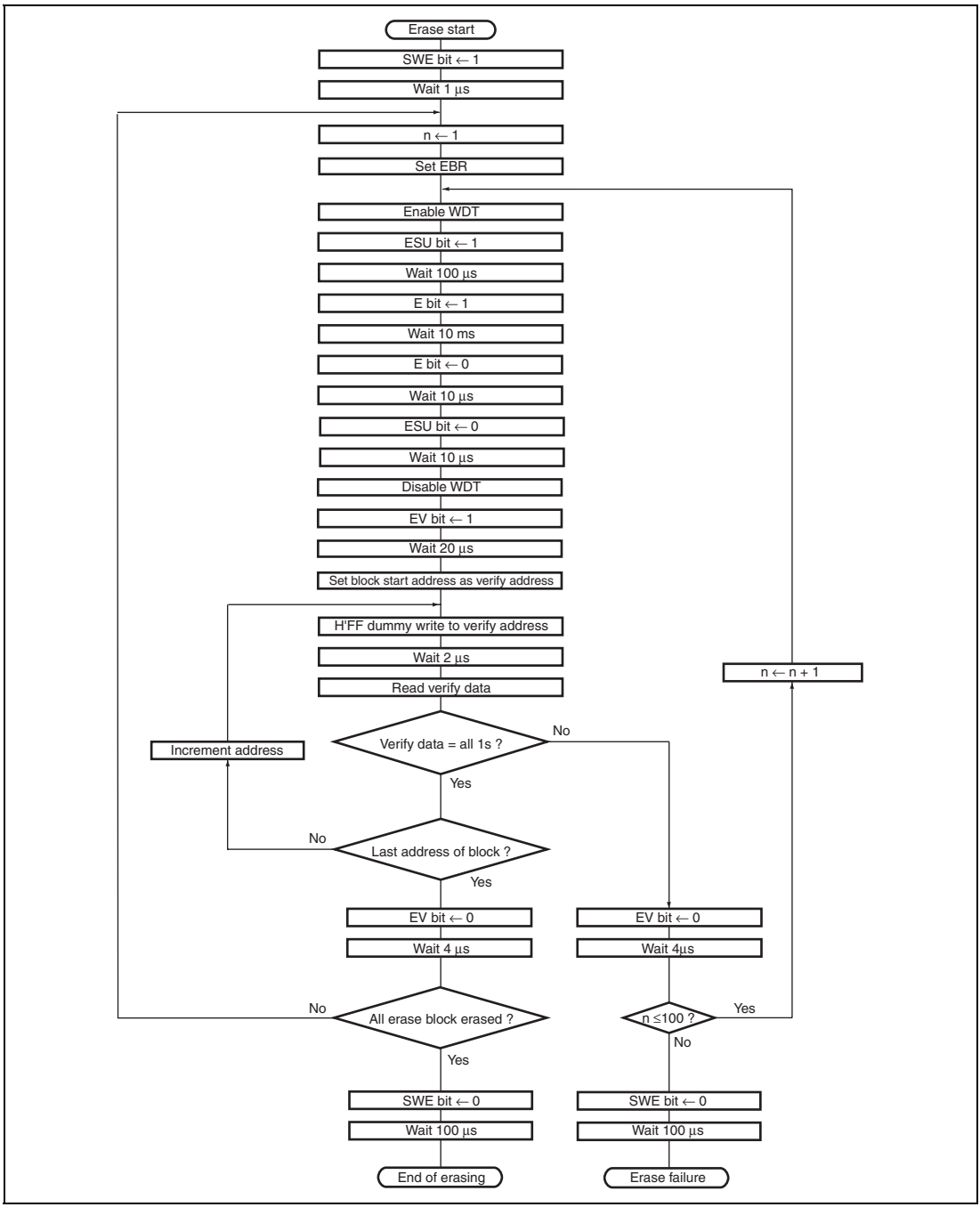



**Figure 6.5 Erase/Erase-Verify Flowchart**

## 6.6 Program/Erase Protection

There are three kinds of flash memory program/erase protection; hardware protection, software protection, and error protection.

### 6.6.1 Hardware Protection

Hardware protection refers to a state in which programming/erasing of flash memory is forcibly disabled or aborted because of a transition to reset, subactive mode, subsleep mode, watch mode, or standby mode. Flash memory control register 1 (FLMCR1), flash memory control register 2 (FLMCR2), and erase block register (EBR) are initialized. In a reset via the $\overline{\text{RES}}$ pin, the reset state is not entered unless the $\overline{\text{RES}}$ pin is held low until oscillation stabilizes after powering on. In the case of a reset during operation, hold the $\overline{\text{RES}}$ pin low for the $\overline{\text{RES}}$ pulse width specified in the AC Characteristics section.

### 6.6.2 Software Protection

Software protection can be implemented against programming/erasing of all flash memory blocks by clearing the SWE bit in FLMCR1. When software protection is in effect, setting the P or E bit in FLMCR1 does not cause a transition to program mode or erase mode. By setting the erase block register (EBR), erase protection can be set for individual blocks. When EBR is set to H'00, erase protection is set for all blocks.

### 6.6.3 Error Protection

In error protection, an error is detected when CPU runaway occurs during flash memory programming/erasing, or operation is not performed in accordance with the program/erase algorithm, and the program/erase operation is aborted. Aborting the program/erase operation prevents damage to the flash memory due to overprogramming or overerasing.

When the following errors are detected during programming/erasing of flash memory, the FLER bit in FLMCR2 is set to 1, and the error protection state is entered.

- When the flash memory of the relevant address area is read during programming/erasing (including vector read and instruction fetch)
- Immediately after exception handling excluding a reset during programming/erasing
- When a SLEEP instruction is executed during programming/erasing

The FLMCR1, FLMCR2, and EBR settings are retained, however program mode or erase mode is aborted at the point at which the error occurred. Program mode or erase mode cannot be re-entered by re-setting the P or E bit. However, PV and EV bit setting is enabled, and a transition can be made to verify mode. Error protection can be cleared only by a power-on reset.

## 6.7 Programmer Mode

In programmer mode, a PROM programmer can be used to perform programming/erasing via a socket adapter, just as a discrete flash memory. Use a PROM programmer that supports the MCU device type with the on-chip Renesas Technology (former Hitachi Ltd.) 64-Kbyte flash memory (F-ZTAT64V3). A 10-MHz input clock is required. For the conditions for transition to programmer mode, see table 6.3.

### 6.7.1 Socket Adapter

The socket adapter converts the pin allocation of the flash memory device to that of the discrete flash memory HN28F101. The address of the on-chip flash memory is H'0000 to H'EFFF. Figure 6.6 shows a socket-adapter-pin correspondence diagram.

### 6.7.2 Programmer Mode Commands

The following commands are supported in programmer mode.

- Memory Read Mode
- Auto-Program Mode
- Auto-Erase Mode
- Status Read Mode

Status polling is used for auto-programming, auto-erasing, and status read modes. In status read mode, detailed internal information is output after the execution of auto-programming or auto-erasing. Table 6.9 shows the sequence of each command. In auto-programming mode, 129 cycles are required since 128 bytes are written at the same time. In memory read mode, the number of cycles depends on the number of address write cycles (n).

**Table 6.9 Command Sequence in Programmer Mode**

| Command Name | Number of Cycles | 1st Cycle | | | 2nd Cycle | | |
|---|---|---|---|---|---|---|---|
| | | Mode | Address | Data | Mode | Address | Data |
| Memory read | 1 + n | Write | X | H'00 | Read | RA | Dout |
| Auto-program | 129 | Write | X | H'40 | Write | WA | Din |
| Auto-erase | 2 | Write | X | H'20 | Write | X | H'20 |
| Status read | 2 | Write | X | H'71 | Write | X | H'71 |

n: the number of address write cycles

| Flash Memory Device | | Socket Adapter (Conversion to 32-Pin Arrangement) | HN28F101 (32 Pins) | |
|---|---|---|---|---|
| Pin No. FP-80A TFP-80C | Pin Name | | Pin Name | Pin No. |
| | | | FWE | 1 |
| 54 | P71 | | A9 | 26 |
| | | | A16 | 2 |
| 60 | P77 | | A15 | 3 |
| 12 | P12 | | $\overline{WE}$ | 31 |
| 45 | P60 | | I/O0 | 13 |
| 46 | P61 | | I/O1 | 14 |
| 47 | P62 | | I/O2 | 15 |
| 48 | P63 | | I/O3 | 17 |
| 49 | P64 | | I/O4 | 18 |
| 50 | P65 | | I/O5 | 19 |
| 51 | P66 | | I/O6 | 20 |
| 52 | P67 | | I/O7 | 21 |
| 68 | P87 | | A0 | 12 |
| 67 | P86 | | A1 | 11 |
| 66 | P85 | | A2 | 10 |
| 65 | P84 | | A3 | 9 |
| 64 | P83 | | A4 | 8 |
| 63 | P82 | | A5 | 7 |
| 62 | P81 | | A6 | 6 |
| 61 | P80 | | A7 | 5 |
| 53 | P70 | | A8 | 27 |
| 71 | P42 | | $\overline{OE}$ | 24 |
| 55 | P72 | | A10 | 23 |
| 56 | P73 | | A11 | 25 |
| 57 | P74 | | A12 | 4 |
| 58 | P75 | | A13 | 28 |
| 59 | P76 | | A14 | 29 |
| 72 | P43 | | $\overline{CE}$ | 22 |
| 26, 32 | CVcc, Vcc | | Vcc | 32 |
| 73 | AVcc | | Vss | 16 |
| 3 | X1 | | | |
| 8 | TEST | | | |
| 30 | V1 | | | |
| 14 | P14 | | | |
| | | | | |
| 2, 5 | AVss, Vss | | | |
| 27 | Vss | | | |
| 74 | PB0 | | | |
| 75 | PB1 | | | |
| 76 | PB2 | | | |
| 7, 6 | OSC1,OSC2 | Oscillator circuit | | |
| 9 | $\overline{RES}$ | Power-on reset circuit | | |
| Other than the above | (OPEN) | | | |

Legend:
FWE: Flash-write enable
I/O7 to I/O0: Data input/output
A16 to A0: Address input
$\overline{CE}$: Chip enable
$\overline{OE}$: Output enable
$\overline{WE}$: Write enable

Note: The oscillation frequency of the oscillator circuit should be 10 MHz.

**Figure 6.6 Socket Adapter Pin Correspondence Diagram**

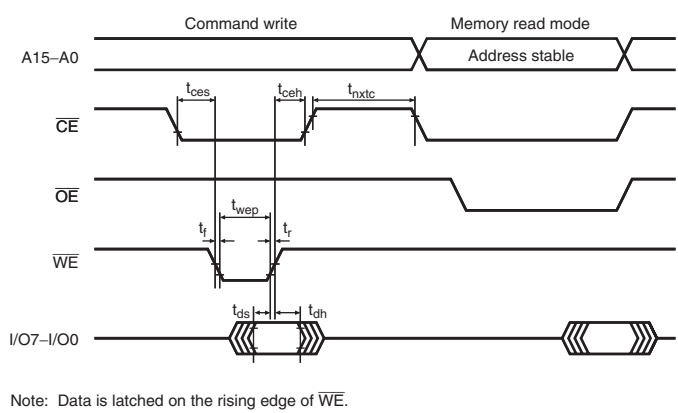
### 6.7.3 Memory Read Mode

1. After completion of auto-program/auto-erase/status read operations, a transition is made to the command wait state. When reading memory contents, a transition to memory read mode must first be made with a command write, after which the memory contents are read. Once memory read mode has been entered, consecutive reads can be performed.
2. In memory read mode, command writes can be performed in the same way as in the command wait state.
3. After powering on, memory read mode is entered.
4. Tables 6.10 to 6.12 show the AC characteristics.

**Table 6.10 AC Characteristics in Transition to Memory Read Mode**

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Command write cycle | $t_{nxtc}$ | 20 | — | μs | Figure 6.7 |
| $\overline{CE}$ hold time | $t_{ceh}$ | 0 | — | ns | |
| $\overline{CE}$ setup time | $t_{ces}$ | 0 | — | ns | |
| Data hold time | $t_{dh}$ | 50 | — | ns | |
| Data setup time | $t_{ds}$ | 50 | — | ns | |
| Write pulse width | $t_{wep}$ | 70 | — | ns | |
| $\overline{WE}$ rise time | $t_r$ | — | 30 | ns | |
| $\overline{WE}$ fall time | $t_f$ | — | 30 | ns | |



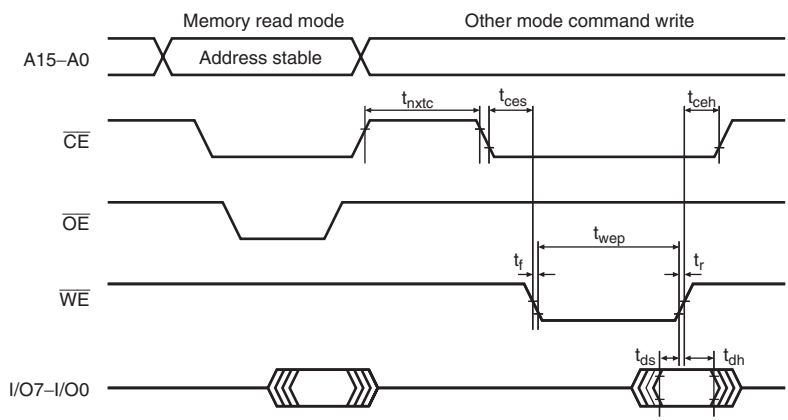

Note: Data is latched on the rising edge of $\overline{WE}$.

**Figure 6.7 Timing Waveforms for Memory Read after Memory Write**

**Table 6.11 AC Characteristics in Transition from Memory Read Mode to Another Mode**

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Command write cycle | $t_{nxtc}$ | 20 | — | μs | Figure 6.8 |
| $\overline{CE}$ hold time | $t_{ceh}$ | 0 | — | ns | |
| $\overline{CE}$ setup time | $t_{ces}$ | 0 | — | ns | |
| Data hold time | $t_{dh}$ | 50 | — | ns | |
| Data setup time | $t_{ds}$ | 50 | — | ns | |
| Write pulse width | $t_{wep}$ | 70 | — | ns | |
| $\overline{WE}$ rise time | $t_r$ | — | 30 | ns | |
| $\overline{WE}$ fall time | $t_f$ | — | 30 | ns | |




Note: Do not enable $\overline{WE}$ and $\overline{OE}$ at the same time.

**Figure 6.8 Timing Waveforms in Transition from Memory Read Mode to Another Mode**

**Table 6.12 AC Characteristics in Memory Read Mode**

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Access time | $t_{acc}$ | — | 20 | μs | Figure 6.9 |
| $\overline{CE}$ output delay time | $t_{ce}$ | — | 150 | ns | Figure 6.10 |
| $\overline{OE}$ output delay time | $t_{oe}$ | — | 150 | ns | |
| Output disable delay time | $t_{df}$ | — | 100 | ns | |
| Data output hold time | $t_{oh}$ | 5 | — | ns | |

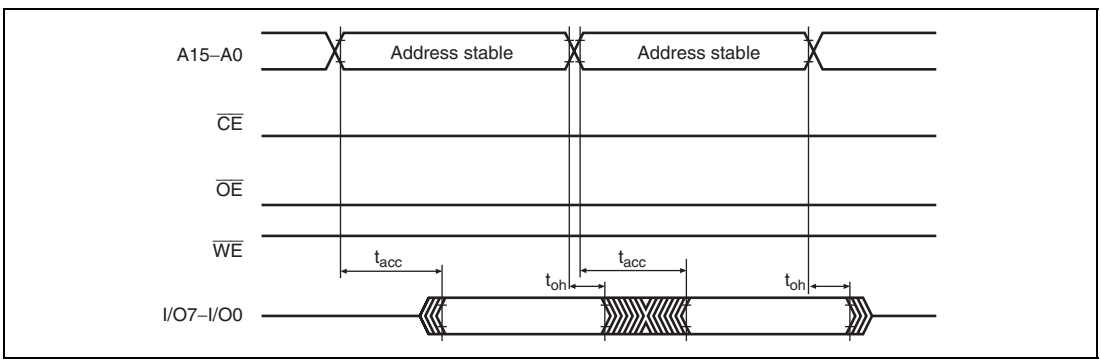



**Figure 6.9 $\overline{CE}$ and $\overline{OE}$ Enable State Read Timing Waveforms**

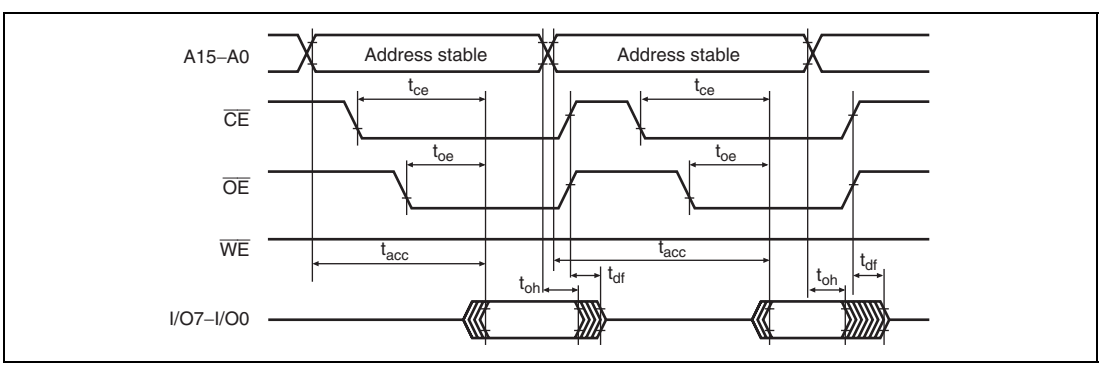



**Figure 6.10 $\overline{CE}$ and $\overline{OE}$ Clock System Read Timing Waveforms**

### 6.7.4 Auto-Program Mode

1. When reprogramming previously programmed addresses, perform auto-erasing before auto-programming.
2. Perform auto-programming once only on the same address block. It is not possible to program an address block that has already been programmed.
3. In auto-program mode, 128 bytes are programmed simultaneously. This should be carried out by executing 128 consecutive byte transfers. A 128-byte data transfer is necessary even when programming fewer than 128 bytes. In this case, H'FF data must be written to the extra addresses.
4. The lower 7 bits of the transfer address must be low. If a value other than an effective address is input, processing will switch to a memory write operation but a write error will be flagged.
5. Memory address transfer is performed in the second cycle (figure 6.11). Do not perform transfer after the third cycle.
6. Do not perform a command write during a programming operation.
7. Perform one auto-program operation for a 128-byte block for each address. Two or more additional programming operations cannot be performed on a previously programmed address block.
8. Confirm normal end of auto-programming by checking I/O6. Alternatively, status read mode can also be used for this purpose (I/O7 status polling uses the auto-program operation end decision pin).
9. Status polling I/O6 and I/O7 pin information is retained until the next command write. As long as the next command write has not been performed, reading is possible by enabling $\overline{CE}$ and $\overline{OE}$.
10. Table 6.13 shows the AC characteristics.

## Table 6.13 AC Characteristics in Auto-Program Mode

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Command write cycle | $t_{nxtc}$ | 20 | — | μs | Figure 6.11 |
| $\overline{CE}$ hold time | $t_{ceh}$ | 0 | — | ns | |
| $\overline{CE}$ setup time | $t_{ces}$ | 0 | — | ns | |
| Data hold time | $t_{dh}$ | 50 | — | ns | |
| Data setup time | $t_{ds}$ | 50 | — | ns | |
| Write pulse width | $t_{wep}$ | 70 | — | ns | |
| Status polling start time | $t_{wsts}$ | 1 | — | ms | |
| Status polling access time | $t_{spa}$ | — | 150 | ns | |
| Address setup time | $t_{as}$ | 0 | — | ns | |
| Address hold time | $t_{ah}$ | 60 | — | ns | |
| Memory write time | $t_{write}$ | 1 | 3000 | ms | |
| $\overline{WE}$ rise time | $t_r$ | — | 30 | ns | |
| $\overline{WE}$ fall time | $t_f$ | — | 30 | ns | |

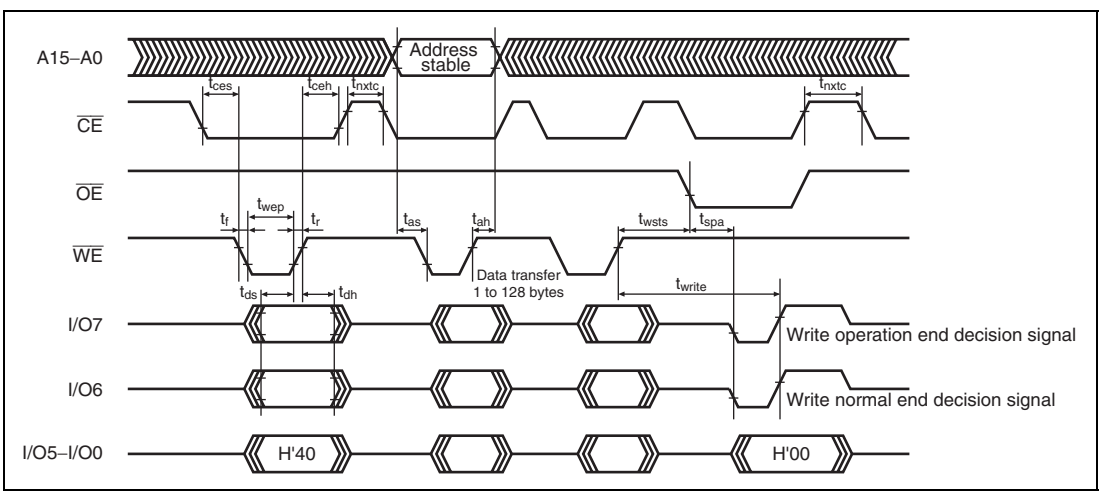



**Figure 6.11 Auto-Program Mode Timing Waveforms**

### 6.7.5 Auto-Erase Mode

1. Auto-erase mode supports only entire memory erasing.
2. Do not perform a command write during auto-erasing.
3. Confirm normal end of auto-erasing by checking I/O6. Alternatively, status read mode can also be used for this purpose (I/O7 status polling uses the auto-erase operation end decision pin).
4. Status polling I/O6 and I/O7 pin information is retained until the next command write. As long as the next command write has not been performed, reading is possible by enabling $\overline{CE}$ and $\overline{OE}$.
5. Table 6.14 shows the AC characteristics.

**Table 6.14 AC Characteristics in Auto-Erase Mode**

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Command write cycle | $t_{nxtc}$ | 20 | — | μs | Figure 6.12 |
| $\overline{CE}$ hold time | $t_{ceh}$ | 0 | — | ns | |
| $\overline{CE}$ setup time | $t_{ces}$ | 0 | — | ns | |
| Data hold time | $t_{dh}$ | 50 | — | ns | |
| Data setup time | $t_{ds}$ | 50 | — | ns | |
| Write pulse width | $t_{wep}$ | 70 | — | ns | |
| Status polling start time | $t_{ests}$ | 1 | — | ms | |
| Status polling access time | $t_{spa}$ | — | 150 | ns | |
| Memory erase time | $t_{erase}$ | 100 | 40000 | ms | |
| $\overline{WE}$ rise time | $t_r$ | — | 30 | ns | |
| $\overline{WE}$ fall time | $t_f$ | — | 30 | ns | |

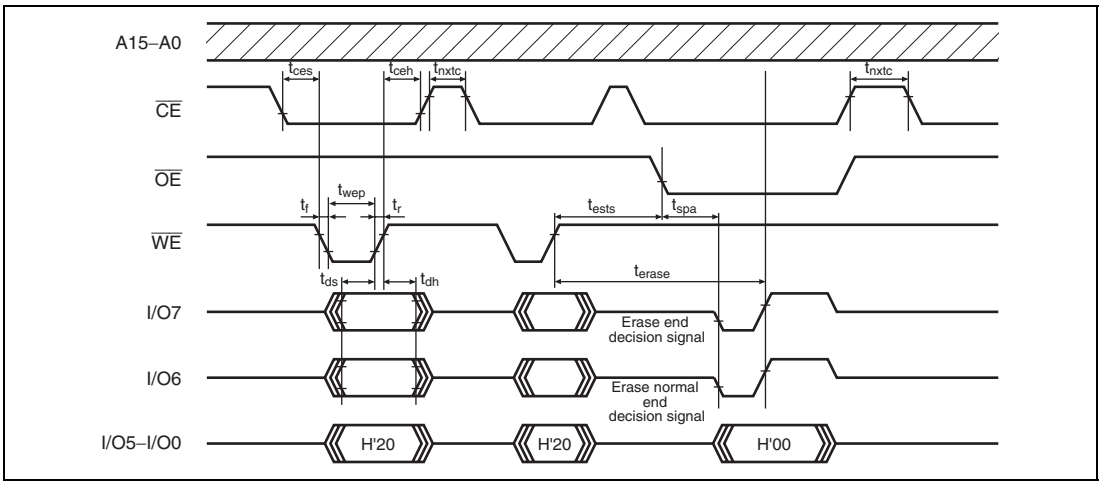



**Figure 6.12 Auto-Erase Mode Timing Waveforms**

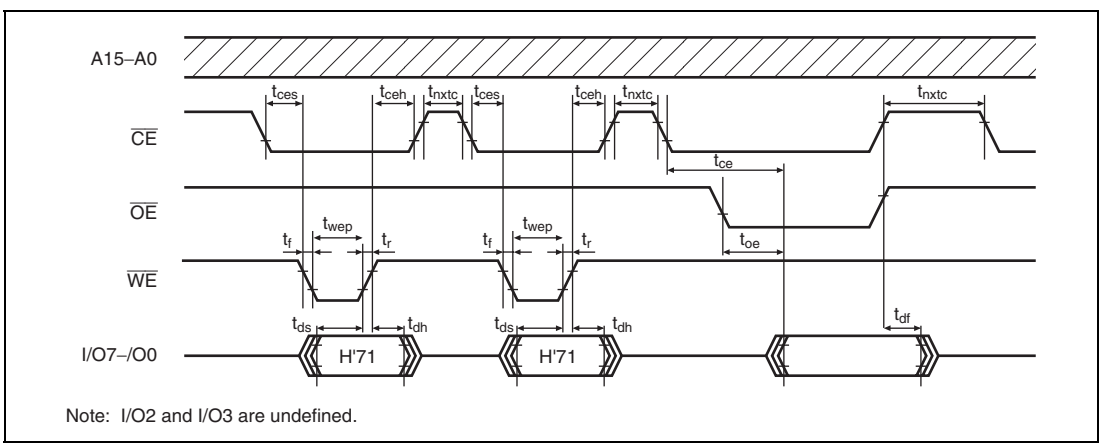
### 6.7.6 Status Read Mode

1. Status read mode is provided to identify the kind of abnormal end. Use this mode when an abnormal end occurs in auto-program mode or auto-erase mode.
2. The return code is retained until a command write other than a status read mode command write is executed.
3. Table 6.15 shows the AC characteristics and 6.16 shows the return codes.

**Table 6.15 AC Characteristics in Status Read Mode**

Conditions: $V_{CC}$ = 3.3 V ±0.3 V, $V_{SS}$ = 0 V, $T_a$ = 25°C ±5°C

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Read time after command write | $t_{nxtc}$ | 20 | — | μs | Figure 6.13 |
| $\overline{CE}$ hold time | $t_{ceh}$ | 0 | — | ns | |
| $\overline{CE}$ setup time | $t_{ces}$ | 0 | — | ns | |
| Data hold time | $t_{dh}$ | 50 | — | ns | |
| Data setup time | $t_{ds}$ | 50 | — | ns | |
| Write pulse width | $t_{wep}$ | 70 | — | ns | |
| $\overline{OE}$ output delay time | $t_{oe}$ | — | 150 | ns | |
| Disable delay time | $t_{df}$ | — | 100 | ns | |
| $\overline{CE}$ output delay time | $t_{ce}$ | — | 150 | ns | |
| $\overline{WE}$ rise time | $t_r$ | — | 30 | ns | |
| $\overline{WE}$ fall time | $t_f$ | — | 30 | ns | |




Note: I/O2 and I/O3 are undefined.

**Figure 6.13 Status Read Mode Timing Waveforms**

**Table 6.16 Status Read Mode Return Codes**

| Pin Name | Initial Value | Indications |
|---|---|---|
| I/O7 | 0 | 1: Abnormal end<br>0: Normal end |
| I/O6 | 0 | 1: Command error<br>0: Otherwise |
| I/O5 | 0 | 1: Programming error<br>0: Otherwise |
| I/O4 | 0 | 1: Erasing error<br>0: Otherwise |
| I/O3 | 0 | — |
| I/O2 | 0 | — |
| I/O1 | 0 | 1: Over counting of writing or erasing<br>0: Otherwise |
| I/O0 | 0 | 1: Effective address error<br>0: Otherwise |

### 6.7.7 Status Polling

1. The I/O7 status polling flag indicates the operating status in auto-program/auto-erase mode.
2. The I/O6 status polling flag indicates a normal or abnormal end in auto-program/auto-erase mode.

**Table 6.17 Status Polling Output Truth Table**

| I/O7 | I/O6 | I/O0 to 5 | Status |
|---|---|---|---|
| 0 | 0 | 0 | During internal operation |
| 1 | 0 | 0 | Abnormal end |
| 1 | 1 | 0 | Normal end |
| 0 | 1 | 0 | — |

### 6.7.8 Programmer Mode Transition Time

Commands cannot be accepted during the oscillation stabilization period or the programmer mode setup period. After the programmer mode setup time, a transition is made to memory read mode.

**Table 6.18 Stipulated Transition Times to Command Wait State**

| Item | Symbol | Min | Max | Unit | Notes |
|---|---|---|---|---|---|
| Oscillation stabilization time (crystal oscillator) | $T_{osc1}$ | 10 | — | ms | Figure 6.14 |
| Oscillation stabilization time (ceramic oscillator) | $T_{osc1}$ | 5 | — | ms | |
| Programmer mode setup time | $T_{bmv}$ | 10 | — | ms | |
| Vcc hold time | $T_{dwn}$ | 0 | — | ms | |

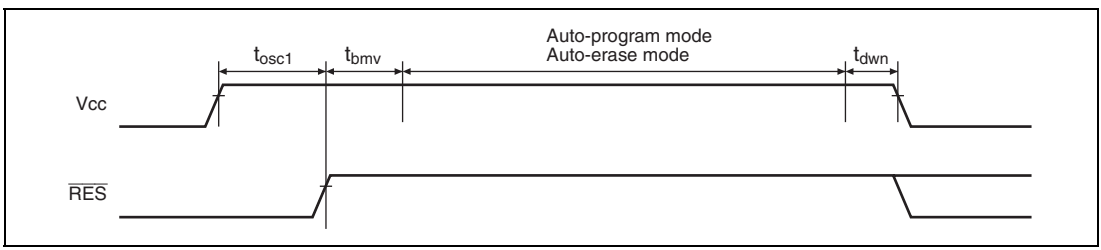



**Figure 6.14 Oscillation Stabilization Time, Boot Program Transfer Time, and Power-Down Sequence**

### 6.7.9 Notes on Memory Programming

1. When performing programming using programmer mode on a chip that has been programmed/erased in an on-board programming mode, auto-erasing is recommended before carrying out auto-programming.
2. The flash memory is initially in the erased state when the device is shipped by Renesas Technology. For other chips for which the erasure history is unknown, it is recommended that auto-erasing be executed to check and supplement the initialization (erase) level.

## 6.8 Power-Down States for Flash Memory

In user mode, the flash memory will operate in either of the following states:

- Normal operating mode
  The flash memory can be read and written to at high speed.
- Power-down operating mode
  The power supply circuit of the flash memory is partly halted and can be read under low power consumption.
- Standby mode
  All flash memory circuits are halted.

Table 6.19 shows the correspondence between the operating modes of this LSI and the flash memory. In subactive mode, the flash memory can be set to operate in power-down mode with the PDWND bit in FLPWCR. When the flash memory returns to its normal operating state from power-down mode or standby mode, a period to stabilize the power supply circuits that were stopped is needed. When the flash memory returns to its normal operating state, bits STS2 to STS0 in SYSCR1 must be set to provide a wait time of at least 20 μs, even when the external clock is being used.

**Table 6.19 Flash Memory Operating States**

| | Flash Memory Operating State | |
|---|---|---|
| **LSI Operating State** | **PDWND = 0 (Initial value)** | **PDWND = 1** |
| Active mode | Normal operating mode | Normal operating mode |
| Subactive mode | Power-down mode | Normal operating mode |
| Sleep mode | Normal operating mode | Normal operating mode |
| Subsleep mode | Standby mode | Standby mode |
| Standby mode | Standby mode | Standby mode |
| Watch mode | Standby mode | Standby mode |

# Section 7 RAM

## 7.1 Overview

The H8/38532 and H8/38533 have 1 Kbyte of high-speed static RAM on-chip, and the H8/38534, H8/38535, H8/38536, and H8/38537 have 2 Kbytes. The RAM is connected to the CPU by a 16-bit data bus, allowing high-speed 2-state access for both byte data and word data.

### 7.1.1 Block Diagram

Figure 7.1 shows a block diagram of the on-chip RAM.

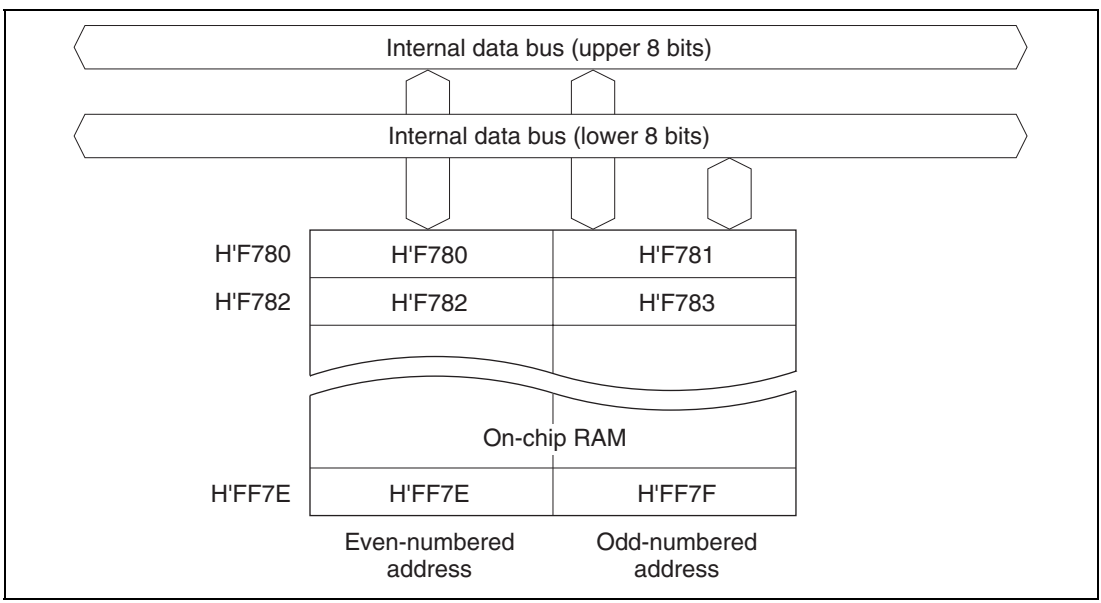



**Figure 7.1 RAM Block Diagram (H8/38534)**

# Section 8 I/O Ports

## 8.1 Overview

The LSI is provided with six 8-bit I/O ports, one 4-bit I/O port, one 3-bit I/O port, one 8-bit input-only port, and one 1-bit input-only port. Table 8.1 indicates the functions of each port.

Each port has of a port control register (PCR) that controls input and output, and a port data register (PDR) for storing output data. Input or output can be assigned to individual bits.
See section 2.8.3, Bit-Manipulation Instruction, for information on executing bit-manipulation instructions to write data in PCR or PDR.

Ports 5, 6, 7, 8, and A are also used as liquid crystal display segment and common pins, selectable in 8-bit units.

Block diagrams of each port are given in Appendix B, I/O Port Block Diagrams.

**Table 8.1 Port Functions**

| Port | Description | Pins | Other Functions | Function Switching Registers |
|---|---|---|---|---|
| Port 1 | • 8-bit I/O port<br>• MOS input pull-up option | $P1_7$ to $P1_5$/ $\overline{IRQ_3}$ to $\overline{IRQ_1}$/ TMIF, TMIC | External interrupts 3 to 1<br>Timer event interrupts TMIF, TMIC | PMR1<br>TCRF, TMC |
| | | $P1_4$/$\overline{IRQ_4}$/$\overline{ADTRG}$ | External interrupt 4 and A/D converter external trigger | PMR1, AMR |
| | | $P1_3$/TMIG | Timer G input capture input | PMR1 |
| | | $P1_2$, $P1_1$/ TMOFH, TMOFL | Timer F output compare output | PMR1 |
| | | $P1_0$/TMOW | Timer A clock output | PMR1 |
| Port 3 | • 8-bit I/O port<br>• MOS input pull-up option<br>• Large-current port | $P3_7$/AEVL<br>$P3_6$/AEVH<br>$P3_5$/$TXD_{31}$<br>$P3_4$/$RXD_{31}$<br>$P3_3$/$SCK_{31}$ | SCI31 data output ($TXD_{31}$), data input ($RXD_{31}$), clock input/output ($SCK_{31}$), and asynchronous counter event inputs AEVL, AEVH | PMR3<br>SCR31<br>SMR31 |
| | | $P3_2$<br>$P3_1$/UD/EXCL<br>$P3_0$/PWM | Timer C count-up/down select input, 14-bit PWM output, and external subclock input | PMR2<br>PMR3 |

| Port | Description | Pins | Other Functions | Function Switching Registers |
|---|---|---|---|---|
| Port 4 | • 1-bit input port | $P4_3/\overline{IRQ}_0$ | External interrupt 0 | PMR3 |
| | • 3-bit I/O port | $P4_2/TXD_{32}$<br>$P4_1/RXD_{32}$<br>$P4_0/SCK_{32}$ | SCI32 data output ($TXD_{32}$), data input ($RXD_{32}$), clock input/output ($SCK_{32}$) | SCR32<br>SMR32 |
| Port 5 | • 8-bit I/O port<br>• MOS input pull-up option | $P5_7$ to $P5_0$/<br>$\overline{WKP}_7$ to $\overline{WKP}_0$/<br>$SEG_8$ to $SEG_1$ | Wakeup input ($\overline{WKP}_7$ to $\overline{WKP}_0$), segment output ($SEG_8$ to $SEG_1$) | PMR5<br>LPCR |
| Port 6 | • 8-bit I/O port<br>• MOS input pull-up option | $P6_7$ to $P6_0$/<br>$SEG_{16}$ to $SEG_9$ | Segment output ($SEG_{16}$ to $SEG_9$) | LPCR |
| Port 7 | • 8-bit I/O port | $P7_7$ to $P7_0$/<br>$SEG_{24}$ to $SEG_{17}$ | Segment output ($SEG_{24}$ to $SEG_{17}$) | LPCR |
| Port 8 | • 8-bit I/O port | $P8_7/SEG_{32}$<br>$P8_6/SEG_{31}$<br>$P8_5/SEG_{30}$<br>$P8_4/SEG_{29}$<br>$P8_3$ to $P8_0$/<br>$SEG_{28}$ to $SEG_{25}$ | Segment output ($SEG_{32}$ to $SEG_{25}$) | LPCR |
| Port A | 4-bit I/O port | $PA_3$ to $PA_0$/<br>$COM_4$ to $COM_1$ | Common output ($COM_4$ to $COM_1$) | LPCR |
| Port B | 8-bit input port | $PB_7$ to $PB_0$/<br>$AN_7$ to $AN_0$ | A/D converter analog input | AMR |

## 8.2 Port 1

### 8.2.1 Overview

Port 1 is a 8-bit I/O port. Figure 8.1 shows its pin configuration.

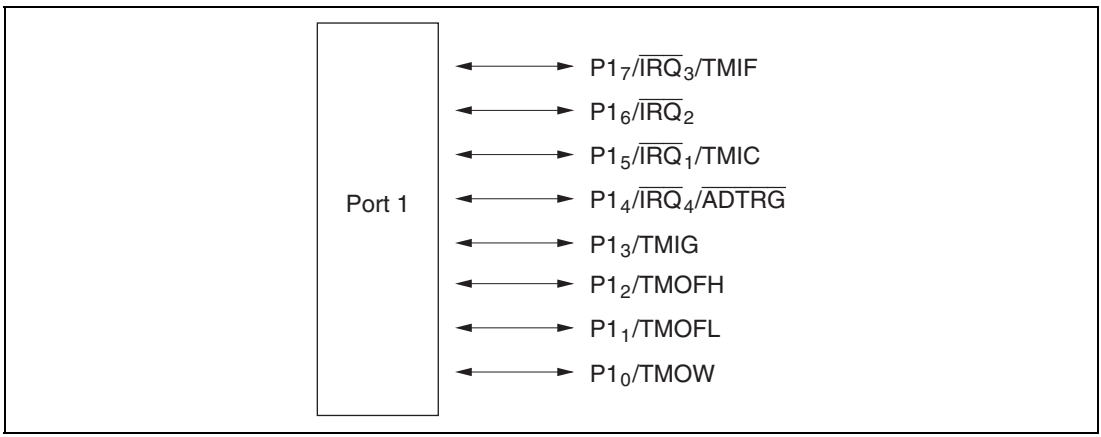



**Figure 8.1 Port 1 Pin Configuration**

### 8.2.2 Register Configuration and Description

Table 8.2 shows the port 1 register configuration.

**Table 8.2 Port 1 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 1 | PDR1 | R/W | H'00 | H'FFD4 |
| Port control register 1 | PCR1 | W | H'00 | H'FFE4 |
| Port pull-up control register 1 | PUCR1 | R/W | H'00 | H'FFE0 |
| Port mode register 1 | PMR1 | R/W | H'00 | H'FFC8 |

### (1) Port Data Register 1 (PDR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P1_7$ | $P1_6$ | $P1_5$ | $P1_4$ | $P1_3$ | $P1_2$ | $P1_1$ | $P1_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR1 is an 8-bit register that stores data for port 1 pins $P1_7$ to $P1_0$. If port 1 is read while PCR1 bits are set to 1, the values stored in PDR1 are read, regardless of the actual pin states. If port 1 is read while PCR1 bits are cleared to 0, the pin states are read.

Upon reset, PDR1 is initialized to H'00.

### (2) Port Control Register 1 (PCR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR1_7$ | $PCR1_6$ | $PCR1_5$ | $PCR1_4$ | $PCR1_3$ | $PCR1_2$ | $PCR1_1$ | $PCR1_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR1 is an 8-bit register for controlling whether each of the port 1 pins $P1_7$ to $P1_0$ functions as an input pin or output pin. Setting a PCR1 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. The settings in PCR1 and in PDR1 are valid only when the corresponding pin is designated in PMR1 as a general I/O pin.

Upon reset, PCR1 is initialized to H'00.

PCR1 is a write-only register, which is always read as all 1s.

### (3) Port Pull-Up Control Register 1 (PUCR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PUCR1_7$ | $PUCR1_6$ | $PUCR1_5$ | $PUCR1_4$ | $PUCR1_3$ | $PUCR1_2$ | $PUCR1_1$ | $PUCR1_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PUCR1 controls whether the MOS pull-up of each of the port 1 pins $P1_7$ to $P1_0$ is on or off. When a PCR1 bit is cleared to 0, setting the corresponding PUCR1 bit to 1 turns on the MOS pull-up for the corresponding pin, while clearing the bit to 0 turns off the MOS pull-up.

Upon reset, PUCR1 is initialized to H'00.

### (4) Port Mode Register 1 (PMR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IRQ3 | IRQ2 | IRQ1 | IRQ4 | TMIG | TMOFH | TMOFL | TMOW |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PMR1 is an 8-bit read/write register, controlling the selection of pin functions for port 1 pins.

Upon reset, PMR1 is initialized to H'00.

**Bit 7:** $P1_7/\overline{IRQ_3}$/TMIF pin function switch (IRQ3)

This bit selects whether pin $P1_7/\overline{IRQ_3}$/TMIF is used as $P1_7$ or as $\overline{IRQ_3}$/TMIF.

| Bit 7<br>IRQ3 | Description | |
|---|---|---|
| 0 | Functions as $P1_7$ I/O pin | (initial value) |
| 1 | Functions as $\overline{IRQ_3}$/TMIF input pin | |

Note: Rising or falling edge sensing can be designated for $IRQ_3$, TMIF. For details on TMIF settings, see (3) Timer Control Register F (TCRF) in section 9.4.2, Register Descriptions.

**Bit 6:** $P1_6/\overline{IRQ_2}$ pin function switch (IRQ2)

This bit selects whether pin $P1_6/\overline{IRQ_2}$ is used as $P1_6$ or as $\overline{IRQ_2}$.

| Bit 6<br>IRQ2 | Description | |
|---|---|---|
| 0 | Functions as $P1_6$ I/O pin | (initial value) |
| 1 | Functions as $\overline{IRQ_2}$ input pin | |

Note: Rising or falling edge sensing can be designated for $\overline{IRQ_2}$.

**Bit 5:** $P1_5/\overline{IRQ_1}$/TMIC pin function switch (IRQ1)

This bit selects whether pin $P1_5/\overline{IRQ_1}$/TMIC is used as $P1_5$ or as $\overline{IRQ_1}$/TMIC.

| Bit 5 IRQ1 | Description | |
|---|---|---|
| 0 | Functions as $P1_5$ I/O pin | (initial value) |
| 1 | Functions as $\overline{IRQ_1}$/TMIC input pin | |

Note: Rising or falling edge sensing can be designated for $\overline{IRQ_1}$/TMIC.
For details of TMIC pin setting, see (1) Timer Mode Register C (TMC) in section 9.3.2, Register Descriptions.

**Bit 4:** $P1_4/\overline{IRQ_4}/\overline{ADTRG}$ pin function switch (IRQ4)

This bit selects whether pin $P1_4/\overline{IRQ_4}/\overline{ADTRG}$ is used as $P1_4$ or as $\overline{IRQ_4}/\overline{ADTRG}$.

| Bit 4 IRQ4 | Description | |
|---|---|---|
| 0 | Functions as $P1_4$ I/O pin | (initial value) |
| 1 | Functions as $\overline{IRQ_4}/\overline{ADTRG}$ input pin | |

Note: For details of $\overline{ADTRG}$ pin setting, see section 12.3.2, Start of A/D Conversion by External Trigger Input.

**Bit 3:** $P1_3$/TMIG pin function switch (TMIG)

This bit selects whether pin $P1_3$/TMIG is used as $P1_3$ or as TMIG.

| Bit 3 TMIG | Description | |
|---|---|---|
| 0 | Functions as $P1_3$ I/O pin | (initial value) |
| 1 | Functions as TMIG input pin | |

**Bit 2:** $P1_2$/TMOFH pin function switch (TMOFH)

This bit selects whether pin $P1_2$/TMOFH is used as $P1_2$ or as TMOFH.

**Bit 1:** $P1_1$/TMOFL pin function switch (TMOFL)

This bit selects whether pin $P1_1$/TMOFL is used as $P1_1$ or as TMOFL.

| Bit 1<br>TMOFL | Description | |
|---|---|---|
| 0 | Functions as $P1_1$ I/O pin | (initial value) |
| 1 | Functions as TMOFL output pin | |

**Bit 0:** $P1_0$/TMOW pin function switch (TMOW)

This bit selects whether pin $P1_0$/TMOW is used as $P1_0$ or as TMOW.

| Bit 0<br>TMOW | Description | |
|---|---|---|
| 0 | Functions as $P1_0$ I/O pin | (initial value) |
| 1 | Functions as TMOW output pin | |

### 8.2.3 Pin Functions

Table 8.3 shows the port 1 pin functions.

**Table 8.3 Port 1 Pin Functions**

| Pin | Pin Functions and Selection Method |
|---|---|
| $P1_7/\overline{IRQ_3}$/TMIF | The pin function depends on bit IRQ3 in PMR1, bits CKSL2 to CKSL0 in TCRF, and bit $PCR1_7$ in PCR1. |

| $IRQ_3$ | 0 | | 1 | |
|---|---|---|---|---|
| $PCR1_7$ | 0 | 1 | * | |
| CKSL2 to CKSL0 | * | | Not 0** | 0** |
| Pin function | $P1_7$ input pin | $P1_7$ output pin | $\overline{IRQ_3}$ input pin | $\overline{IRQ_3}$/TMIF input pin |

Note: When this pin is used as the TMIF input pin, clear bit IEN3 to 0 in IENR1 to disable the $IRQ_3$ interrupt.

| | |
|---|---|
| $P1_6/\overline{IRQ_2}$ | The pin function depends on bits IRQ2 in PMR1 and bit $PCR1_6$ in PCR1. |

| IRQ2 | 0 | | 1 |
|---|---|---|---|
| $PCR1_6$ | 0 | 1 | * |
| Pin function | $P1_6$ input pin | $P1_6$ output pin | $\overline{IRQ_2}$ input pin |

| | |
|---|---|
| $P1_5/\overline{IRQ_1}$ TMIC | The pin function depends on bit IRQ1 in PMR1, bits TMC2 to TMC0 in TMC, and bit $PCR1_5$ in PCR1. |

| IRQ1 | 0 | | 1 | |
|---|---|---|---|---|
| $PCR1_5$ | 0 | 1 | * | |
| TMC2 to TMC0 | * | | Not 111 | 111 |
| Pin function | $P1_5$ input pin | $P1_5$ output pin | $\overline{IRQ_1}$ input pin | $\overline{IRQ_1}$/TMIC input pin |

Note: When this pin is used as the TMIC input pin, clear bit IEN1 to 0 in IENR1 to disable the $IRQ_1$ interrupt.

| Pin | Pin Functions and Selection Method |
|---|---|
| $P1_4/\overline{IRQ_4}$ $\overline{ADTRG}$ | The pin function depends on bit IRQ4 in PMR1, bit TRGE in AMR, and bit $PCR1_4$ in PCR1. |

| IRQ4 | 0 | | 1 | |
|---|---|---|---|---|
| $PCR1_4$ | 0 | 1 | * | |
| TRGE | * | | 0 | 1 |
| Pin function | $P1_4$ input pin | $P1_4$ output pin | $\overline{IRQ_4}$ input pin | $\overline{IRQ_4}/\overline{ADTRG}$ input pin |

Note: When this pin is used as the $\overline{ADTRG}$ input pin, clear bit IEN4 to 0 in IENR1 to disable the $IRQ_4$ interrupt.

$P1_3$/TMIG — The pin function depends on bit TMIG in PMR1 and bit $PCR1_3$ in PCR1.

| TMIG | 0 | | 1 |
|---|---|---|---|
| $PCR1_3$ | 0 | 1 | * |
| Pin function | $P1_3$ input pin | $P1_3$ output pin | TMIG input pin |

$P1_2$/TMOFH — The pin function depends on bit TMOFH in PMR1 and bit $PCR1_2$ in PCR1.

| TMOFH | 0 | | 1 |
|---|---|---|---|
| $PCR1_2$ | 0 | 1 | * |
| Pin function | $P1_2$ input pin | $P1_2$ output pin | TMOFH output pin |

$P1_1$/TMOFL — The pin function depends on bit TMOFL in PMR1 and bit $PCR1_1$ in PCR1.

| TMOFL | 0 | | 1 |
|---|---|---|---|
| $PCR1_1$ | 0 | 1 | * |
| Pin function | $P1_1$ input pin | $P1_1$ output pin | TMOFL output pin |

$P1_0$/TMOW — The pin function depends on bit TMOW in PMR1 and bit $PCR1_0$ in PCR1.

| TMOW | 0 | | 1 |
|---|---|---|---|
| $PCR1_0$ | 0 | 1 | * |
| Pin function | $P1_0$ input pin | $P1_0$ output pin | TMOW output pin |

*: Don't care

### 8.2.4 Pin States

Table 8.4 shows the port 1 pin states in each operating mode.

**Table 8.4 Port 1 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P1_7/\overline{IRQ_3}$/TMIF<br>$P1_6/\overline{IRQ_2}$<br>$P1_5/\overline{IRQ_1}$/TMIC<br>$P1_4/\overline{IRQ_4}/\overline{ADTRG}$<br>$P1_3$/TMIG<br>$P1_2$/TMOFH<br>$P1_1$/TMOFL<br>$P1_0$/TMOW | High-impedance | Retains previous state | Retains previous state | High-impedance* | Retains previous state | Functional | Functional |

Note: * A high-level signal is output when the MOS pull-up is in the on state.

### 8.2.5 MOS Input Pull-Up

Port 1 has a built-in MOS input pull-up function that can be controlled by software. When a PCR1 bit is cleared to 0, setting the corresponding PUCR1 bit to 1 turns on the MOS input pull-up for that pin. The MOS input pull-up function is in the off state after a reset.

| | | | |
|---|---|---|---|
| $PCR1_n$ | 0 | 0 | 1 |
| $PUCR1_n$ | 0 | 1 | * |
| MOS input pull-up | Off | On | Off |

(n = 7 to 0)

*: Don't care

## 8.3 Port 3

### 8.3.1 Overview

Port 3 is a 8-bit I/O port, configured as shown in figure 8.2.

In the flash memory version, the on-chip pull-up MOS for pin $P3_2$ is on during the reset period. It turns off and normal operation resumes after the reset is cleared. This should be considered when making connections to external circuitry. Note that in the mask ROM version $P3_2$ continues to operate normally.

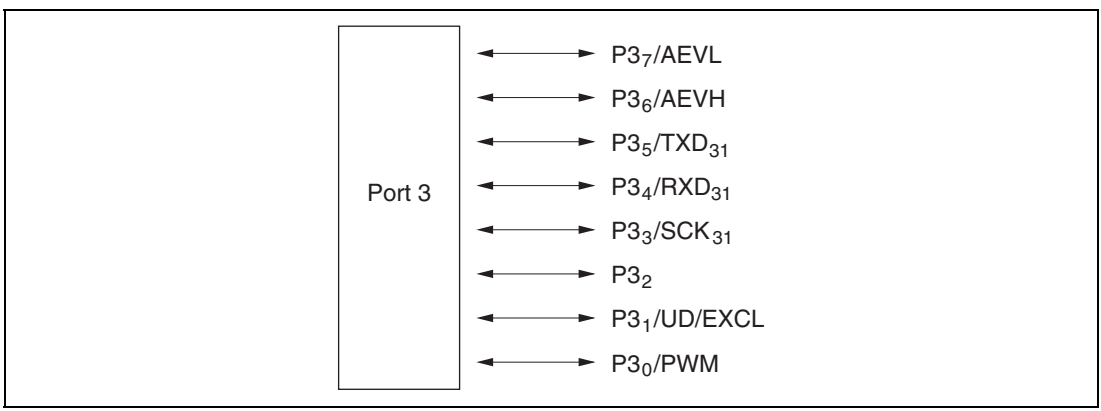



**Figure 8.2 Port 3 Pin Configuration**

### 8.3.2 Register Configuration and Description

Table 8.5 shows the port 3 register configuration.

**Table 8.5 Port 3 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 3 | PDR3 | R/W | H'00 | H'FFD6 |
| Port control register 3 | PCR3 | W | H'00 | H'FFE6 |
| Port pull-up control register 3 | PUCR3 | R/W | H'00 | H'FFE1 |
| Port mode register 2 | PMR2 | R/W | H'58 | H'FFC9 |
| Port mode register 3 | PMR3 | R/W | H'04 | H'FFCA |

### (1) Port Data Register 3 (PDR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR3 is an 8-bit register that stores data for port 3 pins $P3_7$ to $P3_0$. If port 3 is read while PCR3 bits are set to 1, the values stored in PDR3 are read, regardless of the actual pin states. If port 3 is read while PCR3 bits are cleared to 0, the pin states are read.

Upon reset, PDR3 is initialized to H'00.

### (2) Port Control Register 3 (PCR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR3_7$ | $PCR3_6$ | $PCR3_5$ | $PCR3_4$ | $PCR3_3$ | $PCR3_2$ | $PCR3_1$ | $PCR3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR3 is an 8-bit register for controlling whether each of the port 3 pins $P3_7$ to $P3_0$ functions as an input pin or output pin. Setting a PCR3 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. The settings in PCR3 and in PDR3 are valid only when the corresponding pin is designated in PMR3 as a general I/O pin.

Upon reset, PCR3 is initialized to H'00.

PCR3 is a write-only register, which is always read as all 1s.

### (3) Port Pull-Up Control Register 3 (PUCR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PUCR3_7$ | $PUCR3_6$ | $PUCR3_5$ | $PUCR3_4$ | $PUCR3_3$ | $PUCR3_2$ | $PUCR3_1$ | $PUCR3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PUCR3 controls whether the MOS pull-up of each of the port 3 pins $P3_7$ to $P3_0$ is on or off. When a PCR3 bit is cleared to 0, setting the corresponding PUCR3 bit to 1 turns on the MOS pull-up for the corresponding pin, while clearing the bit to 0 turns off the MOS pull-up.

Upon reset, PUCR3 is initialized to H'00.

### (4) Port Mode Register 3 (PMR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AEVL | AEVH | WDCKS | NCS | IRQ0 | — | UD | PWM |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

PMR3 is an 8-bit read/write register, controlling the selection of pin functions for port 3 pins.

Upon reset, PMR3 is initialized to H'04.

**Bit 7:** $P3_7$/AEVL pin function switch (AEVL)

This bit selects whether pin $P3_7$/AEVL is used as $P3_7$ or as AEVL.

| Bit 7<br>AEVL | Description | |
|---|---|---|
| 0 | Functions as $P3_7$ I/O pin | (initial value) |
| 1 | Functions as AEVL input pin | |

**Bit 6:** $P3_6$/AEVH pin function switch (AEVH)

This bit selects whether pin $P3_6$/AEVH is used as $P3_6$ or as AEVH.

| Bit 6<br>AEVH | Description | |
|---|---|---|
| 0 | Functions as $P3_6$ I/O pin | (initial value) |
| 1 | Functions as AEVH input pin | |

**Bit 5:** Watchdog timer source clock select (WDCKS)

This bit selects the watchdog timer source clock.

| Bit 5<br>WDCKS | Description | |
|---|---|---|
| 0 | $\phi$/8192 selected | (initial value) |
| 1 | $\phi$w/32 selected | |

**Bit 4:** TMIG noise canceler select (NCS)

This bit controls the noise canceler for the input capture input signal (TMIG).

| Bit 4<br>NCS | Description | |
|---|---|---|
| 0 | Noise cancellation function not used | (initial value) |
| 1 | Noise cancellation function used | |

**Bit 3:** $P4_3/\overline{IRQ_0}$ pin function switch ($IRQ_0$)

This bit selects whether pin $P4_3/\overline{IRQ_0}$ is used as $P4_3$ or as $\overline{IRQ_0}$.

| Bit 3<br>$IRQ_0$ | Description | |
|---|---|---|
| 0 | Functions as $P4_3$ input pin | (initial value) |
| 1 | Functions as $\overline{IRQ_0}$ input pin | |

**Bit 2:** Reserved bit

This bit cannot be written to.

**Bit 1:** $P3_1$/UD pin function switch (UD)

This bit selects whether pin $P3_1$/UD is used as $P3_1$ or as UD.

| Bit 1<br>UD | Description | |
|---|---|---|
| 0 | Functions as $P3_1$ I/O pin | (initial value) |
| 1 | Functions as UD input pin | |

**Bit 0:** $P3_0$/PWM pin function switch (PWM)

This bit selects whether pin $P3_0$/PWM is used as $P3_0$ or as PWM.

| Bit 0<br>PWM | Description | |
|---|---|---|
| 0 | Functions as $P3_0$ I/O pin | (initial value) |
| 1 | Functions as PWM output pin | |

### (5) Port Mode Register 2 (PMR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EXCL | — | — | — | — | — | — | — |
| Initial value | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | R/W | R | R/W | R | R | R/W | R/W | R/W |

PMR2 is an 8-bit read/write register that controls external clock input to pin $P3_1$.

Upon reset, PMR2 is initialized to H'58.

**Bit 7:** $P3_1$/UD/EXCL pin function switch (EXCL)

This bit selects whether pin $P3_1$/UD/EXCL is used as $P3_1$/UD or as EXCL. When the pin is used as EXCL an external clock should be input to it. See section 4, Clock Pulse Generators, for a connection example.

| Bit 7<br>EXCL | Description | |
|---|---|---|
| 0 | Functions as $P3_1$/UD I/O pin | (initial value) |
| 1 | Functions as EXCL input pin | |

**Bit 6:** Reserved bit

Bit 6 is a reserved bit. It is always read as 1 and cannot be modified.

**Bit 5:** Reserved bit

Bit 5 is a readable/writable reserved bit.

**Bits 4 and 3:** Reserved bits

Bits 4 and 3 are reserved bits. They are always read as 1 and cannot be modified.

**Bits 2 to 0:** Reserved bits

Bits 2 to 0 are readable/writable reserved bits.

### 8.3.3 Pin Functions

Table 8.6 shows the port 3 pin functions.

**Table 8.6 Port 3 Pin Functions**

| Pin | Pin Functions and Selection Method |
|---|---|
| $P3_7$/AEVL | The pin function depends on bit SO1 in PMR3 and bit $PCR3_7$ in PCR3. |

| AEVL | 0 | | 1 |
|---|---|---|---|
| $PCR3_7$ | 0 | 1 | * |
| Pin function | $P3_7$ input pin | $P3_7$ output pin | AEVL input pin |

$P3_6$/AEVH — The pin function depends on bit AEVH in PMR3 and bit $PCR3_6$ in PCR3.

| AEVH | 0 | | 1 |
|---|---|---|---|
| $PCR3_6$ | 0 | 1 | * |
| Pin function | $P3_6$ input pin | $P3_6$ output pin | AEVH input pin |

$P3_5$/$TXD_{31}$ — The pin function depends on bit $TE_{31}$ in SCR31, bit SPC31 in SPCR, and bit $PCR3_5$ in PCR3.

| SPC31 | 0 | | 1 |
|---|---|---|---|
| $TE_{31}$ | 0 | | 1 |
| $PCR3_5$ | 0 | 1 | * |
| Pin function | $P3_5$ input pin | $P3_5$ output pin | $TXD_{31}$ output pin |

$P3_4$/$RXD_{31}$ — The pin function depends on bit $RE_{31}$ in SCR31 and bit $PCR3_4$ in PCR3.

| $RE_{31}$ | 0 | | 1 |
|---|---|---|---|
| $PCR3_4$ | 0 | 1 | * |
| Pin function | $P3_4$ input pin | $P3_4$ output pin | $RXD_{31}$ input pin |

| Pin | Pin Functions and Selection Method |
|---|---|

**$P3_3/SCK_{31}$**

The pin function depends on bits CKE311 and CKE310 in SCR31, bit COM31 in SMR31, and bit $PCR3_3$ in PCR3.

| CKE311 | 0 | | | | 1 |
|---|---|---|---|---|---|
| CKE310 | 0 | | | 1 | * |
| $COM3_1$ | 0 | | 1 | * | * |
| $PCR3_3$ | 0 | 1 | * | | * |
| Pin function | $P3_3$ input pin | $P3_3$ output pin | $SCK_{31}$ output pin | | $SCK_{31}$ input pin |

**$P3_2$**

The pin function depends on bit $PCR3_2$ in PCR3.

| $PCR3_2$ | 0 | 1 |
|---|---|---|
| Pin function | $P3_2$ input pin | $P3_2$ output pin |

**$P3_1$/UD/EXCL**

The pin function depends on bit EXCL in PMR2, bit UD in PMR3, and bit $PCR3_1$ in PCR3.

| EXCL | 0 | | | 1 |
|---|---|---|---|---|
| UD | 0 | | 1 | * |
| $PCR3_1$ | 0 | 1 | * | * |
| Pin function | $P3_1$ input pin | $P3_1$ output pin | UD input pin | EXCL input pin |

**$P3_0$/PWM**

The pin function depends on bit PWM in PMR3 and bit $PCR3_0$ in PCR3.

| PWM | 0 | | 1 |
|---|---|---|---|
| $PCR3_0$ | 0 | 1 | * |
| Pin function | $P3_0$ input pin | $P3_0$ output pin | PWM output pin |

*: Don't care

### 8.3.4 Pin States

Table 8.7 shows the port 3 pin states in each operating mode.

**Table 8.7 Port 3 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P3_7$/AEVL<br>$P3_6$/AEVH<br>$P3_5$/$TXD_{31}$<br>$P3_4$/$RXD_{31}$<br>$P3_3$/$SCK_{31}$ | High-impedance | Retains previous state | Retains previous state | High-impedance*1 | Retains previous state | Functional | Functional |
| $P3_2$*3 | Pull-up MOS on | | | | | | |
| $P3_2$*2<br>$P3_1$/UD/EXCL<br>$P3_0$/PWM | High-impedance | | | | | | |

Notes: 1. A high-level signal is output when the MOS pull-up is in the on state.
2. Applies to the mask ROM version.
3. Applies to the flash memory version.

### 8.3.5 MOS Input Pull-Up

Port 3 has a built-in MOS input pull-up function that can be controlled by software. When a PCR3 bit is cleared to 0, setting the corresponding PUCR3 bit to 1 turns on the MOS pull-up for that pin. The MOS pull-up function is in the off state after a reset.

| | | | |
|---|---|---|---|
| $PCR3_n$ | 0 | 0 | 1 |
| $PUCR3_n$ | 0 | 1 | * |
| MOS input pull-up | Off | On | Off |

(n = 7 to 0)

*: Don't care

## 8.4 Port 4

### 8.4.1 Overview

Port 4 is a 3-bit I/O port and 1-bit input port, configured as shown in figure 8.3.

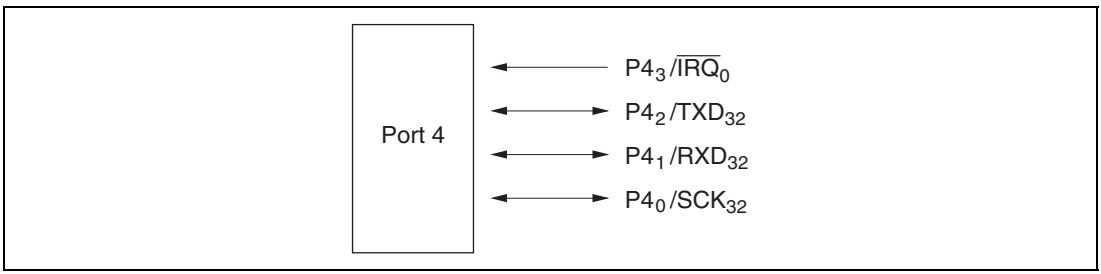



**Figure 8.3 Port 4 Pin Configuration**

### 8.4.2 Register Configuration and Description

Table 8.8 shows the port 4 register configuration.

**Table 8.8 Port 4 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 4 | PDR4 | R/W | H'F8 | H'FFD7 |
| Port control register 4 | PCR4 | W | H'F8 | H'FFE7 |

#### (1) Port Data Register 4 (PDR4)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | $P4_3$ | $P4_2$ | $P4_1$ | $P4_0$ |
| Initial value | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | R | R/W | R/W | R/W |

PDR4 is an 8-bit register that stores data for port 4 pins $P4_2$ to $P4_0$. If port 4 is read while PCR4 bits are set to 1, the values stored in PDR4 are read, regardless of the actual pin states. If port 4 is read while PCR4 bits are cleared to 0, the pin states are read.

Upon reset, PDR4 is initialized to H'F8.

### (2) Port Control Register 4 (PCR4)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | $PCR4_2$ | $PCR4_1$ | $PCR4_0$ |
| Initial value | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | — | W | W | W |

PCR4 is an 8-bit register for controlling whether each of port 4 pins $P4_2$ to $P4_0$ functions as an input pin or output pin. Setting a PCR4 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. PCR4 and PDR4 settings are valid when the corresponding pins are designated for general-purpose input/output by SCR32.

Upon reset, PCR4 is initialized to H'F8.

PCR4 is a write-only register, which always reads all 1s.

## 8.4.3 Pin Functions

Table 8.9 shows the port 4 pin functions.

**Table 8.9 Port 4 Pin Functions**

| Pin | Pin Functions and Selection Method |
|---|---|
| $P4_3/\overline{IRQ_0}$ | The pin function depends on bit IRQ0 in PMR3. |

| IRQ0 | 0 | 1 |
|---|---|---|
| Pin function | $P4_3$ input pin | $\overline{IRQ_0}$ input pin |

| Pin | Pin Functions and Selection Method |
|---|---|
| $P4_2/TXD_{32}$ | The pin function depends on bit $TE_{32}$ in SCR32, bit SPC32 in SPCR, and bit $PCR4_2$ in PCR4. |

| SPC32 | 0 | | 1 |
|---|---|---|---|
| $TE_{32}$ | 0 | | 1 |
| $PCR4_2$ | 0 | 1 | * |
| Pin function | $P4_2$ input pin | $P4_2$ output pin | $TXD_{32}$ output pin |

| Pin | Pin Functions and Selection Method |
|---|---|
| $P4_1/RXD_{32}$ | The pin function depends on bit $RE_{32}$ in SCR32 and bit $PCR4_1$ in PCR4. |

| $RE_{32}$ | 0 | | 1 |
|---|---|---|---|
| $PCR4_1$ | 0 | 1 | * |
| Pin function | $P4_1$ input pin | $P4_1$ output pin | $RXD_{32}$ input pin |

$P4_0/SCK_{32}$ The pin function depends on bits CKE321 and CKE320 in SCR32, bit COM32 in SMR32, and bit $PCR4_0$ in PCR4.

| CKE321 | 0 | | | | 1 |
|---|---|---|---|---|---|
| CKE320 | 0 | | | 1 | * |
| COM32 | 0 | | 1 | * | * |
| $PCR4_0$ | 0 | 1 | * | | * |
| Pin function | $P4_0$ input pin | $P4_0$ output pin | $SCK_{32}$ output pin | | $SCK_{32}$ input pin |

*: Don't care

### 8.4.4 Pin States

Table 8.10 shows the port 4 pin states in each operating mode.

**Table 8.10 Port 4 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P4_3/\overline{IRQ}_0$<br>$P4_2/TXD_{32}$<br>$P4_1/RXD_{32}$<br>$P4_0/SCK_{32}$ | High-impedance | Retains previous state | Retains previous state | High-impedance | Retains previous state | Functional | Functional |

# 8.5 Port 5

## 8.5.1 Overview

Port 5 is an 8-bit I/O port, configured as shown in figure 8.4.

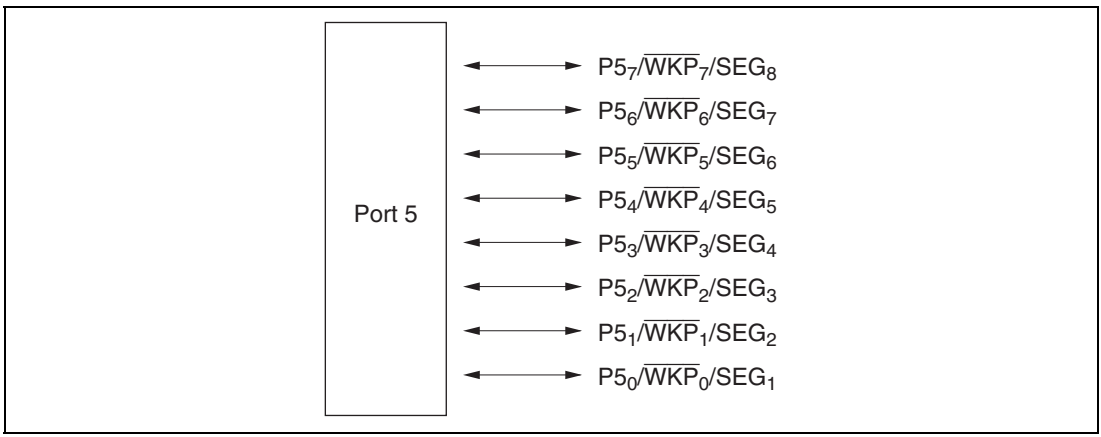



**Figure 8.4 Port 5 Pin Configuration**

## 8.5.2 Register Configuration and Description

Table 8.11 shows the port 5 register configuration.

**Table 8.11 Port 5 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 5 | PDR5 | R/W | H'00 | H'FFD8 |
| Port control register 5 | PCR5 | W | H'00 | H'FFE8 |
| Port pull-up control register 5 | PUCR5 | R/W | H'00 | H'FFE2 |
| Port mode register 5 | PMR5 | R/W | H'00 | H'FFCC |

### (1) Port Data Register 5 (PDR5)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P5_7$ | $P5_6$ | $P5_5$ | $P5_4$ | $P5_3$ | $P5_2$ | $P5_1$ | $P5_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR5 is an 8-bit register that stores data for port 5 pins $P5_7$ to $P5_0$. If port 5 is read while PCR5 bits are set to 1, the values stored in PDR5 are read, regardless of the actual pin states. If port 5 is read while PCR5 bits are cleared to 0, the pin states are read.

Upon reset, PDR5 is initialized to H'00.

### (2) Port Control Register 5 (PCR5)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR5_7$ | $PCR5_6$ | $PCR5_5$ | $PCR5_4$ | $PCR5_3$ | $PCR5_2$ | $PCR5_1$ | $PCR5_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR5 is an 8-bit register for controlling whether each of the port 5 pins $P5_7$ to $P5_0$ functions as an input pin or output pin. Setting a PCR5 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. PCR5 and PDR5 settings are valid when the corresponding pins are designated for general-purpose input/output by PMR5 and bits SGS3 to SGS0 in LPCR.

Upon reset, PCR5 is initialized to H'00.

PCR5 is a write-only register, which is always read as all 1s.

### (3) Port Pull-Up Control Register 5 (PUCR5)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PUCR5_7$ | $PUCR5_6$ | $PUCR5_5$ | $PUCR5_4$ | $PUCR5_3$ | $PUCR5_2$ | $PUCR5_1$ | $PUCR5_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PUCR5 controls whether the MOS pull-up of each of port 5 pins $P5_7$ to $P5_0$ is on or off. When a PCR5 bit is cleared to 0, setting the corresponding PUCR5 bit to 1 turns on the MOS pull-up for the corresponding pin, while clearing the bit to 0 turns off the MOS pull-up.

Upon reset, PUCR5 is initialized to H'00.

### (4) Port Mode Register 5 (PMR5)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $WKP_7$ | $WKP_6$ | $WKP_5$ | $WKP_4$ | $WKP_3$ | $WKP_2$ | $WKP_1$ | $WKP_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PMR5 is an 8-bit read/write register, controlling the selection of pin functions for port 5 pins.

Upon reset, PMR5 is initialized to H'00.

**Bit n:** $P5_n/\overline{WKP}_n/SEG_{n+1}$ pin function switch (WKPn)

When pin P5n/$\overline{WKP}$n/SEGn+1 is not used as $SEG_{n+1}$, these bits select whether the pin is used as P5n or $\overline{WKP}_n$.

| Bit n<br>WKPn | Description | |
|---|---|---|
| 0 | Functions as P5n I/O pin | (initial value) |
| 1 | Functions as $\overline{WKP}_n$ input pin | |

(n = 7 to 0)

Note: For use as $SEG_{n+1}$, see section 13.2.1, LCD Port Control Register (LPCR).

### 8.5.3 Pin Functions

Table 8.12 shows the port 5 pin functions.

**Table 8.12 Port 5 Pin Functions**

| Pin | Pin Functions and Selection Method |
|---|---|
| $P5_7/\overline{WKP}_7/SEG_8$ to $P5_0/\overline{WKP}_0/SEG_1$ | The pin function depends on bit $WKP_n$ in PMR5, bit $PCR5_n$ in PCR5, and bits SGS3 to SGS0 in LPCR. |

(n = 7 to 0)

| SGS3 to SGS0 | 0*** | | | 1*** |
|---|---|---|---|---|
| $WKP_n$ | 0 | | 1 | * |
| $PCR5_n$ | 0 | 1 | * | * |
| Pin function | $P5_n$ input pin | $P5_n$ output pin | $\overline{WKP}$n input pin | SEGn+1 output pin |

*: Don't care

### 8.5.4 Pin States

Table 8.13 shows the port 5 pin states in each operating mode.

**Table 8.13 Port 5 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P5_7/\overline{WKP}_7/SEG_8$ to $P5_0/\overline{WKP}_0/SEG_1$ | High-impedance | Retains previous state | Retains previous state | High-impedance* | Retains previous state | Functional | Functional |

Note: * A high-level signal is output when the MOS pull-up is in the on state.

### 8.5.5 MOS Input Pull-Up

Port 5 has a built-in MOS input pull-up function that can be controlled by software. When a PCR5 bit is cleared to 0, setting the corresponding PUCR5 bit to 1 turns on the MOS pull-up for that pin. The MOS pull-up function is in the off state after a reset.

| | | | |
|---|---|---|---|
| $PCR5_n$ | 0 | 0 | 1 |
| $PUCR5_n$ | 0 | 1 | * |
| MOS input pull-up | Off | On | Off |

(n = 7 to 0)

*: Don't care

## 8.6 Port 6

### 8.6.1 Overview

Port 6 is an 8-bit I/O port. The port 6 pin configuration is shown in figure 8.5.

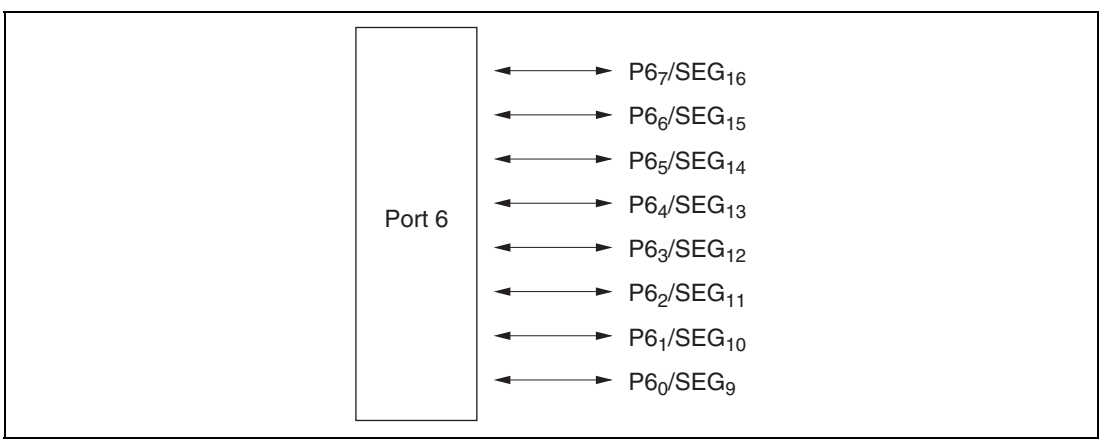



**Figure 8.5 Port 6 Pin Configuration**

### 8.6.2 Register Configuration and Description

Table 8.14 shows the port 6 register configuration.

**Table 8.14 Port 6 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 6 | PDR6 | R/W | H'00 | H'FFD9 |
| Port control register 6 | PCR6 | W | H'00 | H'FFE9 |
| Port pull-up control register 6 | PUCR6 | R/W | H'00 | H'FFE3 |

#### (1) Port Data Register 6 (PDR6)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P6_7$ | $P6_6$ | $P6_5$ | $P6_4$ | $P6_3$ | $P6_2$ | $P6_1$ | $P6_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR6 is an 8-bit register that stores data for port 6 pins $P6_7$ to $P6_0$.

If port 6 is read while PCR6 bits are set to 1, the values stored in PDR6 are read, regardless of the actual pin states. If port 6 is read while PCR6 bits are cleared to 0, the pin states are read.

Upon reset, PDR6 is initialized to H'00.

#### (2) Port Control Register 6 (PCR6)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR6_7$ | $PCR6_6$ | $PCR6_5$ | $PCR6_4$ | $PCR6_3$ | $PCR6_2$ | $PCR6_1$ | $PCR6_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR6 is an 8-bit register for controlling whether each of the port 6 pins $P6_7$ to $P6_0$ functions as an input pin or output pin.

Setting a PCR6 bit to 1 makes the corresponding pin ($P6_7$ to $P6_0$) an output pin, while clearing the bit to 0 makes the pin an input pin. PCR6 and PDR6 settings are valid when the corresponding pins are designated for general-purpose input/output by bits SGS3 to SGS0 in LPCR.

Upon reset, PCR6 is initialized to H'00.

PCR6 is a write-only register, which always reads all 1s.

### (3) Port Pull-Up Control Register 6 (PUCR6)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PUCR6_7$ | $PUCR6_6$ | $PUCR6_5$ | $PUCR6_4$ | $PUCR6_3$ | $PUCR6_2$ | $PUCR6_1$ | $PUCR6_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PUCR6 controls whether the MOS pull-up of each of the port 6 pins $P6_7$ to $P6_0$ is on or off. When a PCR6 bit is cleared to 0, setting the corresponding PUCR6 bit to 1 turns on the MOS pull-up for the corresponding pin, while clearing the bit to 0 turns off the MOS pull-up.

Upon reset, PUCR6 is initialized to H'00.

## 8.6.3 Pin Functions

Table 8.15 shows the port 6 pin functions.

**Table 8.15 Port 6 Pin Functions**

| Pin | Pin Functions and Selection Method | | | |
|---|---|---|---|---|
| $P6_7/SEG_{16}$ to $P6_0/SEG_9$ | The pin function depends on bit PCR6n in PCR6 and bits SGS3 to SGS0 in LPCR. (n = 7 to 0) | | | |
| | SGS3 to SGS0 | 00**, 010* | | 011**, 1*** |
| | $PCR6_n$ | 0 | 1 | * |
| | Pin function | $P6_n$ input pin | $P6_n$ output pin | $SEG_{n+9}$ output pin |

*: Don't care

### 8.6.4 Pin States

Table 8.16 shows the port 6 pin states in each operating mode.

**Table 8.16 Port 6 Pin States**

| Pin | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P6_7/SEG_{16}$ to $P6_0/SEG_9$ | High-impedance | Retains previous state | Retains previous state | High-impedance* | Retains previous state | Functional | Functional |

Note: * A high-level signal is output when the MOS pull-up is in the on state.

### 8.6.5 MOS Input Pull-Up

Port 6 has a built-in MOS pull-up function that can be controlled by software. When a PCR6 bit is cleared to 0, setting the corresponding PUCR6 bit to 1 turns on the MOS pull-up for that pin. The MOS pull-up function is in the off state after a reset.

| | | | |
|---|---|---|---|
| $PCR6_n$ | 0 | 0 | 1 |
| $PUCR6_n$ | 0 | 1 | * |
| MOS input pull-up | Off | On | Off |

(n = 7 to 0)

*: Don't care

# 8.7 Port 7

## 8.7.1 Overview

Port 7 is a 8-bit I/O port, configured as shown in figure 8.6.

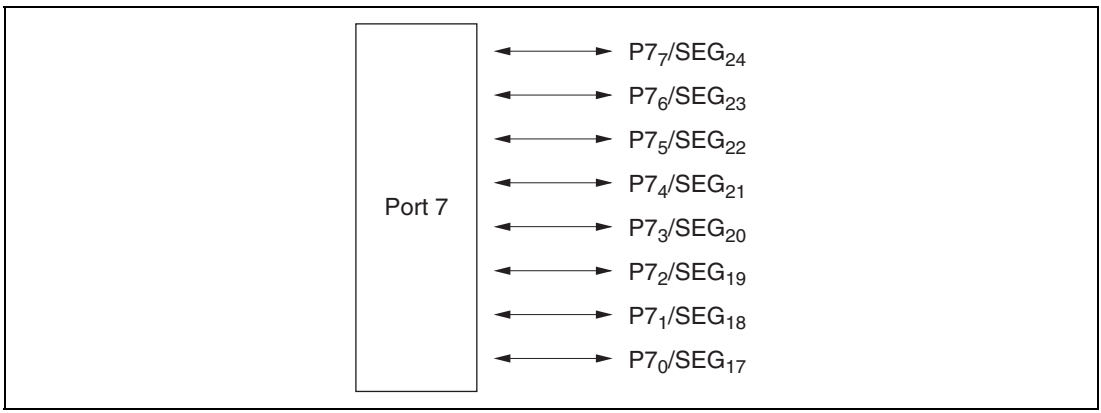



**Figure 8.6 Port 7 Pin Configuration**

## 8.7.2 Register Configuration and Description

Table 8.17 shows the port 7 register configuration.

**Table 8.17 Port 7 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 7 | PDR7 | R/W | H'00 | H'FFDA |
| Port control register 7 | PCR7 | W | H'00 | H'FFEA |

### (1) Port Data Register 7 (PDR7)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P7_7$ | $P7_6$ | $P7_5$ | $P7_4$ | $P7_3$ | $P7_2$ | $P7_1$ | $P7_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR7 is an 8-bit register that stores data for port 7 pins $P7_7$ to $P7_0$. If port 7 is read while PCR7 bits are set to 1, the values stored in PDR7 are read, regardless of the actual pin states. If port 7 is read while PCR7 bits are cleared to 0, the pin states are read.

Upon reset, PDR7 is initialized to H'00.

### (2) Port Control Register 7 (PCR7)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR7_7$ | $PCR7_6$ | $PCR7_5$ | $PCR7_4$ | $PCR7_3$ | $PCR7_2$ | $PCR7_1$ | $PCR7_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR7 is an 8-bit register for controlling whether each of the port 7 pins $P7_7$ to $P7_0$ functions as an input pin or output pin. Setting a PCR7 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. PCR7 and PDR7 settings are valid when the corresponding pins are designated for general-purpose input/output by bits SGS3 to SGS0 in LPCR.

Upon reset, PCR7 is initialized to H'00.

PCR7 is a write-only register, which always reads as all 1s.

### 8.7.3 Pin Functions

Table 8.18 shows the port 7 pin functions.

**Table 8.18 Port 7 Pin Functions**

| Pin | Pin Functions and Selection Method |
|---|---|
| $P7_7/SEG_{24}$ to $P7_0/SEG_{17}$ | The pin function depends on bit $PCR7_n$ in PCR7 and bits SGS3 to SGS0 in LPCR. (n = 7 to 0) |

| SGS3 to SGS0 | 00** | | 01**, 1*** |
|---|---|---|---|
| $PCR7_n$ | 0 | 1 | * |
| Pin function | $P7_n$ input pin | $P7_n$ output pin | $SEG_{n+17}$ output pin |

*: Don't care

### 8.7.4 Pin States

Table 8.19 shows the port 7 pin states in each operating mode.

**Table 8.19 Port 7 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P7_7/SEG_{24}$ to $P7_0/SEG_{17}$ | High-impedance | Retains previous state | Retains previous state | High-impedance | Retains previous state | Functional | Functional |

# 8.8 Port 8

## 8.8.1 Overview

Port 8 is an 8-bit I/O port configured as shown in figure 8.7.

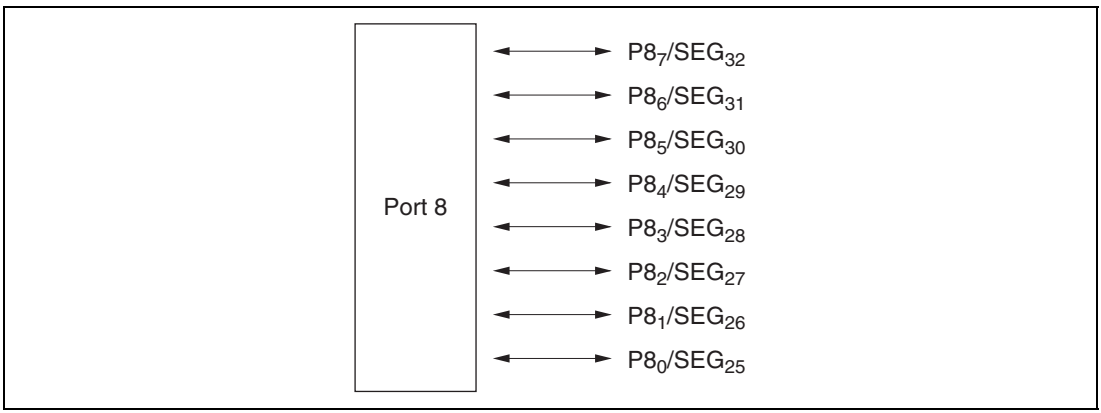



**Figure 8.7 Port 8 Pin Configuration**

## 8.8.2 Register Configuration and Description

Table 8.20 shows the port 8 register configuration.

**Table 8.20 Port 8 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register 8 | PDR8 | R/W | H'00 | H'FFDB |
| Port control register 8 | PCR8 | W | H'00 | H'FFEB |

### (1) Port Data Register 8 (PDR8)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P8_7$ | $P8_6$ | $P8_5$ | $P8_4$ | $P8_3$ | $P8_2$ | $P8_1$ | $P8_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

PDR8 is an 8-bit register that stores data for port 8 pins $P8_7$ to $P8_0$. If port 8 is read while PCR8 bits are set to 1, the values stored in PDR8 are read, regardless of the actual pin states. If port 8 is read while PCR8 bits are cleared to 0, the pin states are read.

Upon reset, PDR8 is initialized to H'00.

### (2) Port Control Register 8 (PCR8)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PCR8_7$ | $PCR8_6$ | $PCR8_5$ | $PCR8_4$ | $PCR8_3$ | $PCR8_2$ | $PCR8_1$ | $PCR8_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PCR8 is an 8-bit register for controlling whether each of the port 8 pins $P8_7$ to $P8_0$ functions as an input or output pin. Setting a PCR8 bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. PCR8 and PDR8 settings are valid when the corresponding pins are designated for general-purpose input/output by bits SGS3 to SGS0 in LPCR.

Upon reset, PCR8 is initialized to H'00.

PCR8 is a write-only register, which is always read as all 1s.

### 8.8.3 Pin Functions

Table 8.21 shows the port 8 pin functions.

**Table 8.21 Port 8 Pin Functions**

| Pin | Pin Functions and Selection Method | | | |
|---|---|---|---|---|
| $P8_7/SEG_{32}$ | The pin function depends on bit $PCR8_7$ in PCR8 and bits SGS3 to SGS0 in LPCR. | | | |
| | SGS3 to SGS0 | 000* | | 001*, 01**, 1*** |
| | $PCR8_7$ | 0 | 1 | * |
| | Pin function | $P8_7$ input pin | $P8_7$ output pin | $SEG_{32}$ output pin |
| $P8_6/SEG_{31}$ | The pin function depends on bit $PCR8_6$ in PCR8 and bits SGS3 to SGS0 in LPCR. | | | |
| | SGS3 to SGS0 | 000* | | 001*, 01**, 1*** |
| | $PCR8_6$ | 0 | 1 | * |
| | Pin function | $P8_6$ input pin | $P8_6$ output pin | $SEG_{31}$ output pin |
| $P8_5/SEG_{30}$ | The pin function depends on bit $PCR8_5$ in PCR8 and bits SGS3 to SGS0 in LPCR. | | | |
| | SGS3 to SGS0 | 000* | | 001*, 01**, 1*** |
| | $PCR8_5$ | 0 | 1 | * |
| | Pin function | $P8_5$ input pin | $P8_5$ output pin | $SEG_{30}$ output pin |
| $P8_4/SEG_{29}$ | The pin function depends on bit $PCR8_4$ in PCR8 and bits SGS3 to SGS0 in LPCR. | | | |
| | SGS3 to SGS0 | 000* | | 001*, 01**, 1*** |
| | $PCR8_4$ | 0 | 1 | * |
| | Pin function | $P8_4$ input pin | $P8_4$ output pin | $SEG_{29}$ output pin |
| $P8_3/SEG_{28}$ to $P8_0/SEG_{25}$ | The pin function depends on bit $PCR8_n$ in PCR8 and bits SGS3 to SGS0 in LPCR. (n = 3 to 0) | | | |
| | SGS3 to SGS0 | 000* | | 001*, 01**, 1*** |
| | $PCR8_n$ | 0 | 1 | * |
| | Pin function | $P8_n$ input pin | $P8_n$ output pin | $SEG_{n+25}$ output pin |

*: Don't care

### 8.8.4 Pin States

Table 8.22 shows the port 8 pin states in each operating mode.

**Table 8.22 Port 8 Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P8_7/SEG_{32}$<br>$P8_6/SEG_{31}$<br>$P8_5/SEG_{30}$<br>$P8_4/SEG_{29}$<br>$P8_3/SEG_{28}$ to<br>$P8_0/SEG_{25}$ | High-impedance | Retains previous state | Retains previous state | High-impedance | Retains previous state | Functional | Functional |

## 8.9 Port A

### 8.9.1 Overview

Port A is a 4-bit I/O port, configured as shown in figure 8.8.

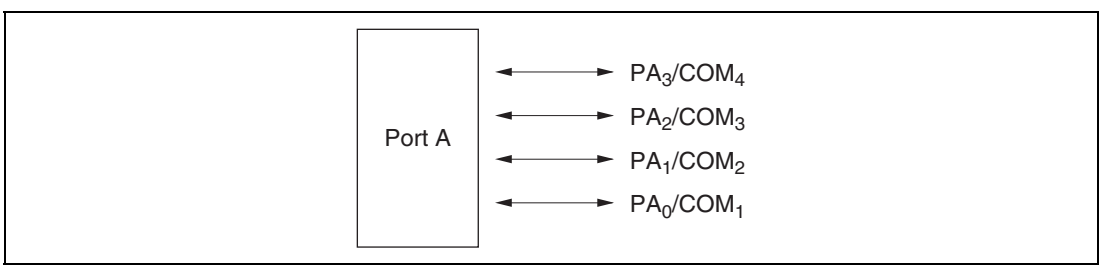



**Figure 8.8 Port A Pin Configuration**

### 8.9.2 Register Configuration and Description

Table 8.23 shows the port A register configuration.

**Table 8.23 Port A Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register A | PDRA | R/W | H'F0 | H'FFDD |
| Port control register A | PCRA | W | H'F0 | H'FFED |

#### (1) Port Data Register A (PDRA)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | $PA_3$ | $PA_2$ | $PA_1$ | $PA_0$ |
| Initial value | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

PDRA is an 8-bit register that stores data for port A pins $PA_3$ to $PA_0$. If port A is read while PCRA bits are set to 1, the values stored in PDRA are read, regardless of the actual pin states. If port A is read while PCRA bits are cleared to 0, the pin states are read.

Upon reset, PDRA is initialized to H'F0.

#### (2) Port Control Register A (PCRA)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | $PCRA_3$ | $PCRA_2$ | $PCRA_1$ | $PCRA_0$ |
| Initial value | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

PCRA controls whether each of port A pins $PA_3$ to $PA_0$ functions as an input pin or output pin. Setting a PCRA bit to 1 makes the corresponding pin an output pin, while clearing the bit to 0 makes the pin an input pin. PCRA and PDRA settings are valid when the corresponding pins are designated for general-purpose input/output by LPCR.

Upon reset, PCRA is initialized to H'F0.

PCRA is a write-only register, which always reads all 1s.

### 8.9.3 Pin Functions

Table 8.24 shows the port A pin functions.

**Table 8.24 Port A Pin Functions**

| Pin | Pin Functions and Selection Method | | | |
|---|---|---|---|---|
| $PA_3/COM_4$ | The pin function depends on bit $PCRA_3$ in PCRA and bits SGS3 to SGS0. | | | |
| | SGS3 to SGS0 | 0000 | 0000 | Not 0000 |
| | $PCRA_3$ | 0 | 1 | * |
| | Pin function | $PA_3$ input pin | $PA_3$ output pin | $COM_4$ output pin |
| $PA_2/COM_3$ | The pin function depends on bit $PCRA_2$ in PCRA and bits SGS3 to SGS0. | | | |
| | SGS3 to SGS0 | 0000 | 0000 | Not 0000 |
| | $PCRA_2$ | 0 | 1 | * |
| | Pin function | $PA_2$ input pin | $PA_2$ output pin | $COM_3$ output pin |
| $PA_1/COM_2$ | The pin function depends on bit $PCRA_1$ in PCRA and bits SGS3 to SGS0. | | | |
| | SGS3 to SGS0 | 0000 | 0000 | Not 0000 |
| | $PCRA_1$ | 0 | 1 | * |
| | Pin function | $PA_1$ input pin | $PA_1$ output pin | $COM_2$ output pin |
| $PA_0/COM_1$ | The pin function depends on bit $PCRA_0$ in PCRA and bits SGS3 to SGS0. | | | |
| | SGS3 to SGS0 | 0000 | | Not 0000 |
| | $PCRA_0$ | 0 | 1 | * |
| | Pin function | $PA_0$ input pin | $PA_0$ output pin | $COM_1$ output pin |

*: Don't care

### 8.9.4 Pin States

Table 8.25 shows the port A pin states in each operating mode.

**Table 8.25 Port A Pin States**

| Pins | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $PA_3/COM_4$<br>$PA_2/COM_3$<br>$PA_1/COM_2$<br>$PA_0/COM_1$ | High-impedance | Retains previous state | Retains previous state | High-impedance | Retains previous state | Functional | Functional |

## 8.10 Port B

### 8.10.1 Overview

Port B is an 8-bit input-only port, configured as shown in figure 8.9.

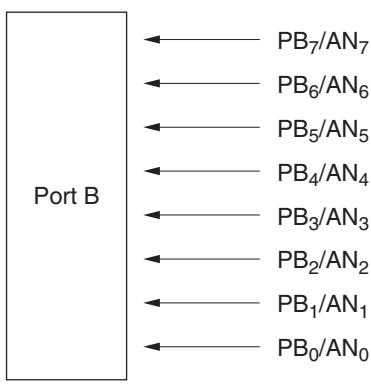



**Figure 8.9 Port B Pin Configuration**

### 8.10.2 Register Configuration and Description

Table 8.26 shows the port B register configuration.

**Table 8.26 Port B Register**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Port data register B | PDRB | R | Undefined | H'FFDE |

**Port Data Register B (PDRB)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PB_7$ | $PB_6$ | $PB_5$ | $PB_4$ | $PB_3$ | $PB_2$ | $PB_1$ | $PB_0$ |
| Read/Write | R | R | R | R | R | R | R | R |

Reading PDRB always gives the pin states. However, if a port B pin is selected as an analog input channel for the A/D converter by AMR bits CH3 to CH0, that pin reads 0 regardless of the input voltage.

## 8.11 Input/Output Data Inversion Function

### 8.11.1 Overview

With input pins $\overline{WKP}_0$ to $\overline{WKP}_7$, $RXD_{31}$, and $RXD_{32}$, and output pins $TXD_{31}$ and $TXD_{32}$, the data can be handled in inverted form.

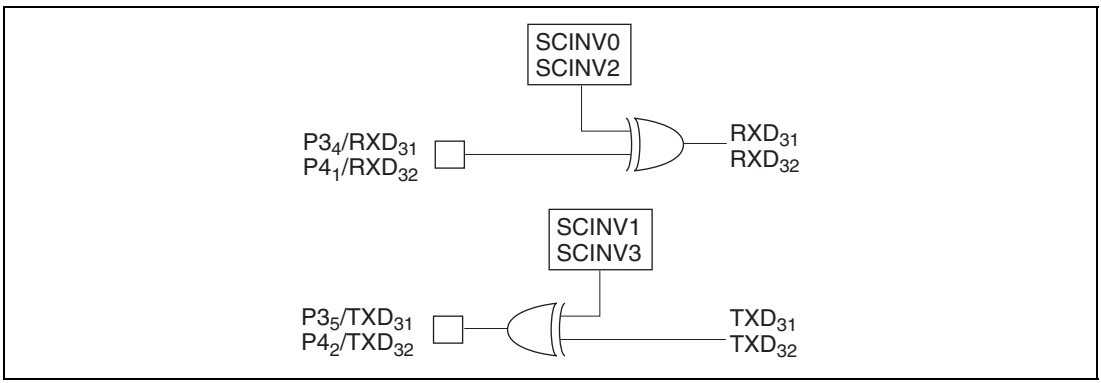



**Figure 8.10 Input/Output Data Inversion Function**

### 8.11.2 Register Configuration and Descriptions

Table 8.27 shows the registers used by the input/output data inversion function.

**Table 8.27 Register Configuration**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Serial port control register | SPCR | R/W | H'C0 | H'FF91 |

### Serial Port Control Register (SPCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | SPC32 | SPC31 | SCINV3 | SCINV2 | SCINV1 | SCINV0 |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | R/W | R/W | R/W | R/W | R/W | R/W |

SPCR is an 8-bit readable/writable register that performs $RXD_{31}$, $RXD_{32}$, $TXD_{31}$, and $TXD_{32}$ pin input/output data inversion switching. SPCR is initialized to H'C0 by a reset.

**Bits 7 and 6:** Reserved bits

Bits 7 and 6 are reserved; they are always read as 1 and cannot be modified.

**Bit 5:** $P4_2$/$TXD_{32}$ pin function switch (SPC32)

This bit selects whether pin $P4_2$/$TXD_{32}$ is used as $P4_2$ or as $TXD_{32}$.

| Bit 5<br>SPC32 | Description | |
|---|---|---|
| 0 | Functions as $P4_2$ I/O pin | (initial value) |
| 1 | Functions as $TXD_{32}$ output pin* | |

Note: * Set the TE bit in SCR3 after setting this bit to 1.

**Bit 4:** $P3_5$/$TXD_{31}$ pin function switch (SPC31)

This bit selects whether pin $P3_5$/$TXD_{31}$ is used as $P3_5$ or as $TXD_{31}$.

| Bit 4<br>SPC31 | Description | |
|---|---|---|
| 0 | Functions as $P3_5$ I/O pin | (initial value) |
| 1 | Functions as $TXD_{31}$ output pin* | |

Note: * Set the TE bit in SCR3 after setting this bit to 1.

**Bit 3:** $TXD_{32}$ pin output data inversion switch

Bit 3 specifies whether or not $TXD_{32}$ pin output data is to be inverted.

| Bit 3<br>SCINV3 | Description | |
|---|---|---|
| 0 | $TXD_{32}$ output data is not inverted | (initial value) |
| 1 | $TXD_{32}$ output data is inverted | |

**Bit 2:** $RXD_{32}$ pin input data inversion switch

Bit 2 specifies whether or not $RXD_{32}$ pin input data is to be inverted.

| Bit 2<br>SCINV2 | Description | |
|---|---|---|
| 0 | $RXD_{32}$ input data is not inverted | (initial value) |
| 1 | $RXD_{32}$ input data is inverted | |

**Bit 1:** $TXD_{31}$ pin output data inversion switch

Bit 1 specifies whether or not $TXD_{31}$ pin output data is to be inverted.

| Bit 1<br>SCINV1 | Description | |
|---|---|---|
| 0 | $TXD_{31}$ output data is not inverted | (initial value) |
| 1 | $TXD_{31}$ output data is inverted | |

**Bit 0:** $RXD_{31}$ pin input data inversion switch

Bit 0 specifies whether or not $RXD_{31}$ pin input data is to be inverted.

| Bit 0<br>SCINV0 | Description | |
|---|---|---|
| 0 | $RXD_{31}$ input data is not inverted | (initial value) |
| 1 | $RXD_{31}$ input data is inverted | |

### 8.11.3 Note on Modification of Serial Port Control Register

When a serial port control register is modified, the data being input or output up to that point is inverted immediately after the modification, and an invalid data change is input or output. When modifying a serial port control register, do so in a state in which data changes are invalidated.

## 8.12 Application Note

### 8.12.1 The Management of the Un-Use Terminal

If an I/O pin not used by the user system is floating, pull it up or down.

- If an unused pin is an input pin, handle it in one of the following ways:
  - Pull it up to $V_{CC}$ with an on-chip pull-up MOS.
  - Pull it up to $V_{CC}$ with an external resistor of approximately 100 kΩ.
  - Pull it down to $V_{SS}$ with an external resistor of approximately 100 kΩ.
  - For a pin also used by the A/D converter, pull it up to $AV_{CC}$.
- If an unused pin is an output pin, handle it in one of the following ways:
  - Set the output of the unused pin to high and pull it up to $V_{CC}$ with an external resistor of approximately 100 kΩ.
  - Set the output of the unused pin to low and pull it down to $V_{SS}$ with an external resistor of approximately 100 kΩ.

# Section 9 Timers

## 9.1 Overview

This LSI provides six timers: timers A, C, F, G, and a watchdog timer, and an asynchronous event counter. The functions of these timers are outlined in table 9.1.

**Table 9.1 Timer Functions**

| Name | Functions | Internal Clock | Event Input Pin | Waveform Output Pin | Remarks |
|---|---|---|---|---|---|
| Timer A | • 8-bit interval timer<br>• Interval function | $\phi$/8 to $\phi$/8192 (8 choices) | — | — | |
| | • Time base | $\phi_w$/128 (choice of 4 overflow periods) | | | |
| | • Clock output | $\phi$/4 to $\phi$/32<br>$\phi_w$, $\phi_w$/4 to $\phi_w$/32<br>(9 choices) | — | TMOW | |
| Timer C | • 8-bit timer<br>• Interval function<br>• Event counting function<br>• Up-count/down-count selectable | $\phi$/4 to $\phi$/8192, $\phi_w$/4 (7 choices) | TMIC | — | Up-count/down-count controllable by software or hardware |
| Timer F | 16-bit timer<br>Event counting function Also usable as two independent 8-bit timers Output compare output function | $\phi$/4 to $\phi$/32, $\phi_w$/4 (4 choices) | TMIF | TMOFL<br>TMOFH | |
| Timer G | • 8-bit timer<br>• Input capture function<br>• Interval function | $\phi$/2 to $\phi$/64, $\phi_w$/4 (4 choices) | TMIG | — | • Counter clearing option<br>• Built-in capture input signal noise canceler |

| Name | Functions | Internal Clock | Event Input Pin | Waveform Output Pin | Remarks |
|---|---|---|---|---|---|
| Watchdog timer | • Reset signal generated when8-bit counter overflows | $\phi$/8192<br>$\phi_W$/32 | — | — | |
| Asynchro-nous event counter | • 16-bit counter<br>• Also usable as two independent 8-bit counters<br>• Counts events asynchronous to $\phi$ and $\phi$w | — | AEVL<br>AEVH | — | |

## 9.2 Timer A

### 9.2.1 Overview

Timer A is an 8-bit timer with interval timing and real-time clock time-base functions. The clock time-base function is available when a 32.768-kHz crystal oscillator is connected. A clock signal divided from 32.768 kHz, from 38.4 kHz (if a 38.4 kHz crystal oscillator is connected), or from the system clock, can be output at the TMOW pin.

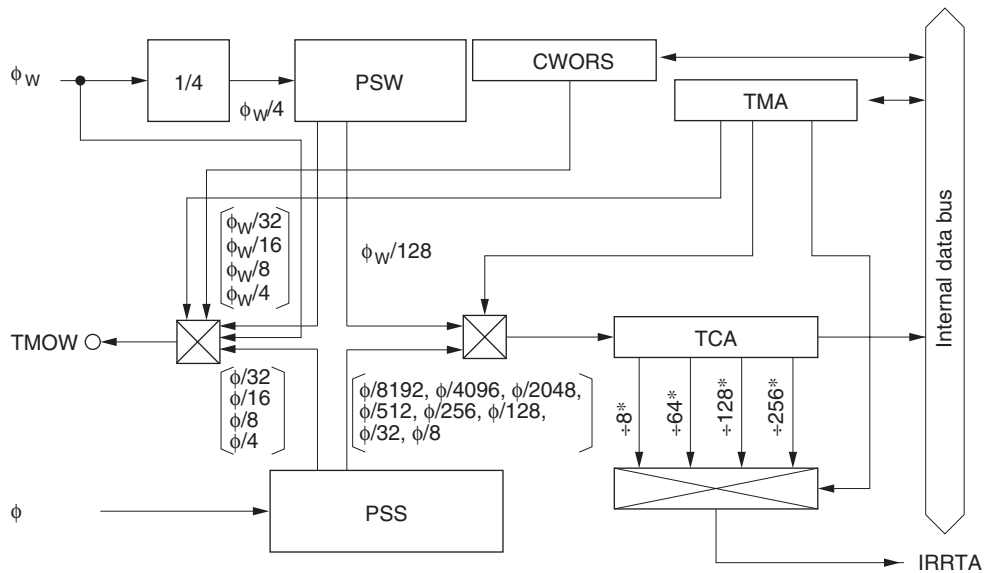
#### (1) Features

Features of timer A are given below.

- Choice of eight internal clock sources (ϕ/8192, ϕ/4096, ϕ/2048, ϕ/512, ϕ/256, ϕ/128, ϕ/32, ϕ/8).
- Choice of four overflow periods (1 s, 0.5 s, 0.25 s, 31.25 ms) when timer A is used as a clock time base (using a 32.768 kHz crystal oscillator).
- An interrupt is requested when the counter overflows.
- Any of nine clock signals can be output at the TMOW pin: 32.768 kHz divided by 32, 16, 8, or 4 (1 kHz, 2 kHz, 4 kHz, 8 kHz, 32,768 kHz) or 38.4 kHz divided by 32, 16, 8, or 4 (1.2 kHz, 2.4 kHz, 4.8 kHz, 9.6 kHz, 38.4 kHz), and the system clock divided by 32, 16, 8, or 4.
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### (2) Block Diagram

Figure 9.1 shows a block diagram of timer A.




Legend:

| | |
|---|---|
| TMA: | Timer mode register A |
| TCA: | Timer counter A |
| IRRTA: | Timer A overflow interrupt request flag |
| PSW: | Prescaler W |
| PSS: | Prescaler S |
| CWOSR: | Subclock output select register |

Note: * Can be selected only when the prescaler W output ($\phi_W$/128) is used as the TCA input clock.

**Figure 9.1 Block Diagram of Timer A**

### (3) Pin Configuration

Table 9.2 shows the timer A pin configuration.

**Table 9.2 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Clock output | TMOW | Output | Output of waveform generated by timer A output circuit |

### (4) Register Configuration

Table 9.3 shows the register configuration of timer A.

**Table 9.3 Timer A Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer mode register A | TMA | R/W | H'10 | H'FFB0 |
| Timer counter A | TCA | R | H'00 | H'FFB1 |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |
| Subclock output select register | CWOSR | R/W | H'FE | H'FF92 |

## 9.2.2 Register Descriptions

### (1) Timer Mode Register A (TMA)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMA7 | TMA6 | TMA5 | — | TMA3 | TMA2 | TMA1 | TMA0 |
| Initial value | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | — | R/W | R/W | R/W | R/W |

TMA is an 8-bit read/write register for selecting the prescaler, input clock, and output clock.

Upon reset, TMA is initialized to H'10.

**Bits 7 to 5:** Clock output select (TMA7 to TMA5)

Bits 7 to 5 choose which of eight clock signals is output at the TMOW pin. The system clock divided by 32, 16, 8, or 4 can be output in active mode and sleep mode. A 32.768 kHz or 38.4 kHz signal divided by 32, 16, 8, or 4 can be output in active mode, sleep mode, and subactive mode. $\phi_w$ is output in all modes except the reset state.

| CWOSR | TMA | | | |
|---|---|---|---|---|
| **CWOS** | **Bit 7 TMA7** | **Bit 6 TMA6** | **Bit 5 TMA5** | **Clock Output** |
| 0 | 0 | 0 | 0 | $\phi/32$ (initial value) |
| | | | 1 | $\phi/16$ |
| | | 1 | 0 | $\phi/8$ |
| | | | 1 | $\phi/4$ |
| | 1 | 0 | 0 | $\phi_w/32$ |
| | | | 1 | $\phi_w/16$ |
| | | 1 | 0 | $\phi_w/8$ |
| | | | 1 | $\phi_w/4$ |
| 1 | * | * | * | $\phi_w$ |

*: Don't care

**Bit 4:** Reserved bit

Bit 4 is reserved; it is always read as 1, and cannot be modified.

**Bits 3 to 0:** Internal clock select (TMA3 to TMA0)

Bits 3 to 0 select the clock input to TCA. The selection is made as follows.

| Bit 3 TMA3 | Bit 2 TMA2 | Bit 1 TMA1 | Bit 0 TMA0 | Description: Prescaler and Divider Ratio or Overflow Period | Function |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | PSS, ϕ/8192 (initial value) | Interval timer |
| | | | 1 | PSS, ϕ/4096 | |
| | | 1 | 0 | PSS, ϕ/2048 | |
| | | | 1 | PSS, ϕ/512 | |
| | 1 | 0 | 0 | PSS, ϕ/256 | |
| | | | 1 | PSS, ϕ/128 | |
| | | 1 | 0 | PSS, ϕ/32 | |
| | | | 1 | PSS, ϕ/8 | |
| 1 | 0 | 0 | 0 | PSW, 1 s | Clock time base (when using 32.768 kHz) |
| | | | 1 | PSW, 0.5 s | |
| | | 1 | 0 | PSW, 0.25 s | |
| | | | 1 | PSW, 0.03125 s | |
| | 1 | 0 | 0 | PSW and TCA are reset | |
| | | | 1 | | |
| | | 1 | 0 | | |
| | | | 1 | | |

### (2) Timer Counter A (TCA)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCA7 | TCA6 | TCA5 | TCA4 | TCA3 | TCA2 | TCA1 | TCA0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

TCA is an 8-bit read-only up-counter, which is incremented by internal clock input. The clock source for input to this counter is selected by bits TMA3 to TMA0 in timer mode register A (TMA). TCA values can be read by the CPU in active mode, but cannot be read in subactive mode. When TCA overflows, the IRRTA bit in interrupt request register 1 (IRR1) is set to 1.

TCA is cleared by setting bits TMA3 and TMA2 of TMA to 11.

Upon reset, TCA is initialized to H'00.

### (3) Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to timer A is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 0:** Timer A module standby mode control (TACKSTP)

Bit 0 controls setting and clearing of module standby mode for timer A.

| TACKSTP | Description | |
|---|---|---|
| 0 | Timer A is set to module standby mode | |
| 1 | Timer A module standby mode is cleared | (initial value) |

### (4) Subclock Output Select Register (CWOSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CWOS |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Read/Write | R | R | R | R | R | R | R | R/W |

CWOSR is an 8-bit read/write register that selects the clock to be output from the TMOW pin.

CWOSR is initialized to H'FE by a reset.

**Bits 7 to 1:** Reserved bits

Bits 7 to 1 are reserved; they are always read as 1 and cannot be modified.

**Bit 0:** TMOW pin clock select (CWOS)

Bit 0 selects the clock to be output from the TMOW pin.

| Bit 0<br>CWOS | Description | |
|---|---|---|
| 0 | Clock output from timer A is output (see TMA) | (initial value) |
| 1 | $\phi_w$ is output | |

### 9.2.3 Timer Operation

#### (1) Interval Timer Operation

When bit TMA3 in timer mode register A (TMA) is cleared to 0, timer A functions as an 8-bit interval timer.

Upon reset, TCA is cleared to H'00 and bit TMA3 is cleared to 0, so up-counting and interval timing resume immediately. The clock input to timer A is selected by bits TMA2 to TMA0 in TMA; any of eight internal clock signals output by prescaler S can be selected.

After the count value in TCA reaches H'FF, the next clock signal input causes timer A to overflow, setting bit IRRTA to 1 in interrupt request register 1 (IRR1). If IENTA = 1 in interrupt enable register 1 (IENR1), a CPU interrupt is requested.*

At overflow, TCA returns to H'00 and starts counting up again. In this mode timer A functions as an interval timer that generates an overflow output at intervals of 256 input clock pulses.

Note: * For details on interrupts, see section 3.3, Interrupts.

#### (2) Real-Time Clock Time Base Operation

When bit TMA3 in TMA is set to 1, timer A functions as a real-time clock time base by counting clock signals output by prescaler W. The overflow period of timer A is set by bits TMA1 and TMA0 in TMA. A choice of four periods is available. In time base operation (TMA3 = 1), setting bit TMA2 to 1 clears both TCA and prescaler W to their initial values of H'00.

### (3) Clock Output

Setting bit TMOW in port mode register 1 (PMR1) to 1 causes a clock signal to be output at pin TMOW. Nine different clock output signals can be selected by means of bits TMA7 to TMA5 in TMA and bit CWOS in CWOSR. The system clock divided by 32, 16, 8, or 4 can be output in active mode and sleep mode. A 32.768 kHz or 38.4 kHz signal divided by 32, 16, 8, or 4 can be output in active mode, sleep mode, watch mode, subactive mode, and subsleep mode. The 32.768 kHz or 38.4 kHz clock is output in all modes except the reset state.

## 9.2.4 Timer A Operation States

Table 9.4 summarizes the timer A operation states.

**Table 9.4 Timer A Operation States**

| Operation Mode | | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|---|
| TCA | Interval | Reset | Functions | Functions | Halted | Halted | Halted | Halted | Halted |
| | Clock time base | Reset | Functions | Functions | Functions | Functions | Functions | Halted | Halted |
| TMA | CWOSR | Reset | Functions | Retained | Retained | Functions | Retained | Retained | Retained |

Note: When the real-time clock time base function is selected as the internal clock of TCA in active mode or sleep mode, the internal clock is not synchronous with the system clock, so it is synchronized by a synchronizing circuit. This may result in a maximum error of 1/ϕ (s) in the count cycle.

## 9.2.5 Application Note

When bit 0 (TACKSTP) of the clock stop register 1 (CKSTPR1) is cleared to 0, bit 3 (TMA3) of the timer mode register A (TMA) cannot be rewritten.

Set bit 0 (TACKSTP) of the clock stop register 1 (CKSTPR1) to 1 before rewriting bit 3 (TMA3) of the timer mode register A (TMA).

# 9.3 Timer C

## 9.3.1 Overview

Timer C is an 8-bit timer that increments each time a clock pulse is input. This timer has two operation modes, interval and auto reload.

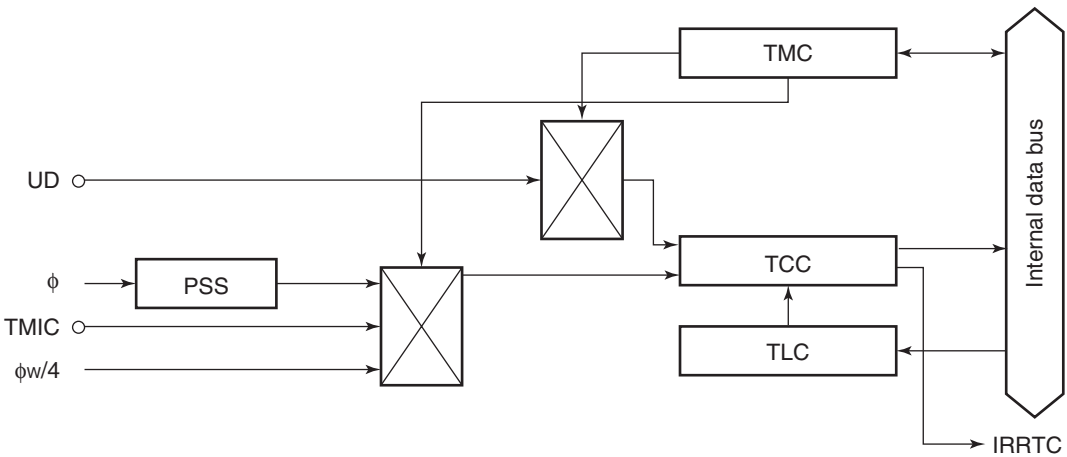
### (1) Features

Features of timer C are given below.

- Choice of seven internal clock sources (ϕ/8192, ϕ/2048, ϕ/512, ϕ/64, ϕ/16, ϕ/4, ϕw/4) or an external clock (can be used to count external events).
- An interrupt is requested when the counter overflows.
- Up/down-counter switching is possible by hardware or software.
- Subactive mode and subsleep mode operation is possible when ϕw/4 is selected as the internal clock, or when an external clock is selected.
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### (2) Block Diagram

Figure 9.2 shows a block diagram of timer C.




Legend:
TMC: Timer mode register C
TCC: Timer counter C
TLC: Timer load register C
IRRTC: Timer C overflow interrupt request flag
PSS: Prescaler S

**Figure 9.2 Block Diagram of Timer C**

### (3) Pin Configuration

Table 9.5 shows the timer C pin configuration.

**Table 9.5 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Timer C event input | TMIC | Input | Input pin for event input to TCC |
| Timer C up/down-count selection | UD | Input | Timer C up/down select |

### (4) Register Configuration

Table 9.6 shows the register configuration of timer C.

**Table 9.6 Timer C Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer mode register C | TMC | R/W | H'18 | H'FFB4 |
| Timer counter C | TCC | R | H'00 | H'FFB5 |
| Timer load register C | TLC | W | H'00 | H'FFB5 |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |

## 9.3.2 Register Descriptions

### (1) Timer Mode Register C (TMC)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMC7 | TMC6 | TMC5 | — | — | TMC2 | TMC1 | TMC0 |
| Initial value | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | — | — | R/W | R/W | R/W |

TMC is an 8-bit read/write register for selecting the auto-reload function and input clock, and performing up/down-counter control.

Upon reset, TMC is initialized to H'18.

**Bit 7:** Auto-reload function select (TMC7)

Bit 7 selects whether timer C is used as an interval timer or auto-reload timer.

| Bit 7<br>TMC7 | Description | |
|---|---|---|
| 0 | Interval timer function selected | (initial value) |
| 1 | Auto-reload function selected | |

**Bits 6 and 5:** Counter up/down control (TMC6, TMC5)

Selects whether TCC up/down control is performed by hardware using UD pin input, or whether TCC functions as an up-counter or a down-counter.

| Bit 6<br>TMC6 | Bit 5<br>TMC5 | Description | |
|---|---|---|---|
| 0 | 0 | TCC is an up-counter | (initial value) |
| 0 | 1 | TCC is a down-counter | |
| 1 | * | Hardware control by UD pin input<br>UD pin input high: Down-counter<br>UD pin input low: Up-counter | |

*: Don't care

**Bits 4 and 3:** Reserved bits

Bits 4 and 3 are reserved; they are always read as 1 and cannot be modified.

**Bits 2 to 0:** Clock select (TMC2 to TMC0)

Bits 2 to 0 select the clock input to TCC. For external event counting, either the rising or falling edge can be selected.

| Bit 2<br>TMC2 | Bit 1<br>TMC1 | Bit 0<br>TMC0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | Internal clock: $\phi$/8192 | (initial value) |
| 0 | 0 | 1 | Internal clock: $\phi$/2048 | |
| 0 | 1 | 0 | Internal clock: $\phi$/512 | |
| 0 | 1 | 1 | Internal clock: $\phi$/64 | |
| 1 | 0 | 0 | Internal clock: $\phi$/16 | |
| 1 | 0 | 1 | Internal clock: $\phi$/4 | |
| 1 | 1 | 0 | Internal clock: $\phi$w/4 | |
| 1 | 1 | 1 | External event (TMIC): rising or falling edge* | |

Note: * The edge of the external event signal is selected by bit IEG1 in the IRQ edge select register (IEGR). See (1) IRQ Edge Select Register (IEGR) in 3.3.2, Interrupt Control Registers, for details. IRQ1 must be set to 1 in port mode register 1 (PMR1) before setting 111 in bits TMC2 to TMC0.

### (2) Timer Counter C (TCC)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCC7 | TCC6 | TCC5 | TCC4 | TCC3 | TCC2 | TCC1 | TCC0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

TCC is an 8-bit read-only up-counter, which is incremented by internal clock or external event input. The clock source for input to this counter is selected by bits TMC2 to TMC0 in timer mode register C (TMC). TCC values can be read by the CPU at any time.

When TCC overflows from H'FF to H'00 or to the value set in TLC, or underflows from H'00 to H'FF or to the value set in TLC, the IRRTC bit in IRR2 is set to 1.

TCC is allocated to the same address as TLC.

Upon reset, TCC is initialized to H'00.

### (3) Timer Load Register C (TLC)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TLC7 | TLC6 | TLC5 | TLC4 | TLC3 | TLC2 | TLC1 | TLC0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

TLC is an 8-bit write-only register for setting the reload value of timer counter C (TCC).

When a reload value is set in TLC, the same value is loaded into timer counter C as well, and TCC starts counting up from that value. When TCC overflows or underflows during operation in auto-reload mode, the TLC value is loaded into TCC. Accordingly, overflow/underflow periods can be set within the range of 1 to 256 input clocks.

The same address is allocated to TLC as to TCC.

Upon reset, TLC is initialized to H'00.

### (4) Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to timer C is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 1:** Timer C module standby mode control (TCCKSTP)

Bit 1 controls setting and clearing of module standby mode for timer C.

| TCCKSTP | Description | |
|---|---|---|
| 0 | Timer C is set to module standby mode | |
| 1 | Timer C module standby mode is cleared | (initial value) |

## 9.3.3 Timer Operation

### (1) Interval Timer Operation

When bit TMC7 in timer mode register C (TMC) is cleared to 0, timer C functions as an 8-bit interval timer.

Upon reset, TCC is initialized to H'00 and TMC to H'18, so TCC continues up-counting as an interval up-counter without halting immediately after a reset. The timer C operating clock is selected from seven internal clock signals output by prescalers S and W, or an external clock input at pin TMIC. The selection is made by bits TMC2 to TMC0 in TMC.

TCC up/down-count control can be performed either by software or hardware. The selection is made by bits TMC6 and TMC5 in TMC.

After the count value in TCC reaches H'FF (H'00), the next clock input causes timer C to overflow (underflow), setting bit IRRTC to 1 in IRR2. If IENTC = 1 in interrupt enable register 2 (IENR2), a CPU interrupt is requested.

At overflow (underflow), TCC returns to H'00 (H'FF) and starts counting up (down) again.

During interval timer operation (TMC7 = 0), when a value is set in timer load register C (TLC), the same value is set in TCC.

Note: For details on interrupts, see section 3.3, Interrupts.

### (2) Auto-Reload Timer Operation

Setting bit TMC7 in TMC to 1 causes timer C to function as an 8-bit auto-reload timer. When a reload value is set in TLC, the same value is loaded into TCC, becoming the value from which TCC starts its count.

After the count value in TCC reaches H'FF (H'00), the next clock signal input causes timer C to overflow/underflow. The TLC value is then loaded into TCC, and the count continues from that value. The overflow/underflow period can be set within a range from 1 to 256 input clocks, depending on the TLC value.

The clock sources, up/down control, and interrupts in auto-reload mode are the same as in interval mode.

In auto-reload mode (TMC7 = 1), when a new value is set in TLC, the TLC value is also set in TCC.

### (3) Event Counter Operation

Timer C can operate as an event counter, counting rising or falling edges of an external event signal input at pin TMIC. External event counting is selected by setting bits TMC2 to TMC0 in timer mode register C to all 1s (111).

When timer C is used to count external event input, bit IRQ1 in PMR1 should be set to 1 and bit IEN1 in IENR1 cleared to 0 to disable interrupt $IRQ_1$ requests.

### (4) TCC Up/Down Control by Hardware

With timer C, TCC up/down control can be performed by UD pin input. When bit TMC6 is set to 1 in TMC, TCC functions as an up-counter when UD pin input is high, and as a down-counter when low.

When using UD pin input, set bit UD to 1 in PMR3.

### 9.3.4 Timer C Operation States

Table 9.7 summarizes the timer C operation states.

**Table 9.7 Timer C Operation States**

| Operation Mode | | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|---|
| TCC | Interval | Reset | Functions | Functions | Halted | Functions/ Halted* | Functions/ Halted* | Halted | Halted |
| | Auto reload | Reset | Functions | Functions | Halted | Functions/ Halted* | Functions/ Halted* | Halted | Halted |
| TMC | | Reset | Functions | Retained | Retained | Functions | Retained | Retained | Retained |

Note: * When $\phi w/4$ is selected as the TCC internal clock in active mode or sleep mode, since the system clock and internal clock are mutually asynchronous, synchronization is maintained by a synchronization circuit. This results in a maximum count cycle error of $1/\phi$ (s). When the counter is operated in subactive mode or subsleep mode, either select $\phi w/4$ as the internal clock or select an external clock. The counter will not operate on any other internal clock. If $\phi w/4$ is selected as the internal clock for the counter when $\phi w/8$ has been selected as subclock $\phi_{SUB}$, the lower 2 bits of the counter operate on the same cycle, and the operation of the least significant bit is unrelated to the operation of the counter.

### 9.3.5 Usage Note

Note the following regarding the operation of timer C.

(1) Counting errors caused by external event input

Timer counter errors may occur under the following conditions.

Conditions

- An external event (TMIC) is used in subsleep mode.

Symptom

- The counter increments or decrements twice for a single external event input.

Approximate rate of occurrence

The approximate rate of occurrence in cases where the external event input is not synchronized with internal operation is defined by the following equation.

$$\text{Approximate rate of occurrence } P = 30 \text{ ns} / \text{tsubcyc}$$

For example, if tsubcyc = 61.06 μs (subclock ϕw/2), P = 0.0005 (0.05%). If 2,000 external event inputs occur, there is a likelihood that one of them will cause the counter to increment or decrement twice (+2 or –2).

The symptom described is caused by the internal circuit configuration of the device and therefore difficult to avoid. Therefore, it is not advisable to use the clock counter for applications requiring a high degree of accuracy.

## 9.4 Timer F

### 9.4.1 Overview

Timer F is a 16-bit timer with a built-in output compare function. As well as counting external events, timer F also provides for counter resetting, interrupt request generation, toggle output, etc., using compare match signals. Timer F can also be used as two independent 8-bit timers (timer FH and timer FL).

#### (1) Features

Features of timer F are given below.

- Choice of four internal clock sources (ϕ/32, ϕ/16, ϕ/4, ϕw/4) or an external clock (can be used as an external event counter)
- TMOFH pin (TMOFL pin) toggle output provided using a single compare match signal (toggle output initial value can be set)
- Counter resetting by a compare match signal
- Two interrupt sources: one compare match, one overflow
- Can operate as two independent 8-bit timers (timer FH and timer FL) (in 8-bit mode).

| | **Timer FH 8-Bit Timer*** | **Timer FL 8-Bit Timer/Event Counter** |
|---|---|---|
| Internal clock | Choice of 4 (ϕ/32, ϕ/16, ϕ/4, ϕw/4) | |
| Event input | — | TMIF pin |
| Toggle output | One compare match signal, output to TMOFH pin(initial value settable) | One compare match signal, output to TMOFL pin (initial value settable) |
| Counter reset | Counter can be reset by compare match signal | |
| Interrupt sources | One compare match<br>One overflow | |

Note: * When timer F operates as a 16-bit timer, it operates on the timer FL overflow signal.

- Operation in watch mode, subactive mode, and subsleep mode

  When ϕw/4 is selected as the internal clock, timer F can operate in watch mode, subactive mode, and subsleep mode.
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### (2) Block Diagram

Figure 9.3 shows a block diagram of timer F.

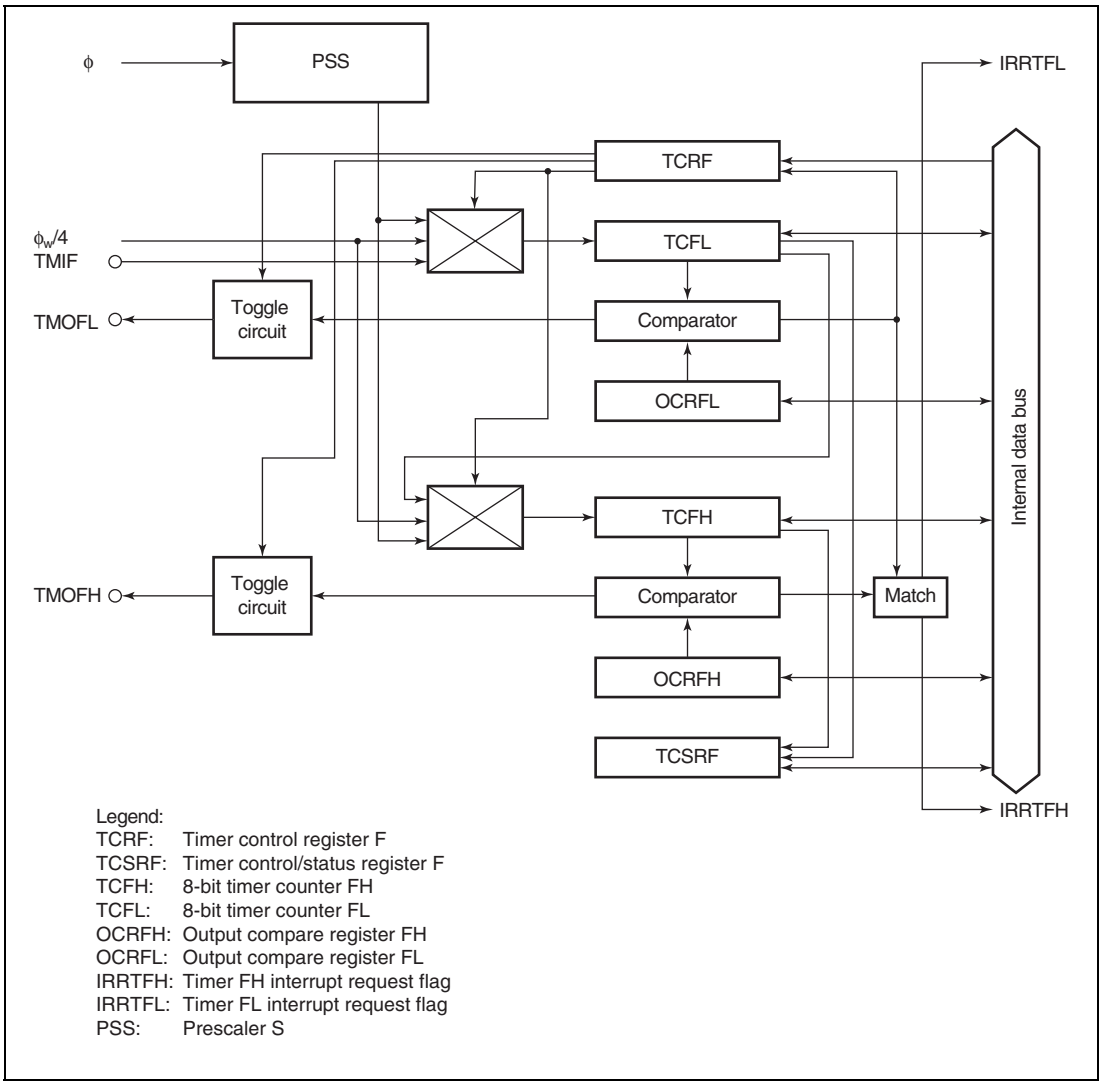



**Figure 9.3 Block Diagram of Timer F**

### (3) Pin Configuration

Table 9.8 shows the timer F pin configuration.

**Table 9.8 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Timer F event input | TMIF | Input | Event input pin for input to TCFL |
| Timer FH output | TMOFH | Output | Timer FH toggle output pin |
| Timer FL output | TMOFL | Output | Timer FL toggle output pin |

### (4) Register Configuration

Table 9.9 shows the register configuration of timer F.

**Table 9.9 Timer F Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer control register F | TCRF | W | H'00 | H'FFB6 |
| Timer control/status register F | TCSRF | R/W | H'00 | H'FFB7 |
| 8-bit timer counter FH | TCFH | R/W | H'00 | H'FFB8 |
| 8-bit timer counter FL | TCFL | R/W | H'00 | H'FFB9 |
| Output compare register FH | OCRFH | R/W | H'FF | H'FFBA |
| Output compare register FL | OCRFL | R/W | H'FF | H'FFBB |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |

### 9.4.2 Register Descriptions

#### (1) 16-bit Timer Counter (TCF)
#### 8-bit Timer Counter (TCFH)
#### 8-bit Timer Counter (TCFL)

| | TCF | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | TCFH | | | | | | | | TCFL | | | | | | | |

TCF is a 16-bit read/write up-counter configured by cascaded connection of 8-bit timer counters TCFH and TCFL. In addition to the use of TCF as a 16-bit counter with TCFH as the upper 8 bits and TCFL as the lower 8 bits, TCFH and TCFL can also be used as independent 8-bit counters.

TCFH and TCFL can be read and written by the CPU, but when they are used in 16-bit mode, data transfer to and from the CPU is performed via a temporary register (TEMP). For details of TEMP, see section 9.4.3, CPU Interface.

TCFH and TCFL are each initialized to H'00 upon reset.

a. 16-bit mode (TCF)

   When CKSH2 is cleared to 0 in TCRF, TCF operates as a 16-bit counter. The TCF input clock is selected by bits CKSL2 to CKSL0 in TCRF.

   TCF can be cleared in the event of a compare match by means of CCLRH in TCSRF.

   When TCF overflows from H'FFFF to H'0000, OVFH is set to 1 in TCSRF. If OVIEH in TCSRF is 1 at this time, IRRTFH is set to 1 in IRR2, and if IENTFH in IENR2 is 1, an interrupt request is sent to the CPU.

b. 8-bit mode (TCFL/TCFH)

   When CKSH2 is set to 1 in TCRF, TCFH and TCFL operate as two independent 8-bit counters. The TCFH (TCFL) input clock is selected by bits CKSH2 to CKSH0 (CKSL2 to CKSL0) in TCRF.

   TCFH (TCFL) can be cleared in the event of a compare match by means of CCLRH (CCLRL) in TCSRF.

When TCFH (TCFL) overflows from H'FF to H'00, OVFH (OVFL) is set to 1 in TCSRF. If OVIEH (OVIEL) in TCSRF is 1 at this time, IRRTFH (IRRTFL) is set to 1 in IRR2, and if IENTFH (IENTFL) in IENR2 is 1, an interrupt request is sent to the CPU.

**(2) 16-bit Output Compare Register (OCRF)**
**8-bit Output Compare Register (OCRFH)**
**8-bit Output Compare Register (OCRFL)**

| | OCRF | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | OCRFH | | | | | | | | OCRFL | | | | | | | |

OCRF is a 16-bit read/write register composed of the two registers OCRFH and OCRFL. In addition to the use of OCRF as a 16-bit register with OCRFH as the upper 8 bits and OCRFL as the lower 8 bits, OCRFH and OCRFL can also be used as independent 8-bit registers.

OCRFH and OCRFL can be read and written by the CPU, but when they are used in 16-bit mode, data transfer to and from the CPU is performed via a temporary register (TEMP). For details of TEMP, see section 9.4.3, CPU Interface.

OCRFH and OCRFL are each initialized to H'FF upon reset.

a. 16-bit mode (OCRF)

When CKSH2 is cleared to 0 in TCRF, OCRF operates as a 16-bit register. OCRF contents are constantly compared with TCF, and when both values match, CMFH is set to 1 in TCSRF. At the same time, IRRTFH is set to 1 in IRR2. If IENTFH in IENR2 is 1 at this time, an interrupt request is sent to the CPU.

Toggle output can be provided from the TMOFH pin by means of compare matches, and the output level can be set (high or low) by means of TOLH in TCRF.

b. 8-bit mode (OCRFH/OCRFL)

When CKSH2 is set to 1 in TCRF, OCRFH and OCRFL operate as two independent 8-bit registers. OCRFH contents are compared with TCFH, and OCRFL contents are with TCFL. When the OCRFH (OCRFL) and TCFH (TCFL) values match, CMFH (CMFL) is set to 1 in TCSRF. At the same time, IRRTFH (IRRTFL) is set to 1 in IRR2. If IENTFH (IENTFL) in IENR2 is 1 at this time, an interrupt request is sent to the CPU.

Toggle output can be provided from the TMOFH pin (TMOFL pin) by means of compare matches, and the output level can be set (high or low) by means of TOLH (TOLL) in TCRF.

### (3) Timer Control Register F (TCRF)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TOLH | CKSH2 | CKSH1 | CKSH0 | TOLL | CKSL2 | CKSL1 | CKSL0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

TCRF is an 8-bit write-only register that switches between 16-bit mode and 8-bit mode, selects the input clock from among four internal clock sources or external event input, and sets the output level of the TMOFH and TMOFL pins.

TCRF is initialized to H'00 upon reset.

**Bit 7:** Toggle output level H (TOLH)

Bit 7 sets the TMOFH pin output level. The output level is effective immediately after this bit is written.

| Bit 7<br>TOLH | Description | |
|---|---|---|
| 0 | Low level | (initial value) |
| 1 | High level | |

**Bits 6 to 4:** Clock select H (CKSH2 to CKSH0)

Bits 6 to 4 select the clock input to TCFH from among four internal clock sources or TCFL overflow.

| Bit 6 CKSH2 | Bit 5 CKSH1 | Bit 4 CKSH0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | 16-bit mode, counting on TCFL overflow signal (initial value) |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | Use prohibited |
| 1 | 0 | 0 | Internal clock: Counting on $\phi/32$ |
| 1 | 0 | 1 | Internal clock: Counting on $\phi/16$ |
| 1 | 1 | 0 | Internal clock: Counting on $\phi/4$ |
| 1 | 1 | 1 | Internal clock: Counting on $\phi w/4$ |

**Bit 3:** Toggle output level L (TOLL)

Bit 3 sets the TMOFL pin output level. The output level is effective immediately after this bit is written.

| Bit 3 TOLL | Description |
|---|---|
| 0 | Low level (initial value) |
| 1 | High level |

**Bits 2 to 0:** Clock select L (CKSL2 to CKSL0)

Bits 2 to 0 select the clock input to TCFL from among four internal clock sources or external event input.

| Bit 2 CKSL2 | Bit 1 CKSL1 | Bit 0 CKSL0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | Counting on external event (TMIF) rising/falling edge* (initial value) |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | Use prohibited |
| 1 | 0 | 0 | Internal clock: Counting on $\phi/32$ |
| 1 | 0 | 1 | Internal clock: Counting on $\phi/16$ |
| 1 | 1 | 0 | Internal clock: Counting on $\phi/4$ |
| 1 | 1 | 1 | Internal clock: Counting on $\phi w/4$ |

Note: * External event edge selection is set by IEG3 in the IRQ edge select register (IEGR). For details, see (1) IRQ Edge Select Register (IEGR) in section 3.3.2, Interrupt Control Registers.

Note that the timer F counter may increment if the setting of IRQ3 in port mode register 1 (PMR1) is changed from 0 to 1 while the TMIF pin is low in order to change the TMIF pin function.

### (4) Timer Control/Status Register F (TCSRF)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OVFH | CMFH | OVIEH | CCLRH | OVFL | CMFL | OVIEL | CCLRL |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W* | R/W* | R/W | R/W | R/W* | R/W* | R/W | R/W |

Note: * Bits 7, 6, 3, and 2 can only be written with 0, for flag clearing.

TCSRF is an 8-bit read/write register that performs counter clear selection, overflow flag setting, and compare match flag setting, and controls enabling of overflow interrupt requests.

TCSRF is initialized to H'00 upon reset.

**Bit 7:** Timer overflow flag H (OVFH)

Bit 7 is a status flag indicating that TCFH has overflowed from H'FF to H'00. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 7<br>OVFH | Description | |
|---|---|---|
| 0 | Clearing condition:<br>After reading OVFH = 1, cleared by writing 0 to OVFH | (initial value) |
| 1 | Setting condition:<br>Set when TCFH overflows from H'FF to H'00 | |

**Bit 6:** Compare match flag H (CMFH)

Bit 6 is a status flag indicating that TCFH has matched OCRFH. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 6<br>CMFH | Description | |
|---|---|---|
| 0 | Clearing condition:<br>After reading CMFH = 1, cleared by writing 0 to CMFH | (initial value) |
| 1 | Setting condition:<br>Set when the TCFH value matches the OCRFH value | |

**Bit 5:** Timer overflow interrupt enable H (OVIEH)

Bit 5 selects enabling or disabling of interrupt generation when TCFH overflows.

| Bit 5<br>OVIEH | Description | |
|---|---|---|
| 0 | TCFH overflow interrupt request is disabled | (initial value) |
| 1 | TCFH overflow interrupt request is enabled | |

**Bit 4:** Counter clear H (CCLRH)

In 16-bit mode, bit 4 selects whether TCF is cleared when TCF and OCRF match.

In 8-bit mode, bit 4 selects whether TCFH is cleared when TCFH and OCRFH match.

| Bit 4<br>CCLRH | Description | |
|---|---|---|
| 0 | 16-bit mode: TCF clearing by compare match is disabled<br>8-bit mode: TCFH clearing by compare match is disabled | (initial value) |
| 1 | 16-bit mode: TCF clearing by compare match is enabled<br>8-bit mode: TCFH clearing by compare match is enabled | |

**Bit 3:** Timer overflow flag L (OVFL)

Bit 3 is a status flag indicating that TCFL has overflowed from H'FF to H'00. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 3<br>OVFL | Description | |
|---|---|---|
| 0 | Clearing condition:<br>After reading OVFL = 1, cleared by writing 0 to OVFL | (initial value) |
| 1 | Setting condition:<br>Set when TCFL overflows from H’FF to H’00 | |

**Bit 2:** Compare match flag L (CMFL)

Bit 2 is a status flag indicating that TCFL has matched OCRFL. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 2<br>CMFL | Description | |
|---|---|---|
| 0 | Clearing condition:<br>After reading CMFL = 1, cleared by writing 0 to CMFL | (initial value) |
| 1 | Setting condition:<br>Set when the TCFL value matches the OCRFL value | |

**Bit 1:** Timer overflow interrupt enable L (OVIEL)

Bit 1 selects enabling or disabling of interrupt generation when TCFL overflows.

| Bit 1<br>OVIEL | Description | |
|---|---|---|
| 0 | TCFL overflow interrupt request is disabled | (initial value) |
| 1 | TCFL overflow interrupt request is enabled | |

**Bit 0:** Counter clear L (CCLRL)

Bit 0 selects whether TCFL is cleared when TCFL and OCRFL match.

| Bit 0<br>CCLRL | Description | |
|---|---|---|
| 0 | TCFL clearing by compare match is disabled | (initial value) |
| 1 | TCFL clearing by compare match is enabled | |

### (5) Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to timer F is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 2:** Timer F module standby mode control (TFCKSTP)

Bit 2 controls setting and clearing of module standby mode for timer F.

| TFCKSTP | Description | |
|---|---|---|
| 0 | Timer F is set to module standby mode | |
| 1 | Timer F module standby mode is cleared | (initial value) |

### 9.4.3 CPU Interface

TCF and OCRF are 16-bit read/write registers, but the CPU is connected to the on-chip peripheral modules by an 8-bit data bus. When the CPU accesses these registers, it therefore uses an 8-bit temporary register (TEMP).

In 16-bit mode, TCF read/write access and OCRF write access must be performed 16 bits at a time (using two consecutive byte-size MOV instructions), and the upper byte must be accessed before the lower byte. Data will not be transferred correctly if only the upper byte or only the lower byte is accessed.

In 8-bit mode, there are no restrictions on the order of access.

#### (1) Write Access

Write access to the upper byte results in transfer of the upper-byte write data to TEMP. Next, write access to the lower byte results in transfer of the data in TEMP to the upper register byte, and direct transfer of the lower-byte write data to the lower register byte.

Figure 9.4 shows an example in which H'AA55 is written to TCF.

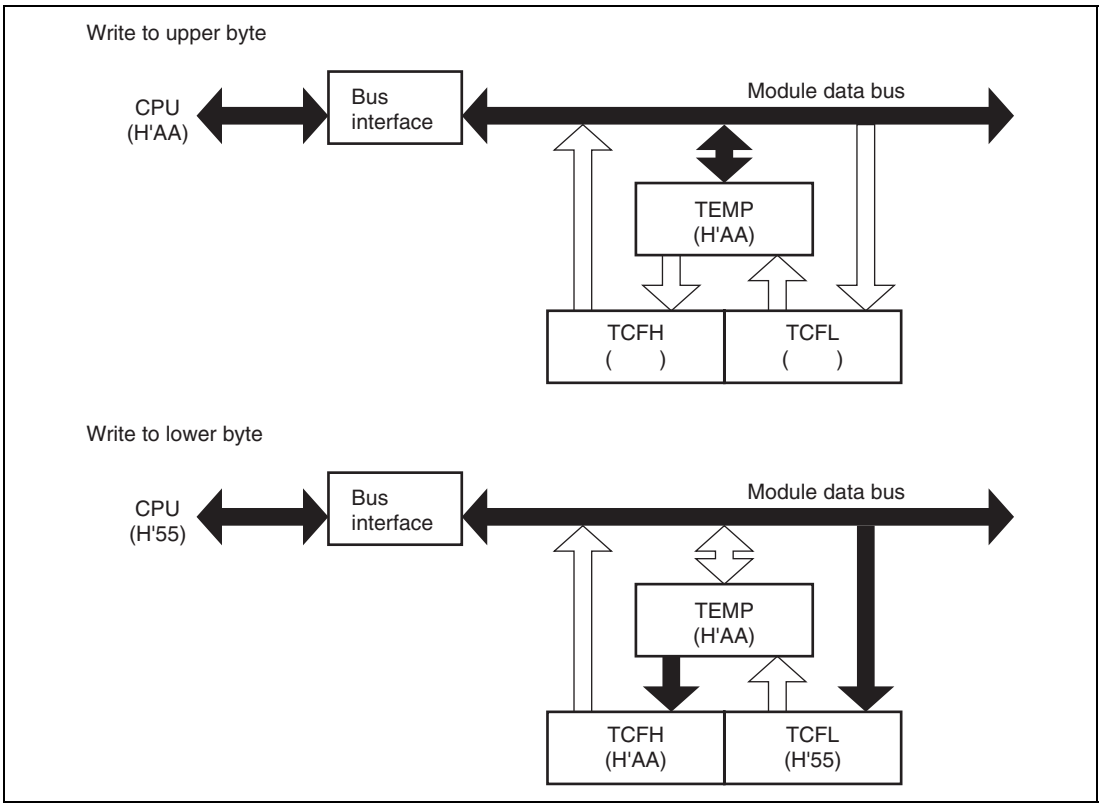



**Figure 9.4 Write Access to TCR (CPU → TCF)**

### (2) Read Access

In access to TCF, when the upper byte is read the upper-byte data is transferred directly to the CPU and the lower-byte data is transferred to TEMP. Next, when the lower byte is read, the lower-byte data in TEMP is transferred to the CPU.

In access to OCRF, when the upper byte is read the upper-byte data is transferred directly to the CPU. When the lower byte is read, the lower-byte data is transferred directly to the CPU.

Figure 9.5 shows an example in which TCF is read when it contains H'AAFF.

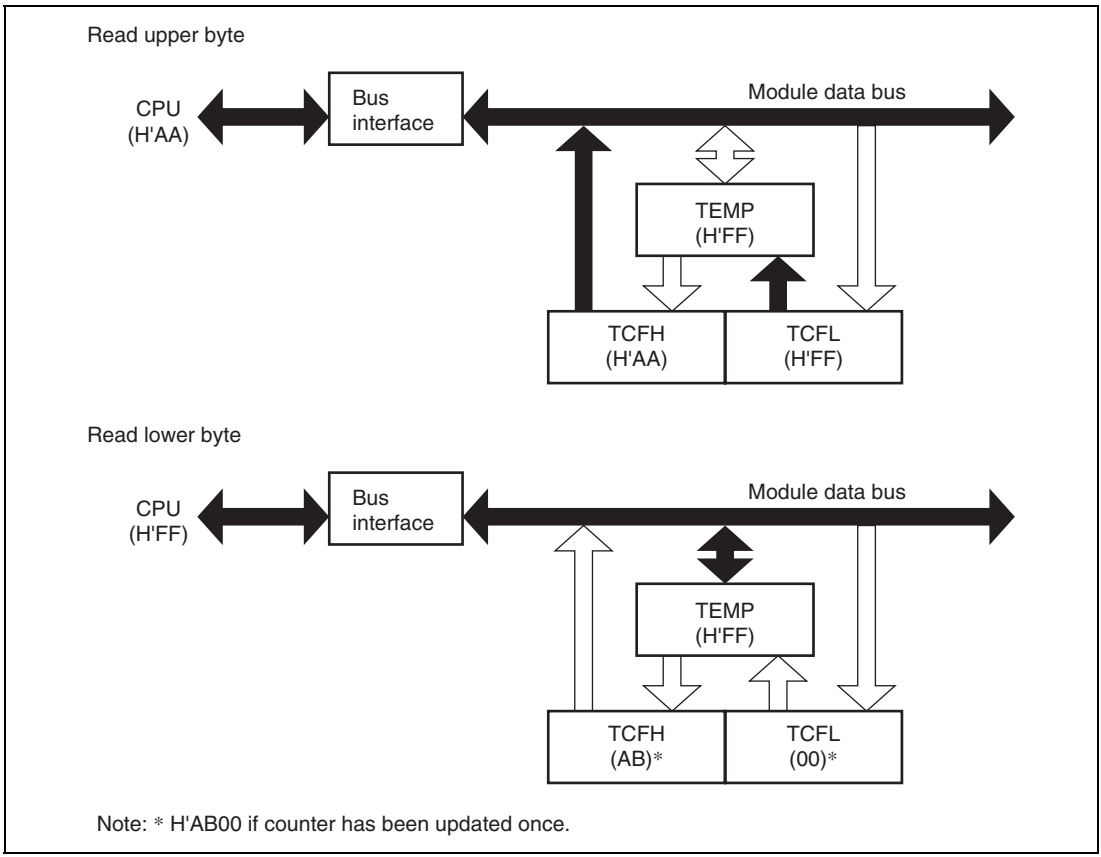



**Figure 9.5 Read Access to TCF (TCF → CPU)**

### 9.4.4 Operation

Timer F is a 16-bit counter that increments on each input clock pulse. The timer F value is constantly compared with the value set in output compare register F, and the counter can be cleared, an interrupt requested, or port output toggled, when the two values match. Timer F can also function as two independent 8-bit timers.

### (1) Timer F Operation

Timer F has two operating modes, 16-bit timer mode and 8-bit timer mode. The operation in each of these modes is described below.

a. Operation in 16-bit timer mode

When CKSH2 is cleared to 0 in timer control register F (TCRF), timer F operates as a 16-bit timer.

Following a reset, timer counter F (TCF) is initialized to H'0000, output compare register F (OCRF) to H'FFFF, and timer control register F (TCRF) and timer control/status register F (TCSRF) to H'00. The counter starts incrementing on external event (TMIF) input. The external event edge selection is set by IEG3 in the IRQ edge select register (IEGR).

The timer F operating clock can be selected from four internal clocks or an external clock by means of bits CKSL2 to CKSL0 in TCRF.

OCRF contents are constantly compared with TCF, and when both values match, CMFH is set to 1 in TCSRF. If IENTFH in IENR2 is 1 at this time, an interrupt request is sent to the CPU, and at the same time, TMOFH pin output is toggled. If CCLRH in TCSRF is 1, TCF is cleared. TMOFH pin output can also be set by TOLH in TCRF.

When TCF overflows from H'FFFF to H'0000, OVFH is set to 1 in TCSRF. If OVIEH in TCSRF and IENTFH in IENR2 are both 1, an interrupt request is sent to the CPU.

b. Operation in 8-bit timer mode

When CKSH2 is set to 1 in TCRF, TCF operates as two independent 8-bit timers, TCFH and TCFL. The TCFH/TCFL input clock is selected by CKSH2 to CKSH0/CKSL2 to CKSL0 in TCRF.

When the OCRFH/OCRFL and TCFH/TCFL values match, CMFH/CMFL is set to 1 in TCSRF. If IENTFH/IENTFL in IENR2 is 1, an interrupt request is sent to the CPU, and at the same time, TMOFH pin/TMOFL pin output is toggled. If CCLRH/CCLRL in TCSRF is 1, TCFH/TCFL is cleared. TMOFH pin/TMOFL pin output can also be set by TOLH/TOLL in TCRF.

When TCFH/TCFL overflows from H'FF to H'00, OVFH/OVFL is set to 1 in TCSRF. If OVIEH/OVIEL in TCSRF and IENTFH/IENTFL in IENR2 are both 1, an interrupt request is sent to the CPU.

### (2) TCF Increment Timing

TCF is incremented by clock input (internal clock or external event input).

a. Internal clock operation

Bits CKSH2 to CKSH0 or CKSL2 to CKSL0 in TCRF select one of four internal clock sources (ϕ/32, ϕ/16, ϕ/4, or ϕw/4) created by dividing the system clock (ϕ or ϕw).

b. External event operation

External event input is selected by clearing CKSL2 to 0 in TCRF. TCF can increment on either the rising or falling edge of external event input. External event edge selection is set by IEG3 in the interrupt controller's IEGR register. An external event pulse width of at least 2 system clocks (ϕ) is necessary. Shorter pulses will not be counted correctly.

### (3) TMOFH/TMOFL Output Timing

In TMOFH/TMOFL output, the value set in TOLH/TOLL in TCRF is output. The output is toggled by the occurrence of a compare match. Figure 9.6 shows the output timing.

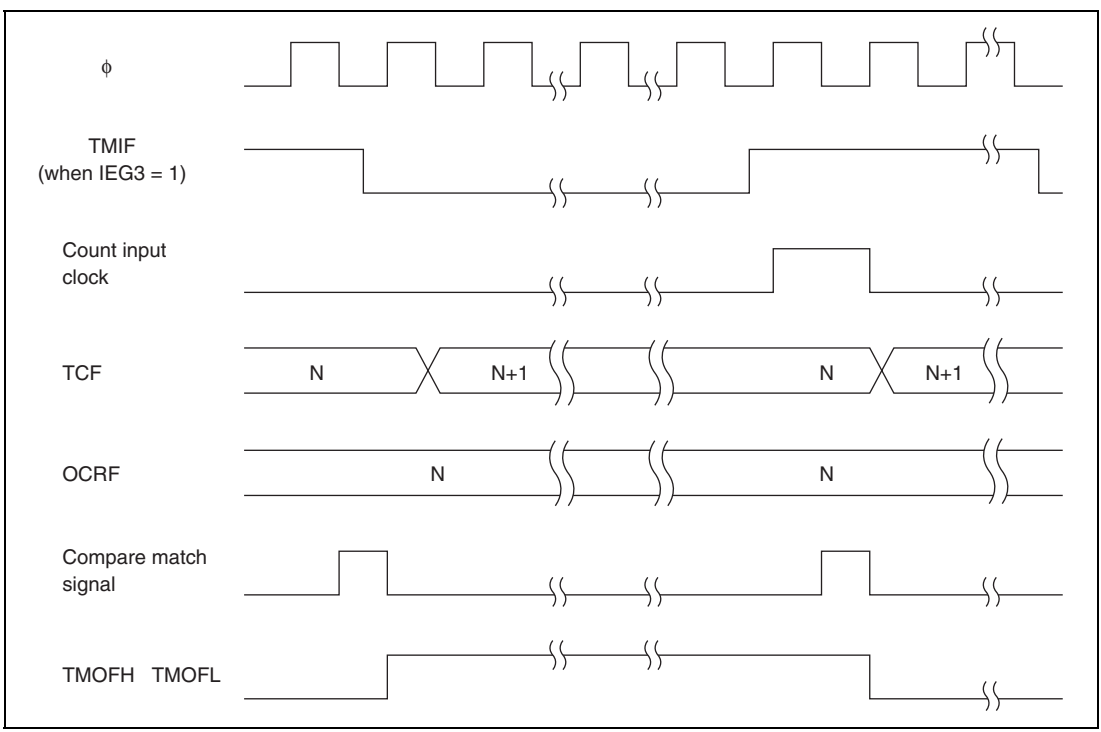



**Figure 9.6 TMOFH/TMOFL Output Timing**

### (4) TCF Clear Timing

TCF can be cleared by a compare match with OCRF.

### (5) Timer Overflow Flag (OVF) Set Timing

OVF is set to 1 when TCF overflows from H'FFFF to H'0000.

### (6) Compare Match Flag set Timing

The compare match flag (CMFH or CMFL) is set to 1 when the TCF and OCRF values match. The compare match signal is generated in the last state during which the values match (when TCF is updated from the matching value to a new value). When TCF matches OCRF, the compare match signal is not generated until the next counter clock.

### (7) Timer F Operation Modes

Timer F operation modes are shown in table 9.10.

**Table 9.10 Timer F Operation Modes**

| Operation Mode | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|
| TCF | Reset | Functions | Functions | Functions/ Halted* | Functions/ Halted* | Functions/ Halted* | Halted | Halted |
| OCRF | Reset | Functions | Held | Held | Functions | Held | Held | Held |
| TCRF | Reset | Functions | Held | Held | Functions | Held | Held | Held |
| TCSRF | Reset | Functions | Held | Held | Functions | Held | Held | Held |

Note: * When $\phi_w/4$ is selected as the TCF internal clock in active mode or sleep mode, since the system clock and internal clock are mutually asynchronous, synchronization is maintained by a synchronization circuit. This results in a maximum count cycle error of $1/\phi$ (s). When the counter is operated in subactive mode, watch mode, or subsleep mode, $\phi_w/4$ must be selected as the internal clock. The counter will not operate if any other internal clock is selected.

### 9.4.5 Application Notes

The following types of contention and operation can occur when timer F is used.

#### (1) 16-bit Timer Mode

In toggle output, TMOFH pin output is toggled when all 16 bits match and a compare match signal is generated. If a TCRF write by a MOV instruction and generation of the compare match signal occur simultaneously, TOLH data is output to the TMOFH pin as a result of the TCRF write. TMOFL pin output is unstable in 16-bit mode, and should not be used; the TMOFL pin should be used as a port pin.

If an OCRFL write and compare match signal generation occur simultaneously, the compare match signal is invalid. However, if the written data and the counter value match, a compare match signal will be generated at that point. As the compare match signal is output in synchronization with the TCFL clock, a compare match will not result in compare match signal generation if the clock is stopped.

Compare match flag CMFH is set when all 16 bits match and a compare match signal is generated. Compare match flag CMFL is set if the setting conditions for the lower 8 bits are satisfied.

When TCF overflows, OVFH is set. OVFL is set if the setting conditions are satisfied when the lower 8 bits overflow. If a TCFL write and overflow signal output occur simultaneously, the overflow signal is not output.

#### (2) 8-bit Timer Mode

a. TCFH, OCRFH

   In toggle output, TMOFH pin output is toggled when a compare match occurs. If a TCRF write by a MOV instruction and generation of the compare match signal occur simultaneously, TOLH data is output to the TMOFH pin as a result of the TCRF write.

   If an OCRFH write and compare match signal generation occur simultaneously, the compare match signal is invalid. However, if the written data and the counter value match, a compare match signal will be generated at that point. The compare match signal is output in synchronization with the TCFH clock.

   If a TCFH write and overflow signal output occur simultaneously, the overflow signal is not output.

b. TCFL, OCRFL

In toggle output, TMOFL pin output is toggled when a compare match occurs. If a TCRF write by a MOV instruction and generation of the compare match signal occur simultaneously, TOLL data is output to the TMOFL pin as a result of the TCRF write.

If an OCRFL write and compare match signal generation occur simultaneously, the compare match signal is invalid. However, if the written data and the counter value match, a compare match signal will be generated at that point. As the compare match signal is output in synchronization with the TCFL clock, a compare match will not result in compare match signal generation if the clock is stopped.

If a TCFL write and overflow signal output occur simultaneously, the overflow signal is not output.

### (3) Clear Timer FH, Timer FL Interrupt Request Flags (IRRTFH, IRRTFL), Timer Overflow Flags H, L (OVFH, OVFL) and Compare Match Flags H, L (CMFH, CMFL)

When ϕw/4 is selected as the internal clock, “Interrupt factor generation signal” will be operated with ϕw and the signal will be outputted with ϕw width. And, “Overflow signal” and “Compare match signal” are controlled with 2 cycles of ϕw signals. Those signals are outputted with 2 cycles width of ϕw (figure 9.7)

In active (high-speed, medium-speed) mode, even if you cleared interrupt request flag during the term of validity of “Interrupt factor generation signal”, same interrupt request flag is set. (figure 9.7 (1)) And, you cannot be cleared timer overflow flag and compare match flag during the term of validity of “Overflow signal” and “Compare match signal”.

For interrupt request flag is set right after interrupt request is cleared, interrupt process to one time timer FH, timer FL interrupt might be repeated. (figure 9.7 (2)) Therefore, to definitely clear interrupt request flag in active (high-speed, medium-speed) mode, clear should be processed after the time that calculated with below (1) formula. And, to definitely clear timer overflow flag and compare match flag, clear should be processed after read timer control status register F (TCSRF) after the time that calculated with below (1) formula. For ST of (1) formula, please substitute the longest number of execution states in used instruction. (10 states of RTE instruction when MULXU, DIVXU instruction is not used, 14 states when MULXU, DIVXU instruction is used) In subactive mode, there are not limitation for interrupt request flag, timer overflow flag, and compare match flag clear.

The term of validity of “Interrupt factor generation signal”

= 1 cycle of $\phi w$ + waiting time for completion of executing instruction
+ interrupt time synchronized with $\phi = 1/\phi w + ST \times (1/\phi) + (2/\phi)$ (second).....(1)

ST: Executing number of execution states

Method 1 is recommended to operate for time efficiency.

Method 1

1. Prohibit interrupt in interrupt handling routine (set IENFH, IENFL to 0).
2. After program process returned normal handling, clear interrupt request flags (IRRTFH, IRRTFL) after more than that calculated with (1) formula.
3. After read timer control status register F (TCSRF), clear timer overflow flags (OVFH, OVFL) and compare match flags (CMFH, CMFL).
4. Operate interrupt permission (set IENFH, IENFL to 1).

Method 2

1. Set interrupt handling routine time to more than time that calculated with (1) formula.
2. Clear interrupt request flags (IRRTFH, IRRTFL) at the end of interrupt handling routine.
3. After read timer control status register F (TCSRF), clear timer overflow flags (OVFH, OVFL) and compare match flags (CMFH, CMFL).

All above attentions are also applied in 16-bit mode and 8-bit mode.

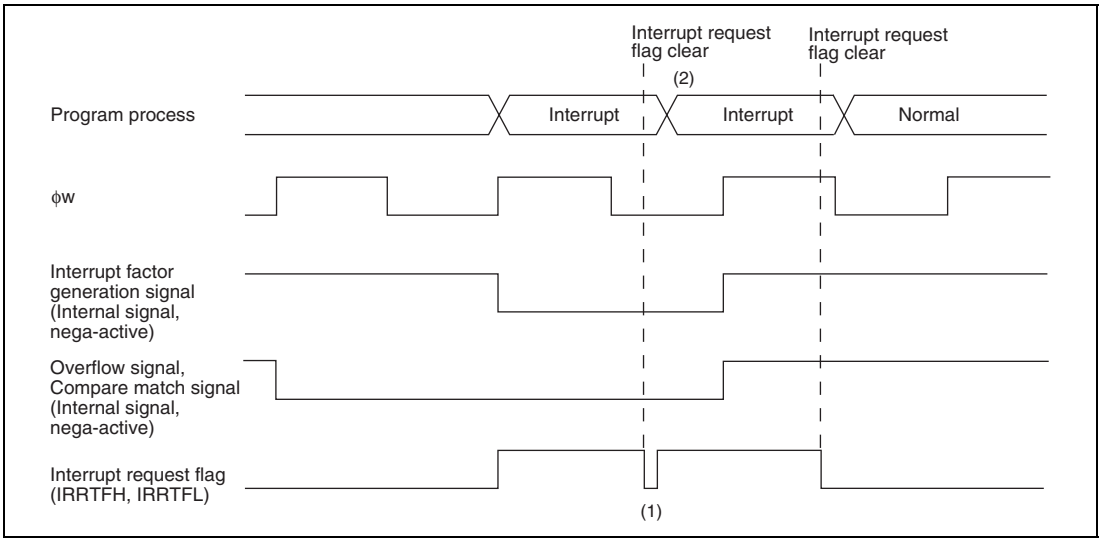



**Figure 9.7 Clear Interrupt Request Flag when Interrupt Factor Generation Signal is Valid**

### (4) Timer Counter (TCF) Read/Write

When ϕw/4 is selected as the internal clock in active (high-speed, medium-speed) mode, write on TCF is impossible. And, when read TCF, as the system clock and internal clock are mutually asynchronous, TCF synchronizes with synchronization circuit. This results in a maximum TCF read value error of ±1.

When read/write TCF in active (high-speed, medium-speed) mode is needed, please select internal clock except for ϕw/4 before read/write.

In subactive mode, even ϕw/4 is selected as the internal clock, normal read/write TCF is possible.

# 9.5 Timer G

## 9.5.1 Overview

Timer G is an 8-bit timer with dedicated input capture functions for the rising/falling edges of pulses input from the input capture input pin (input capture input signal). High-frequency component noise in the input capture input signal can be eliminated by a noise canceler, enabling accurate measurement of the input capture input signal duty cycle. If input capture input is not set, timer G functions as an 8-bit interval timer.

### (1) Features

Features of timer G are given below.

- Choice of four internal clock sources (ϕ/64, ϕ/32, ϕ/2, ϕw/4)
- Dedicated input capture functions for rising and falling edges
- Level detection at counter overflow

  It is possible to detect whether overflow occurred when the input capture input signal was high or when it was low.
- Selection of whether or not the counter value is to be cleared at the input capture input signal rising edge, falling edge, or both edges
- Two interrupt sources: one input capture, one overflow. The input capture input signal rising or falling edge can be selected as the interrupt source.
- A built-in noise canceler eliminates high-frequency component noise in the input capture input signal.
- Watch mode, subactive mode and subsleep mode operation is possible when ϕw/4 is selected as the internal clock.
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### (2) Block Diagram

Figure 9.8 shows a block diagram of timer G.

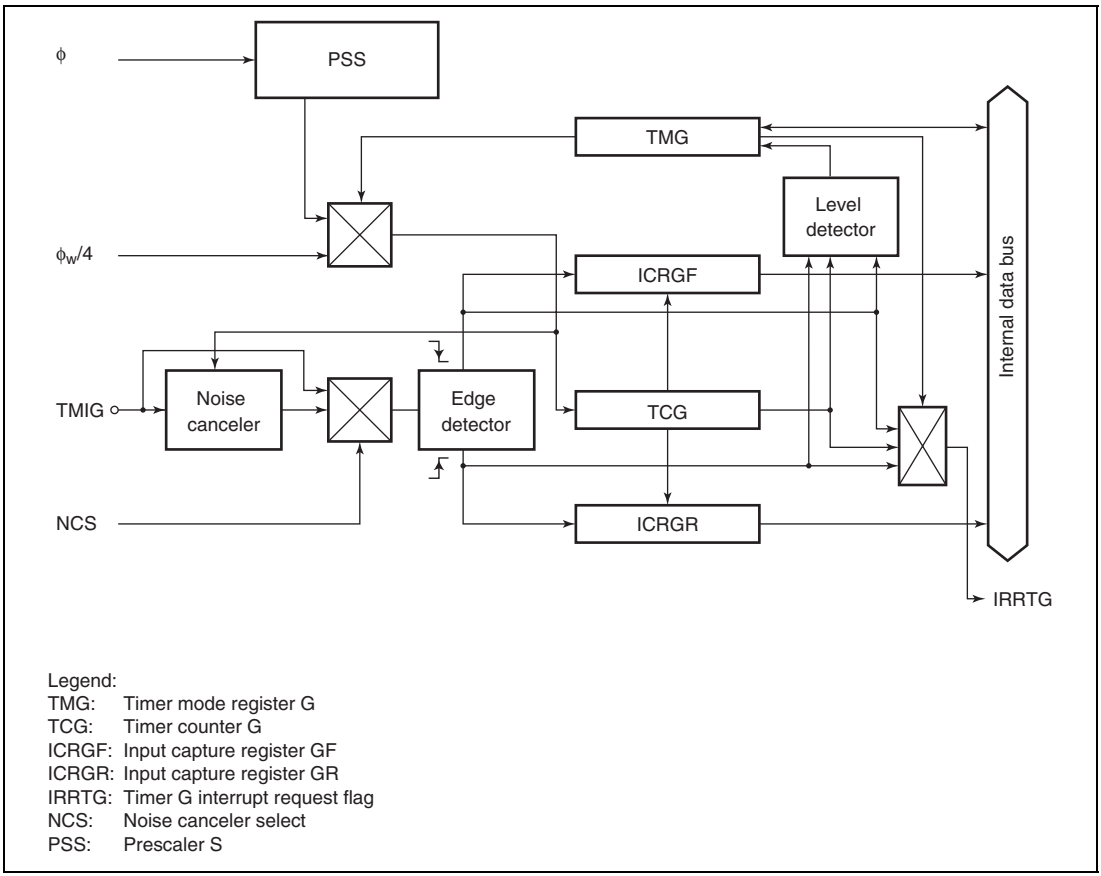



**Figure 9.8 Block Diagram of Timer G**

### (3) Pin Configuration

Table 9.11 shows the timer G pin configuration.

**Table 9.11 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Input capture input | TMIG | Input | Input capture input pin |

### (4) Register Configuration

Table 9.12 shows the register configuration of timer G.

**Table 9.12 Timer G Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer control register G | TMG | R/W | H'00 | H'FFBC |
| Timer counter G | TCG | — | H'00 | — |
| Input capture register GF | ICRGF | R | H'00 | H'FFBD |
| Input capture register GR | ICRGR | R | H'00 | H'FFBE |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |

## 9.5.2 Register Descriptions

### (1) Timer Counter (TCG)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCG7 | TCG6 | TCG5 | TCG4 | TCG3 | TCG2 | TCG1 | TCG0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | — | — | — | — |

TCG is an 8-bit up-counter which is incremented by clock input. The input clock is selected by bits CKS1 and CKS0 in TMG.

TMIG in PMR1 is set to 1 to operate TCG as an input capture timer, or cleared to 0 to operate TCG as an interval timer*. In input capture timer operation, the TCG value can be cleared by the rising edge, falling edge, or both edges of the input capture input signal, according to the setting made in TMG.

When TCG overflows from H'FF to H'00, if OVIE in TMG is 1, IRRTG is set to 1 in IRR2, and if IENTG in IENR2 is 1, an interrupt request is sent to the CPU.

For details of the interrupt, see section 3.3, Interrupts.

TCG cannot be read or written by the CPU. It is initialized to H'00 upon reset.

Note: * An input capture signal may be generated when TMIG is modified.

### (2) Input Capture Register GF (ICRGF)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICRGF7 | ICRGF6 | ICRGF5 | ICRGF4 | ICRGF3 | ICRGF2 | ICRGF1 | ICRGF0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

ICRGF is an 8-bit read-only register. When a falling edge of the input capture input signal is detected, the current TCG value is transferred to ICRGF. If IIEGS in TMG is 1 at this time, IRRTG is set to 1 in IRR2, and if IENTG in IENR2 is 1, an interrupt request is sent to the CPU.

For details of the interrupt, see section 3.3, Interrupts.

To ensure dependable input capture operation, the pulse width of the input capture input signal must be at least $2\phi$ or $2\phi_{SUB}$ (when the noise canceler is not used).

ICRGF is initialized to H'00 upon reset.

### (3) Input Capture Register GR (ICRGR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICRGR7 | ICRGR6 | ICRGR5 | ICRGR4 | ICRGR3 | ICRGR2 | ICRGR1 | ICRGR0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

ICRGR is an 8-bit read-only register. When a rising edge of the input capture input signal is detected, the current TCG value is transferred to ICRGR. If IIEGS in TMG is 1 at this time, IRRTG is set to 1 in IRR2, and if IENTG in IENR2 is 1, an interrupt request is sent to the CPU.

For details of the interrupt, see section 3.3, Interrupts.

To ensure dependable input capture operation, the pulse width of the input capture input signal must be at least $2\phi$ or $2\phi_{SUB}$ (when the noise canceler is not used).

ICRGR is initialized to H'00 upon reset.

### (4) Timer Mode Register G (TMG)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OVFH | OVFL | OVIE | IIEGS | CCLR1 | CCLR0 | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write: | R/W* | R/W* | R/W | R/W | R/W | R/W | R/W | R/W |

Note: * Bits 7 and 6 can only be written with 0, for flag clearing.

TMG is an 8-bit read/write register that performs TCG clock selection from four internal clock sources, counter clear selection, and edge selection for the input capture input signal interrupt request, controls enabling of overflow interrupt requests, and also contains the overflow flags.

TMG is initialized to H'00 upon reset.

**Bit 7:** Timer overflow flag H (OVFH)

Bit 7 is a status flag indicating that TCG has overflowed from H'FF to H'00 when the input capture input signal is high. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 7<br>OVFH | Description |
|---|---|
| 0 | Clearing condition: (initial value)<br>After reading OVFH = 1, cleared by writing 0 to OVFH |
| 1 | Setting condition:<br>Set when TCG overflows from H'FF to H'00 |

**Bit 6:** Timer overflow flag L (OVFL)

Bit 6 is a status flag indicating that TCG has overflowed from H'FF to H'00 when the input capture input signal is low, or in interval operation. This flag is set by hardware and cleared by software. It cannot be set by software.

| Bit 6<br>OVFL | Description | |
|---|---|---|
| 0 | Clearing condition:<br>After reading OVFL = 1, cleared by writing 0 to OVFL | (initial value) |
| 1 | Setting condition:<br>Set when TCG overflows from H'FF to H'00 | |

**Bit 5:** Timer overflow interrupt enable (OVIE)

Bit 5 selects enabling or disabling of interrupt generation when TCG overflows.

| Bit 5<br>OVIE | Description | |
|---|---|---|
| 0 | TCG overflow interrupt request is disabled | (initial value) |
| 1 | TCG overflow interrupt request is enabled | |

**Bit 4:** Input capture interrupt edge select (IIEGS)

Bit 4 selects the input capture input signal edge that generates an interrupt request.

| Bit 4<br>IIEGS | Description | |
|---|---|---|
| 0 | Interrupt generated on rising edge of input capture input signal | (initial value) |
| 1 | Interrupt generated on falling edge of input capture input signal | |

**Bits 3 and 2:** Counter clear 1 and 0 (CCLR1, CCLR0)

Bits 3 and 2 specify whether or not TCG is cleared by the rising edge, falling edge, or both edges of the input capture input signal.

| Bit 3 CCLR1 | Bit 2 CCLR0 | Description | |
|---|---|---|---|
| 0 | 0 | TCG clearing is disabled | (initial value) |
| 0 | 1 | TCG cleared by falling edge of input capture input signal | |
| 1 | 0 | TCG cleared by rising edge of input capture input signal | |
| 1 | 1 | TCG cleared by both edges of input capture input signal | |

**Bits 1 and 0:** Clock select (CKS1, CKS0)

Bits 1 and 0 select the clock input to TCG from among four internal clock sources.

| Bit 1 CKS1 | Bit 0 CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | Internal clock: Counting on $\phi/64$ | (initial value) |
| 0 | 1 | Internal clock: Counting on $\phi/32$ | |
| 1 | 0 | Internal clock: Counting on $\phi/2$ | |
| 1 | 1 | Internal clock: Counting on $\phi w/4$ | |

### (5) Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to timer G is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 3:** Timer G module standby mode control (TGCKSTP)

Bit 3 controls setting and clearing of module standby mode for timer G.

| TGCKSTP | Description | |
|---|---|---|
| 0 | Timer G is set to module standby mode | |
| 1 | Timer G module standby mode is cleared | (initial value) |

### 9.5.3 Noise Canceler

The noise canceler consists of a digital low-pass filter that eliminates high-frequency component noise from the pulses input from the input capture input pin. The noise canceler is set by NCS* in PMR3.

Figure 9.9 shows a block diagram of the noise canceler.

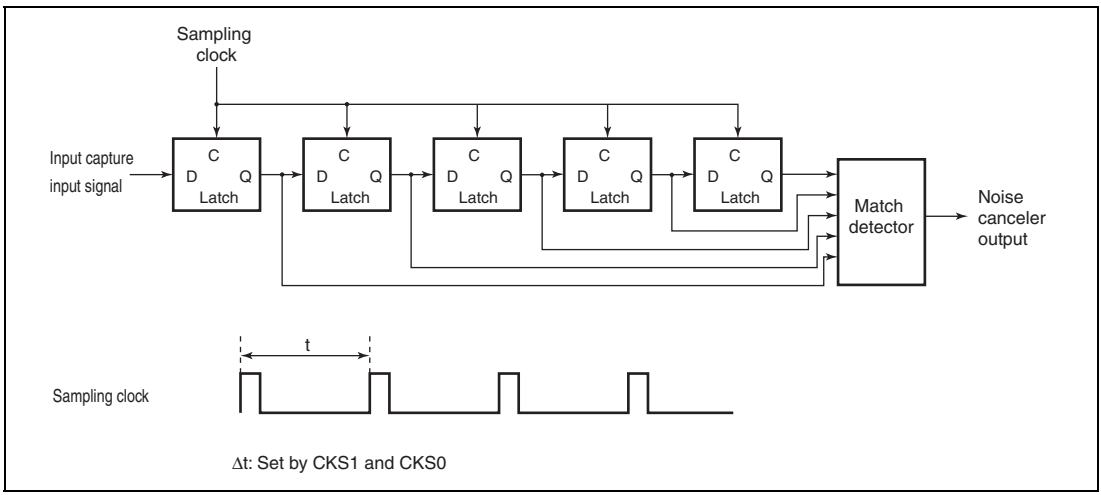



**Figure 9.9 Noise Canceler Block Diagram**

The noise canceler consists of five latch circuits connected in series and a match detector circuit. When the noise cancellation function is not used (NCS = 0), the system clock is selected as the sampling clock. When the noise cancellation function is used (NCS = 1), the sampling clock is the internal clock selected by CKS1 and CKS0 in TMG, the input capture input is sampled on the rising edge of this clock, and the data is judged to be correct when all the latch outputs match. If all the outputs do not match, the previous value is retained. After a reset, the noise canceler output is initialized when the falling edge of the input capture input signal has been sampled five times. Therefore, after making a setting for use of the noise cancellation function, a pulse with at least five times the width of the sampling clock is a dependable input capture signal. Even if noise cancellation is not used, an input capture input signal pulse width of at least $2\phi$ or $2\phi_{SUB}$ is necessary to ensure that input capture operations are performed properly

Note: * An input capture signal may be generated when the NCS bit is modified.

Figure 9.10 shows an example of noise canceler timing.

In this example, high-level input of less than five times the width of the sampling clock at the input capture input pin is eliminated as noise.

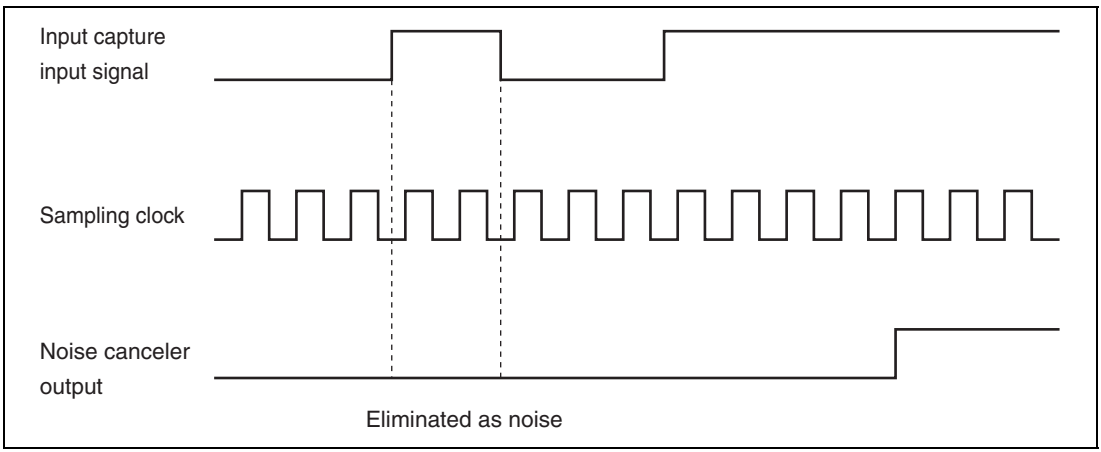



**Figure 9.10 Noise Canceler Timing (Example)**

### 9.5.4 Operation

Timer G is an 8-bit timer with built-in input capture and interval functions.

#### (1) Timer G Functions

Timer G is an 8-bit up-counter with two functions, an input capture timer function and an interval timer function.

The operation of these two functions is described below.

a. Input capture timer operation

When the TMIG bit is set to 1 in port mode register 1 (PMR1), timer G functions as an input capture timer*.

In a reset, timer mode register G (TMG), timer counter G (TCG), input capture register GF (ICRGF), and input capture register GR (ICRGR) are all initialized to H'00.

Following a reset, TCG starts incrementing on the ϕ/64 internal clock.

The input clock can be selected from four internal clock sources by bits CKS1 and CKS0 in TMG.

When a rising edge/falling edge is detected in the input capture signal input from the TMIG pin, the TCG value at that time is transferred to ICRGR/ICRGF. When the edge selected by IIEGS in TMG is input, IRRTG is set to 1 in IRR2, and if the IENTG bit in IENR2 is 1 at this time, an interrupt request is sent to the CPU. For details of the interrupt, see section 3.3, Interrupts.

TCG can be cleared by a rising edge, falling edge, or both edges of the input capture signal, according to the setting of bits CCLR1 and CCLR0 in TMG. If TCG overflows when the input capture signal is high, the OVFH bit is set in TMG; if TCG overflows when the input capture signal is low, the OVFL bit is set in TMG. If the OVIE bit in TMG is 1 when these bits are set, IRRTG is set to 1 in IRR2, and if the IENTG bit in IENR2 is 1, timer G sends an interrupt request to the CPU. For details of the interrupt, see section 3.3, Interrupts.

Timer G has a built-in noise canceler that enables high-frequency component noise to be eliminated from pulses input from the TMIG pin. For details, see section 9.5.3, Noise Canceler.

Note: * An input capture signal may be generated when TMIG is modified.

b. Interval timer operation

When the TMIG bit is cleared to 0 in PMR1, timer G functions as an interval timer. Following a reset, TCG starts incrementing on the ϕ/64 internal clock. The input clock can be selected from four internal clock sources by bits CKS1 and CKS0 in TMG. TCG increments on the selected clock, and when it overflows from H'FF to H'00, the OVFL bit is set to 1 in TMG. If the OVIE bit in TMG is 1 at this time, IRRTG is set to 1 in IRR2, and if the IENTG bit in IENR2 is 1, timer G sends an interrupt request to the CPU. For details of the interrupt, see section 3.3, Interrupts.

### (2) Increment Timing

TCG is incremented by internal clock input. Bits CKS1 and CKS0 in TMG select one of four internal clock sources (ϕ/64, ϕ/32, ϕ/2, or ϕw/4) created by dividing the system clock (ϕ) or watch clock (ϕw).

### (3) Input Capture Input Timing

a. Without noise cancellation function

For input capture input, dedicated input capture functions are provided for rising and falling edges.

Figure 9.11 shows the timing for rising/falling edge input capture input.

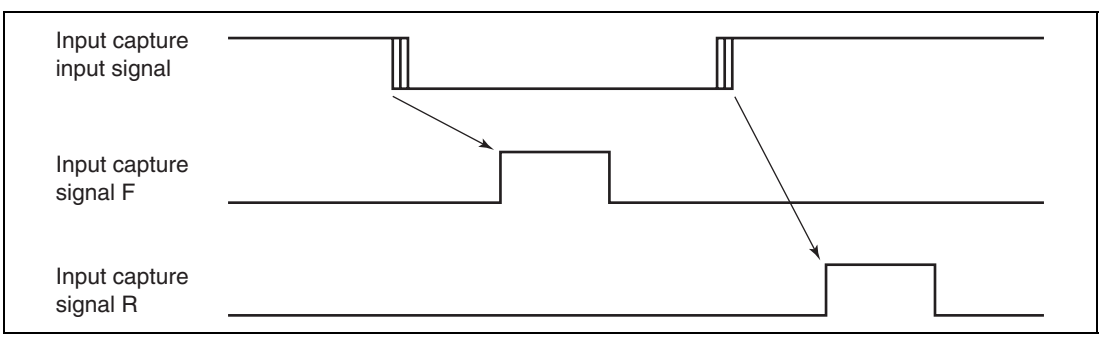



**Figure 9.11 Input Capture Input Timing (without Noise Cancellation Function)**

b. With noise cancellation function

When noise cancellation is performed on the input capture input, the passage of the input capture signal through the noise canceler results in a delay of five sampling clock cycles from the input capture input signal edge.

Figure 9.12 shows the timing in this case.

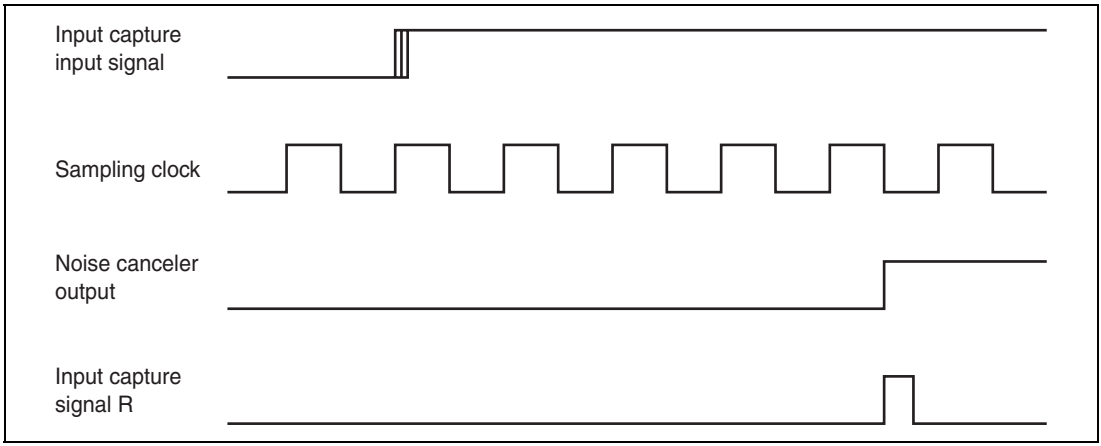



**Figure 9.12 Input Capture Input Timing (with Noise Cancellation Function)**

### (4) Timing of Input Capture by Input Capture Input

Figure 9.13 shows the timing of input capture by input capture input

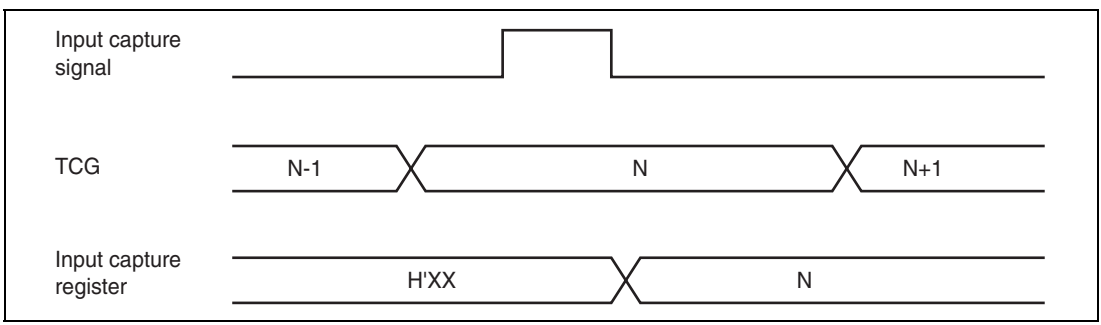



**Figure 9.13 Timing of Input Capture by Input Capture Input**

### (5) TGC Clear Timing

TCG can be cleared by the rising edge, falling edge, or both edges of the input capture input signal.

Figure 9.14 shows the timing for clearing by both edges.

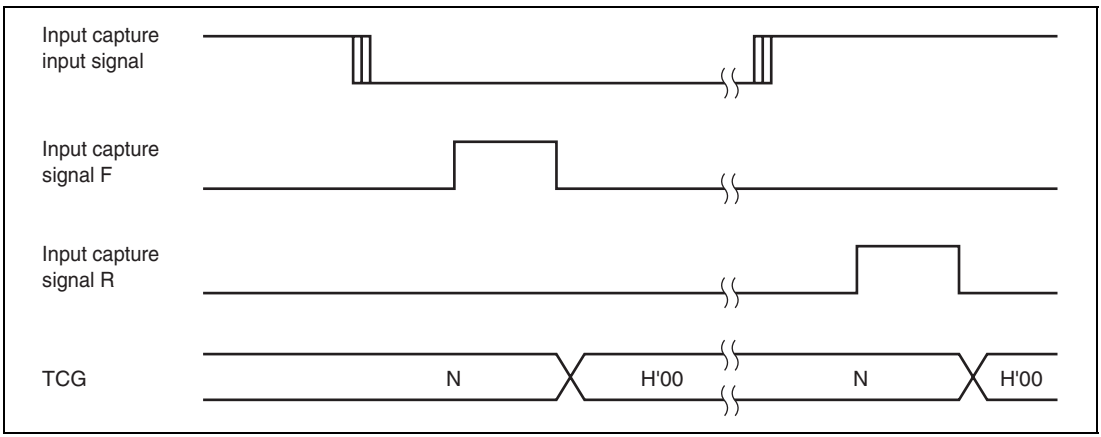



**Figure 9.14 TCG Clear Timing**

### (6) Timer G Operation Modes

Timer G operation modes are shown in table 9.13.

**Table 9.13 Timer G Operation Modes**

| Operation Mode | | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|---|
| TCG | Input capture | Reset | Functions* | Functions* | Functions/ halted* | Functions/ halted* | Functions/ halted* | Halted | Halted |
| | Interval | Reset | Functions* | Functions* | Functions/ halted* | Functions/ halted* | Functions/ halted* | Halted | Halted |
| ICRGF | | Reset | Functions* | Functions* | Functions/ halted* | Functions/ halted* | Functions/ halted* | Held | Held |
| ICRGR | | Reset | Functions* | Functions* | Functions halted* | Functions/ halted* | Functions/ halted* | Held | Held |
| TMG | | Reset | Functions | Held | Held | Functions | Held | Held | Held |

Note: * When $\phi w/4$ is selected as the TCG internal clock in active mode or sleep mode, since the system clock and internal clock are mutually asynchronous, synchronization is maintained by a synchronization circuit. This results in a maximum count cycle error of $1/\phi$ (s). When $\phi w/4$ is selected as the TCG internal clock in watch mode, TCG and the noise canceler operate on the $\phi w/4$ internal clock without regard to the $\phi_{SUB}$ subclock ($\phi w/8$, $\phi w/4$, $\phi w/2$). Note that when another internal clock is selected, TCG and the noise canceler do not operate, and input of the input capture input signal does not result in input capture.

To be operated Timer G in subactive mode or subsleep mode, select $\phi w/4$ for internal clock of TCG and also select $\phi w/2$ for sub clock $\phi_{SUB}$. When another internal clock is selected and when another sub clock ($\phi w/8$, $\phi w/4$) is selected, TCG and noise canceler do not operate.

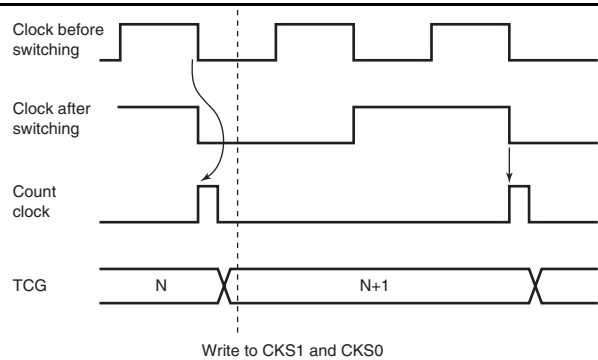
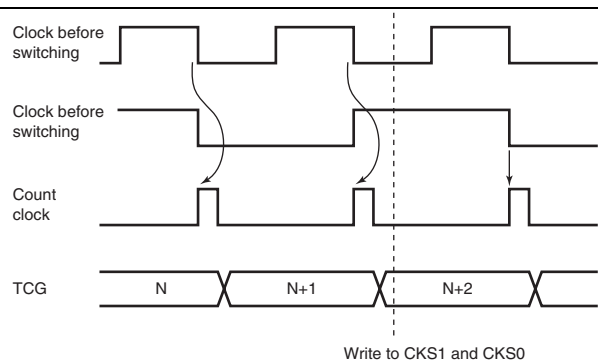
### 9.5.5 Application Notes

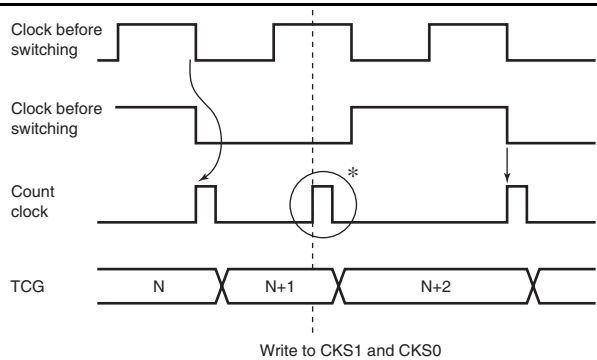
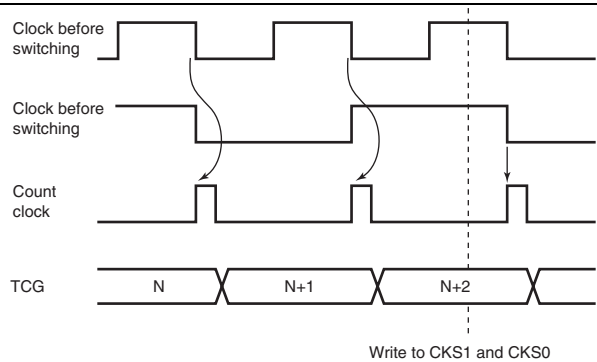
#### (1) Internal Clock Switching and TCG Operation

Depending on the timing, TCG may be incremented by a switch between difference internal clock sources. Table 9.14 shows the relation between internal clock switchover timing (by write to bits CKS1 and CKS0) and TCG operation.

When TCG is internally clocked, an increment pulse is generated on detection of the falling edge of an internal clock signal, which is divided from the system clock (ϕ) or subclock (ϕw). For this reason, in a case like No. 3 in table 9.14 where the switch is from a high clock signal to a low clock signal, the switchover is seen as a falling edge, causing TCG to increment.

**Table 9.14 Internal Clock Switching and TCG Operation**

| No. | Clock Levels Before and After Modifying Bits CKS1 and CKS0 | TCG Operation |
|---|---|---|
| 1 | Goes from low level to low level |  |
| 2 | Goes from low level to high level |  |

| No. | Clock Levels Before and After Modifying Bits CKS1 and CKS0 | TCG Operation |
|---|---|---|
| 3 | Goes from high level to low level |  |
| 4 | Goes from high level to high level |  |

Note: * The switchover is seen as a falling edge, and TCG is incremented.

### (2) Notes on Port Mode Register Modification

The following points should be noted when a port mode register is modified to switch the input capture function or the input capture input noise canceler function.

- Switching input capture input pin function

  Note that when the pin function is switched by modifying TMIG in port mode register 1 (PMR1), which performs input capture input pin control, an edge will be regarded as having been input at the pin even though no valid edge has actually been input. Input capture input signal input edges, and the conditions for their occurrence, are summarized in table 9.15.

**Table 9.15 Input Capture Input Signal Input Edges Due to Input Capture Input Pin Switching, and Conditions for Their Occurrence**

| Input Capture Input Signal Input Edge | Conditions |
|---|---|
| Generation of rising edge | When TMIG is modified from 0 to 1 while the TMIG pin is high |
| | When NCS is modified from 0 to 1 while the TMIG pin is high, then TMIG is modified from 0 to 1 before the signal is sampled five times by the noise canceler |
| Generation of falling edge | When TMIG is modified from 1 to 0 while the TMIG pin is high |
| | When NCS is modified from 0 to 1 while the TMIG pin is low, then TMIG is modified from 0 to 1 before the signal is sampled five times by the noise canceler |
| | When NCS is modified from 0 to 1 while the TMIG pin is high, then TMIG is modified from 1 to 0 after the signal is sampled five times by the noise canceler |

Note: When the $P1_3$ pin is not set as an input capture input pin, the timer G input capture input signal is low.

- Switching input capture input noise canceler function

  When performing noise canceler function switching by modifying NCS in port mode register 3 (PMR3), which controls the input capture input noise canceler, TMIG should first be cleared to 0. Note that if NCS is modified without first clearing TMIG, an edge will be regarded as having been input at the pin even though no valid edge has actually been input. Input capture input signal input edges, and the conditions for their occurrence, are summarized in table 9.16.

**Table 9.16 Input Capture Input Signal Input Edges Due to Noise Canceler Function Switching, and Conditions for Their Occurrence**

| Input Capture Input Signal Input Edge | Conditions |
|---|---|
| Generation of rising edge | When the TMIG pin level is switched from low to high while TMIG is set to 1, then NCS is modified from 0 to 1 before the signal is sampled five times by the noise canceler |
| Generation of falling edge | When the TMIG pin level is switched from high to low while TMIG is set to 1, then NCS is modified from 1 to 0 before the signal is sampled five times by the noise canceler |

When the pin function is switched and an edge is generated in the input capture input signal, if this edge matches the edge selected by the input capture interrupt select (IIEGS) bit, the interrupt request flag will be set to 1. The interrupt request flag should therefore be cleared to 0 before use. Figure 9.15 shows the procedure for port mode register manipulation and interrupt request flag clearing. When switching the pin function, set the interrupt-disabled state before manipulating the port mode register, then, after the port mode register operation has been performed, wait for the time required to confirm the input capture input signal as an input capture signal (at least two system clocks when the noise canceler is not used; at least five sampling clocks when the noise canceler is used), before clearing the interrupt enable flag to 0. There are two ways of preventing interrupt request flag setting when the pin function is switched: by controlling the pin level so that the conditions shown in tables 9.15 and 9.16 are not satisfied, or by setting the opposite of the generated edge in the IIEGS bit in TMG.

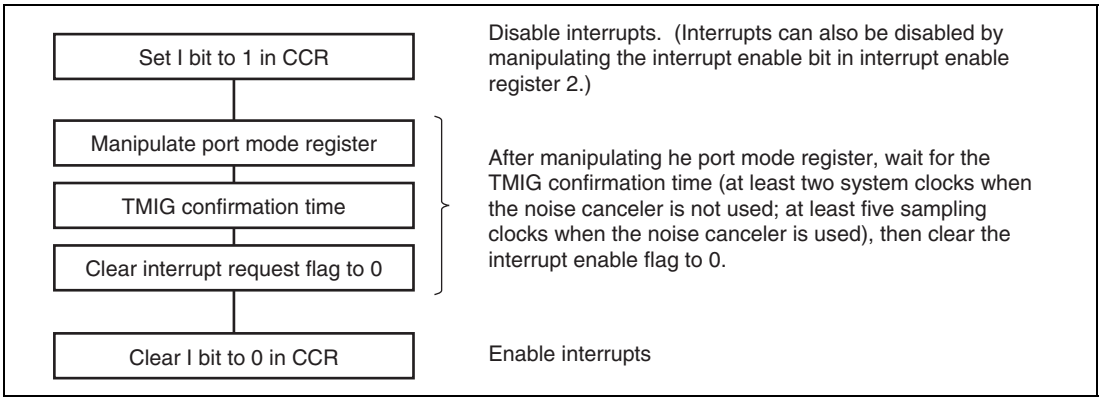



**Figure 9.15 Port Mode Register Manipulation and Interrupt Enable Flag Clearing Procedure**

### 9.5.6 Timer G Application Example

Using timer G, it is possible to measure the high and low widths of the input capture input signal as absolute values. For this purpose, CCLR1 and CCLR0 should both be set to 1 in TMG.

Figure 9.16 shows an example of the operation in this case.

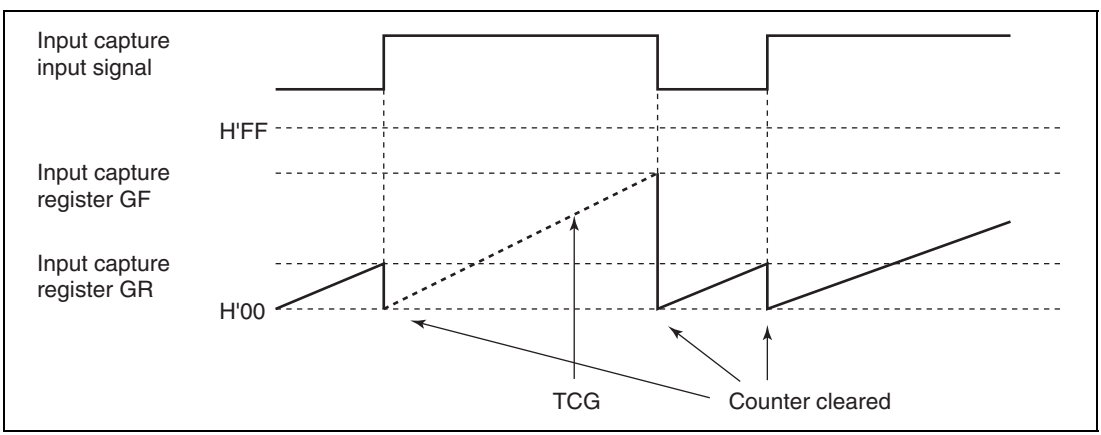



**Figure 9.16 Timer G Application Example**

## 9.6 Watchdog Timer

### 9.6.1 Overview

The watchdog timer has an 8-bit counter that is incremented by an input clock. If a system runaway allows the counter value to overflow before being rewritten, the watchdog timer can reset the chip internally.

#### (1) Features

Features of the watchdog timer are given below.

- Incremented by internal clock source (ϕ/8192 or ϕw/32).
- A reset signal is generated when the counter overflows. The overflow period can be set from from 1 to 256 times 8192/ϕ or 32/ϕw (from approximately 4 ms to 1000 ms when ϕ = 2.00 MHz).
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

#### (2) Block Diagram

Figure 9.17 shows a block diagram of the watchdog timer.

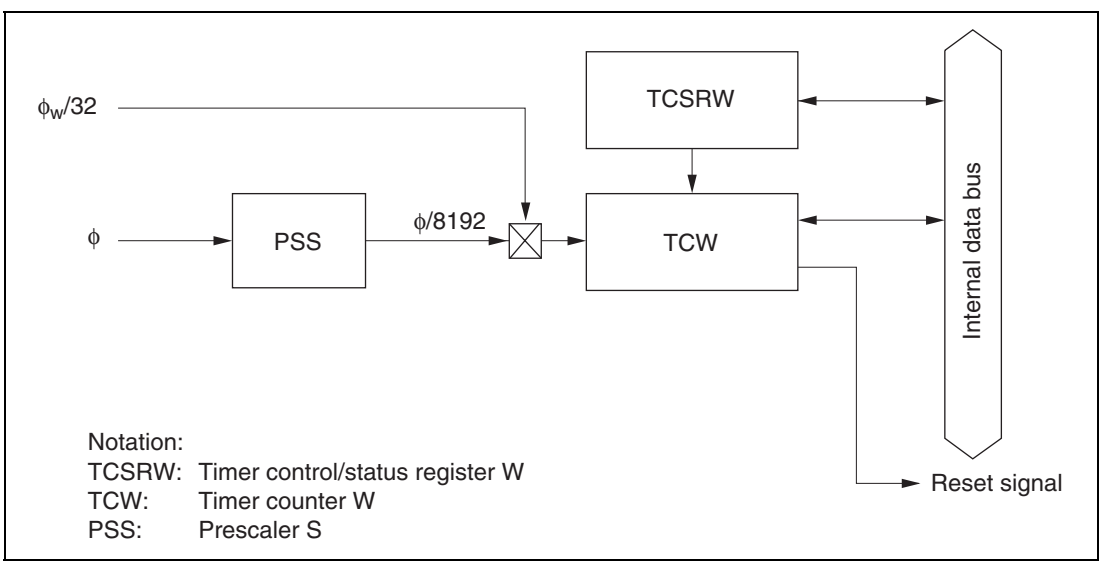



**Figure 9.17 Block Diagram of Watchdog Timer**

### (3) Register Configuration

Table 9.17 shows the register configuration of the watchdog timer.

**Table 9.17 Watchdog Timer Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer control/status register W | TCSRW | R/W | H'AA | H'FFB2 |
| Timer counter W | TCW | R/W | H'00 | H'FFB3 |
| Clock stop register 2 | CKSTPR2 | R/W | H'FF | H'FFFB |
| Port mode register 3 | PMR3 | R/W | H'04 | H'FFCA |

## 9.6.2 Register Descriptions

### (1) Timer Control/Status Register W (TCSRW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | B6WI | TCWE | B4WI | TCSRWE | B2WI | WDON | B0WI | WRST |
| Initial value | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Read/Write | R | R/(W)* | R | R/(W)* | R | R/(W)* | R | R/(W)* |

Note: * Write is permitted only under certain conditions, which are given in the descriptions of the individual bits.

TCSRW is an 8-bit read/write register that controls write access to TCW and TCSRW itself, controls watchdog timer operations, and indicates operating status.

**Bit 7:** Bit 6 write inhibit (B6WI)

Bit 7 controls the writing of data to bit 6 in TCSRW.

| Bit 7 B6WI | Description | |
|---|---|---|
| 0 | Bit 6 is write-enabled | |
| 1 | Bit 6 is write-protected | (initial value) |

This bit is always read as 1. Data written to this bit is not stored.

**Bit 6:** Timer counter W write enable (TCWE)

Bit 6 controls the writing of data to TCW.

| Bit 6<br>TCWE | Description | |
|---|---|---|
| 0 | Data cannot be written to TCW | (initial value) |
| 1 | Data can be written to TCW | |

**Bit 5:** Bit 4 write inhibit (B4WI)

Bit 5 controls the writing of data to bit 4 in TCSRW.

| Bit 5<br>B4WI | Description | |
|---|---|---|
| 0 | Bit 4 is write-enabled | |
| 1 | Bit 4 is write-protected | (initial value) |

This bit is always read as 1. Data written to this bit is not stored.

**Bit 4:** Timer control/status register W write enable (TCSRWE)

Bit 4 controls the writing of data to TCSRW bits 2 and 0.

| Bit 4<br>TCSRWE | Description | |
|---|---|---|
| 0 | Data cannot be written to bits 2 and 0 | (initial value) |
| 1 | Data can be written to bits 2 and 0 | |

**Bit 3:** Bit 2 write inhibit (B2WI)

Bit 3 controls the writing of data to bit 2 in TCSRW.

| Bit 3<br>B2WI | Description | |
|---|---|---|
| 0 | Bit 2 is write-enabled | |
| 1 | Bit 2 is write-protected | (initial value) |

This bit is always read as 1. Data written to this bit is not stored.

**Bit 2:** Watchdog timer on (WDON)

Bit 2 enables watchdog timer operation.

| Bit 2<br>WDON | Description | |
|---|---|---|
| 0 | Watchdog timer operation is disabled<br>Clearing condition:<br>Reset, or when TCSRWE = 1 and 0 is written in both B2WI and WDON | (initial value) |
| 1 | Watchdog timer operation is enabled<br>Setting condition:<br>When TCSRWE = 1 and 0 is written in B2WI and 1 is written in WDON | |

Counting starts when this bit is set to 1, and stops when this bit is cleared to 0.

**Bit 1:** Bit 0 write inhibit (B0WI)

Bit 1 controls the writing of data to bit 0 in TCSRW.

| Bit 1<br>B0WI | Description | |
|---|---|---|
| 0 | Bit 0 is write-enabled | |
| 1 | Bit 0 is write-protected | (initial value) |

This bit is always read as 1. Data written to this bit is not stored.

**Bit 0:** Watchdog timer reset (WRST)

Bit 0 indicates that TCW has overflowed, generating an internal reset signal. The internal reset signal generated by the overflow resets the entire chip. WRST is cleared to 0 by a reset from the $\overline{\text{RES}}$ pin, or when software writes 0.

| Bit 0<br>WRST | Description |
|---|---|
| 0 | Clearing condition:<br>Reset by $\overline{\text{RES}}$ pin<br>When TCSRWE = 1, and 0 is written in both B0WI and WRST |
| 1 | Setting condition:<br>When TCW overflows and an internal reset signal is generated |

### (2) Timer Counter W (TCW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCW7 | TCW6 | TCW5 | TCW4 | TCW3 | TCW2 | TCW1 | TCW0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

TCW is an 8-bit read/write up-counter, which is incremented by internal clock input. The input clock is ϕ/8192 or ϕw/32. The TCW value can always be written or read by the CPU.

When TCW overflows from H'FF to H'00, an internal reset signal is generated and WRST is set to 1 in TCSRW. Upon reset, TCW is initialized to H'00.

### (3) Clock Stop Register 2 (CKSTPR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | AECKSTP | WDCKSTP | PWCKSTP | LDCKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

CKSTPR2 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to the watchdog timer is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 2:** Watchdog timer module standby mode control (WDCKSTP)

Bit 2 controls setting and clearing of module standby mode for the watchdog timer.

| WDCKSTP | Description | |
|---|---|---|
| 0 | Watchdog timer is set to module standby mode | |
| 1 | Watchdog timer module standby mode is cleared | (initial value) |

Note: WDCKSTP is valid when the WDON bit is cleared to 0 in timer control/status register W (TCSRW). If WDCKSTP is set to 0 while WDON is set to 1 (during watchdog timer operation), 0 will be set in WDCKSTP but the watchdog timer will continue its watchdog function and will not enter module standby mode. When the watchdog function ends and WDON is cleared to 0 by software, the WDCKSTP setting will become valid and the watchdog timer will enter module standby mode.

### (4) Port Mode Register 3 (PMR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AEVL | AEVH | WDCKS | NCS | IRQ0 | — | UD | PWM |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

PMR3 is an 8-bit read/write register, mainly controlling the selection of pin functions for port 3 pins. Only the bit relating to the watchdog timer is described here. For details of the other bits, see section 8, I/O Ports.

**Bit 5:** Watchdog timer source clock select (WDCKS)

| WDCKS | Description | |
|---|---|---|
| 0 | $\phi$/8192 selected | (initial value) |
| 1 | $\phi$w/32 selected | |

### 9.6.3 Timer Operation

The watchdog timer has an 8-bit counter (TCW) that is incremented by clock input (ϕ/8192 or ϕw/32). The input clock is selected by bit WDCKS in port mode register 3 (PMR3): ϕ/8192 is selected when WDCKS is cleared to 0, and ϕw/32 when set to 1. When TCSRWE = 1 in TCSRW, if 0 is written in B2WI and 1 is simultaneously written in WDON, TCW starts counting up. When the TCW count reaches H'FF, the next clock input causes the watchdog timer to overflow, and an internal reset signal is generated one reference clock (ϕ or $\phi_{SUB}$) cycle later. The internal reset signal is output for 512 clock cycles of the $\phi_{OSC}$ clock. It is possible to write to TCW, causing TCW to count up from the written value. The overflow period can be set in the range from 1 to 256 input clocks, depending on the value written in TCW.

Figure 9.18 shows an example of watchdog timer operations.

Example: $\phi$ = 2 MHz and the desired overflow period is 30 ms.

$$\frac{2 \times 10^6}{8192} \times 30 \times 10^{-3} = 7.3$$

The value set in TCW should therefore be 256 – 8 = 248 (H'F8).

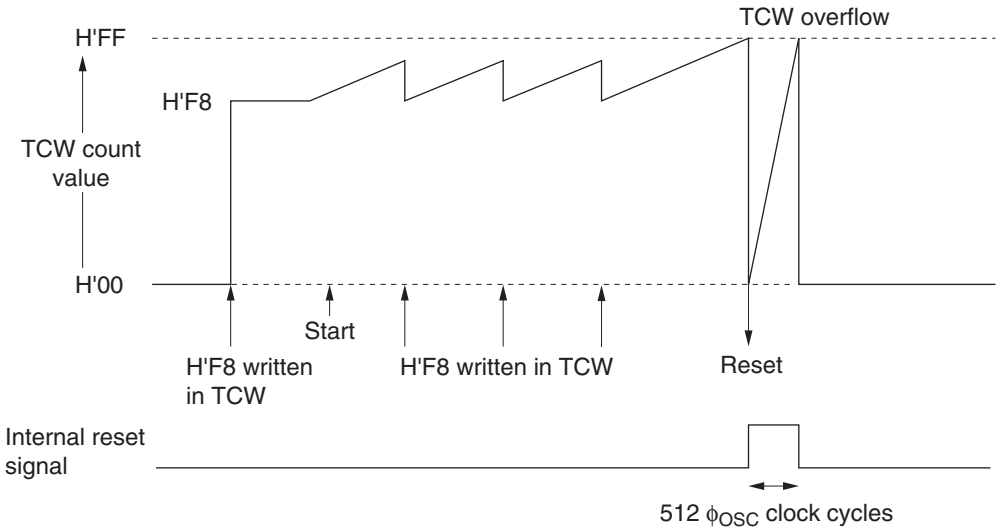



**Figure 9.18 Typical Watchdog Timer Operations (Example)**

### 9.6.4 Watchdog Timer Operation States

Table 9.18 summarizes the watchdog timer operation states.

**Table 9.18 Watchdog Timer Operation States**

| Operation Mode | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|
| TCW | Reset | Functions | Functions | Halted | Functions/ Halted* | Halted | Halted | Halted |
| TCSRW | Reset | Functions | Functions | Retained | Functions/ Halted* | Retained | Retained | Retained |

Note: * Functions when $\phi$w/32 is selected as the input clock.

## 9.7 Asynchronous Event Counter (AEC)

### 9.7.1 Overview

The asynchronous event counter is incremented by external event clock input.

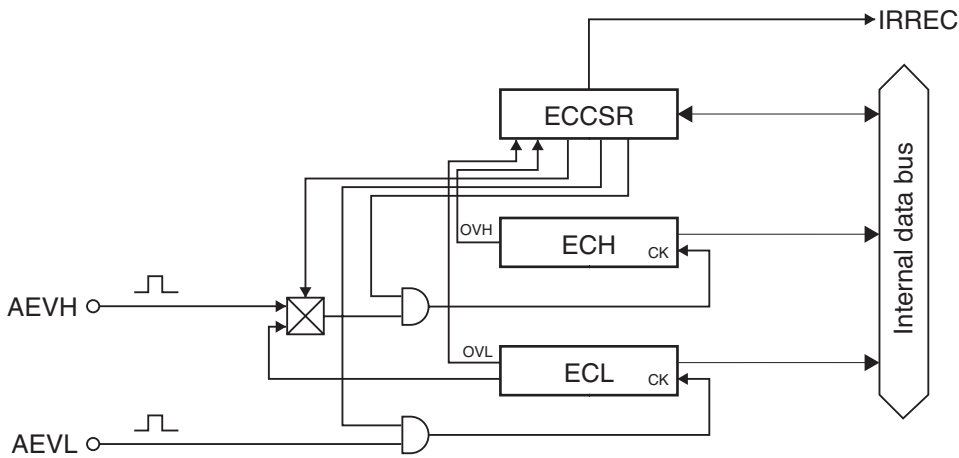
#### (1) Features

Features of the asynchronous event counter are given below.

- Can count asynchronous events
  Can count external events input asynchronously without regard to the operation of base clocks $\phi$ and $\phi_{SUB}$.
  The counter has a 16-bit configuration, enabling it to count up to 65536 ($2^{16}$) events.
  — Can also be used as two independent 8-bit event counter channels.
  — Counter resetting and halting of the count-up function controllable by software
  — Automatic interrupt generation on detection of event counter overflow
  — Use of module standby mode enables this module to be placed in standby mode independently when not used.

### (2) Block Diagram

Figure 9.19 shows a block diagram of the asynchronous event counter.




Legend:
ECCSR: Event counter control/status register
ECH: Event counter H
ECL: Event counter L
AEVH: Asynchronous event input H
AEVL: Asynchronous event input L
IRREC: Event counter overflow interrupt request flag

**Figure 9.19 Block Diagram of Asynchronous Event Counter**

### (3) Pin Configuration

Table 9.19 shows the asynchronous event counter pin configuration.

**Table 9.19 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Asynchronous event input H | AEVH | Input | Event input pin for input to event counter H |
| Asynchronous event input L | AEVL | Input | Event input pin for input to event counter L |

### (4) Register Configuration

Table 9.20 shows the register configuration of the asynchronous event counter.

**Table 9.20 Asynchronous Event Counter Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Event counter control/status register | ECCSR | R/W | H'00 | H'FF95 |
| Event counter H | ECH | R | H'00 | H'FF96 |
| Event counter L | ECL | R | H'00 | H'FF97 |
| Clock stop register 2 | CKSTPR2 | R/W | H'FF | H'FFFB |

## 9.7.2 Register Descriptions

### (1) Event Counter Control/Status Register (ECCSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OVH | OVL | — | CH2 | CUEH | CUEL | CRCH | CRCL |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W* | R/W* | R/W | R/W | R/W | R/W | R/W | R/W |

Note: * Bits 7 and 6 can only be written with 0, for flag clearing.

ECCSR is an 8-bit read/write register that controls counter overflow detection, counter resetting, and halting of the count-up function.

ECCSR is initialized to H'00 upon reset.

**Bit 7:** Counter overflow flag H (OVH)

Bit 7 is a status flag indicating that ECH has overflowed from H'FF to H'00. This flag is set when ECH overflows. It is cleared by software but cannot be set by software. OVH is cleared by reading it when set to 1, then writing 0.

When ECH and ECL are used as a 16-bit event counter with CH2 cleared to 0, OVH functions as a status flag indicating that the 16-bit event counter has overflowed from H'FFFF to H'0000.

| Bit 7<br>OVH | Description | |
|---|---|---|
| 0 | ECH has not overflowed<br>Clearing condition:<br>After reading OVH = 1, cleared by writing 0 to OVH | (initial value) |
| 1 | ECH has overflowed<br>Setting condition:<br>Set when ECH overflows from H'FF to H'00 | |

**Bit 6:** Counter overflow flag L (OVL)

Bit 6 is a status flag indicating that ECL has overflowed from H'FF to H'00. This flag is set when ECL overflows. It is cleared by software but cannot be set by software. OVL is cleared by reading it when set to 1, then writing 0.

| Bit 6<br>OVL | Description | |
|---|---|---|
| 0 | ECL has not overflowed<br>Clearing condition:<br>After reading OVL = 1, cleared by writing 0 to OVL | (initial value) |
| 1 | ECL has overflowed<br>Setting condition:<br>Set when ECL overflows from H'FF to H'00 while CH2 is set to 1 | |

**Bit 5:** Reserved bit

Bit 5 is reserved; it can be read and written, and is initialized to 0 upon reset.

**Bit 4:** Channel select (CH2)

Bit 4 selects whether ECH and ECL are used as a single-channel 16-bit event counter or as two independent 8-bit event counter channels. When CH2 is cleared to 0, ECH and ECL function as a 16-bit event counter which is incremented each time an event clock is input to the AEVL pin as asynchronous event input. In this case, the overflow signal from ECL is selected as the ECH input clock. When CH2 is set to 1, ECH and ECL function as independent 8-bit event counters which are incremented each time an event clock is input to the AEVH or AEVL pin, respectively, as asynchronous event input.

| Bit 4<br>CH2 | Description |
|---|---|
| 0 | ECH and ECL are used together as a single-channel 16-bit event counter (initial value) |
| 1 | ECH and ECL are used as two independent 8-bit event counter channels |

**Bit 3:** Count-up enable H (CUEH)

Bit 3 enables event clock input to ECH. When 1 is written to this bit, event clock input is enabled and increments the counter. When 0 is written to this bit, event clock input is disabled and the ECH value is held. The AEVH pin or the ECL overflow signal can be selected as the event clock source by bit CH2.

| Bit 3<br>CUEH | Description |
|---|---|
| 0 | ECH event clock input is disabled (initial value)<br>ECH value is held |
| 1 | ECH event clock input is enabled |

**Bit 2:** Count-up enable L (CUEL)

Bit 2 enables event clock input to ECL. When 1 is written to this bit, event clock input is enabled and increments the counter. When 0 is written to this bit, event clock input is disabled and the ECL value is held.

| Bit 2<br>CUEL | Description |
|---|---|
| 0 | ECL event clock input is disabled (initial value)<br>ECL value is held |
| 1 | ECL event clock input is enabled |

**Bit 1:** Counter reset control H (CRCH)

Bit 1 controls resetting of ECH. When this bit is cleared to 0, ECH is reset. When 1 is written to this bit, the counter reset is cleared and the ECH count-up function is enabled.

| Bit 1<br>CRCH | Description | |
|---|---|---|
| 0 | ECH is reset | (initial value) |
| 1 | ECH reset is cleared and count-up function is enabled | |

**Bit 0:** Counter reset control L (CRCL)

Bit 0 controls resetting of ECL. When this bit is cleared to 0, ECL is reset. When 1 is written to this bit, the counter reset is cleared and the ECL count-up function is enabled.

| Bit 0<br>CRCL | Description | |
|---|---|---|
| 0 | ECL is reset | (initial value) |
| 1 | ECL reset is cleared and count-up function is enabled | |

### (2) Event Counter H (ECH)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ECH7 | ECH6 | ECH5 | ECH4 | ECH3 | ECH2 | ECH1 | ECH0 |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

ECH is an 8-bit read-only up-counter that operates either as an independent 8-bit event counter or as the upper 8-bit up-counter of a 16-bit event counter configured in combination with ECL. Either the external asynchronous event AEVH pin or the overflow signal from lower 8-bit counter ECL can be selected as the input clock source by bit CH2. ECH can be cleared to H'00 by software, and is also initialized to H'00 upon reset.

### (3) Event Counter L (ECL)

ECL is an 8-bit read-only up-counter that operates either as an independent 8-bit event counter or as the lower 8-bit up-counter of a 16-bit event counter configured in combination with ECH. The event clock from the external asynchronous event AEVL pin is used as the input clock source. ECL can be cleared to H'00 by software, and is also initialized to H'00 upon reset.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ECL7 | ECL6 | ECL5 | ECL4 | ECL3 | ECL2 | ECL1 | ECL0 |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

### (4) Clock Stop Register 2 (CKSTPR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | AECKSTP | WDCKSTP | PWCKSTP | LDCKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

CKSTPR2 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to the asynchronous event counter is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 3:** Asynchronous event counter module standby mode control (AECKSTP)

Bit 3 controls setting and clearing of module standby mode for the asynchronous event counter.

| AECKSTP | Description | |
|---|---|---|
| 0 | Asynchronous event counter is set to module standby mode | |
| 1 | Asynchronous event counter module standby mode is cleared | (initial value) |

### 9.7.3 Operation

#### (1) 16-bit Event Counter Operation

When bit CH2 is cleared to 0 in ECCSR, ECH and ECL operate as a 16-bit event counter. Figure 9.20 shows an example of the software processing when ECH and ECL are used as a 16-bit event counter.

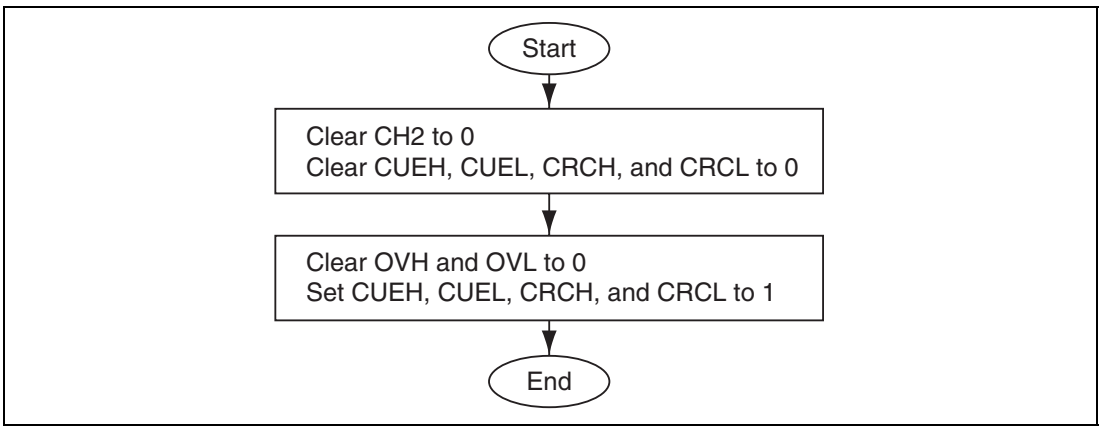



**Figure 9.20 Example of Software Processing when Using ECH and ECL as 16-Bit Event Counter**

As CH2 is cleared to 0 by a reset, ECH and ECL operate as a 16-bit event counter after a reset. They can also be used as a 16-bit event counter by carrying out the software processing shown in the example in figure 9.20. The operating clock source is asynchronous event input from the AEVL pin. When the next clock is input after the count value reaches H'FF in both ECH and ECL, ECH and ECL overflow from H'FFFF to H'0000, the OVH flag is set to 1 in ECCSR, the ECH and ECL count values each return to H'00, and counting up is restarted. When overflow occurs, the IRREC bit is set to 1 in IRR2. If the IENEC bit in IENR2 is 1 at this time, an interrupt request is sent to the CPU.

### (2) 8-bit Event Counter Operation

When bit CH2 is set to 1 in ECCSR, ECH and ECL operate as independent 8-bit event counters. Figure 9.21 shows an example of the software processing when ECH and ECL are used as 8-bit event counters.

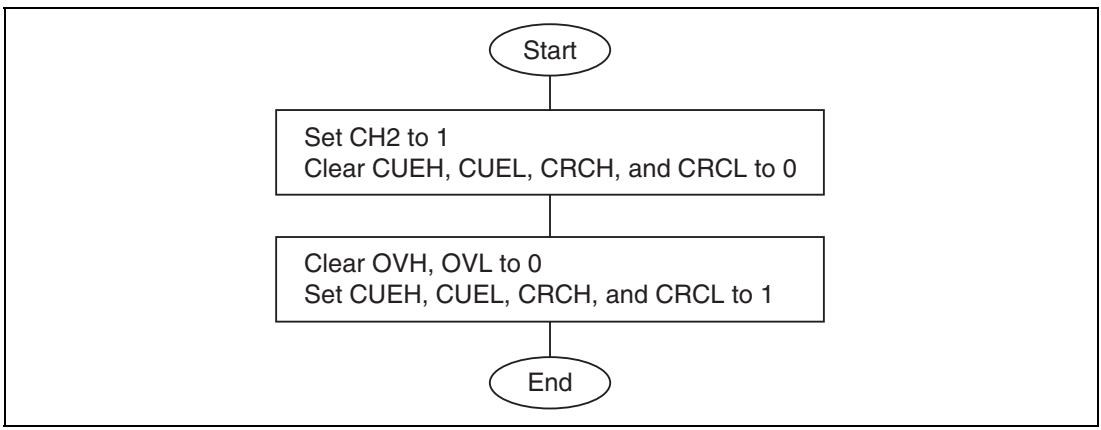



**Figure 9.21 Example of Software Processing when Using ECH and ECL as 8-Bit Event Counters**

ECH and ECL can be used as 8-bit event counters by carrying out the software processing shown in the example in figure 9.21. The 8-bit event counter operating clock source is asynchronous event input from the AEVH pin for ECH, and asynchronous event input from the AEVL pin for ECL. When the next clock is input after the ECH count value reaches H'FF, ECH overflows, the OVH flag is set to 1 in ECCSR, the ECH count value returns to H'00, and counting up is restarted. Similarly, when the next clock is input after the ECL count value reaches H'FF, ECL overflows, the OVL flag is set to 1 in ECCSR, the ECL count value returns to H'00, and counting up is restarted. When overflow occurs, the IRREC bit is set to 1 in IRR2. If the IENEC bit in IENR2 is 1 at this time, an interrupt request is sent to the CPU.

### 9.7.4 Asynchronous Event Counter Operation Modes

Asynchronous event counter operation modes are shown in table 9.21.

**Table 9.21 Asynchronous Event Counter Operation Modes**

| Operation Mode | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|
| ECCSR | Reset | Functions | Functions | Held* | Functions | Functions | Held* | Held |
| ECH | Reset | Functions | Functions | Functions* | Functions | Functions | Functions* | Halted |
| ECL | Reset | Functions | Functions | Functions* | Functions | Functions | Functions* | Halted |

Note: * When an asynchronous external event is input, the counter increments but the counter overflow H/L flags are not affected.

### 9.7.5 Application Notes

1. When reading the values in ECH and ECL, the correct value will not be returned if the event counter increments during the read operation. Therefore, if the counter is being used in the 8-bit mode, clear bits CUEH and CUEL in ECCSR to 0 before reading ECH or ECL. If the counter is being used in the 16-bit mode, clear CUEL only to 0 before reading ECH or ECL.

2. The maximum clock frequency to be input to the AEVH and AEVL pins is 16 MHz. In addition, ensure that the high and low widths of the clock are at least 32 ns. The duty cycle is immaterial.

| Mode | | | Maximum AEVH/AEVL Pin Input Clock Frequency |
|---|---|---|---|
| 16-bit mode | | | $V_{cc}$ = 2.7 to 5.5 V/16 MHz |
| 8-bit mode | Active (high-speed), sleep (high-speed) | | |
| 8-bit mode | Active (medium-speed), sleep (medium-speed) | (ϕ/16) | $2 \cdot f_{osc}$ |
| | | (ϕ/32) | $f_{osc}$ |
| | | (ϕ/64) | $1/2 \cdot f_{osc}$ |
| | $f_{osc}$ = 1 MHz to 16 MHz | (ϕ/128) | $1/4 \cdot f_{osc}$ |
| 8-bit mode | Watch, subactive, subsleep, standby | (ϕw/2) | 1000 kHz |
| | | (ϕw/4) | 500 kHz |
| | $\phi_w$ = 32.768 kHz or 38.4 kHz | (ϕw/8) | 250 kHz |

3. When AEC uses with 16-bit mode, set CUEH in ECCSR to “1” first, set CRCH in ECCSR to “1” second, or set both CUEH and CRCH to “1” at same time before clock entry. While AEC is operating on 16-bit mode, do not change CUEH. Otherwise, ECH will be miscounted up. Also, to reset the counter, clear CRCH and CRCL to 0 simultaneously or clear CRCL and CRCH to 0 sequentially, in that order.

# Section 10 Serial Communication Interface

## 10.1 Overview

This LSI is provided with two serial communication interfaces, SCI3-1 and SCI3-2. These two SCIs have identical functions. In this manual, the generic term SCI3 is used to refer to both SCIs.

Serial communication interface 3 (SCI3) can carry out serial data communication in either asynchronous or synchronous mode.

### 10.1.1 Features

Features of SCI3 are listed below.

- Choice of asynchronous or synchronous mode for serial data communication
  - Asynchronous mode

    Serial data communication is performed asynchronously, with synchronization provided character by character. In this mode, serial data can be exchanged with standard asynchronous communication LSIs such as a Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA).

    There is a choice of 16 data transfer formats.

| | |
|---|---|
| Data length | 7, 8, 5 bits |
| Stop bit length | 1 or 2 bits |
| Parity | Even, odd, or none |
| Receive error detection | Parity, overrun, and framing errors |
| Break detection | Break detected by reading the $RXD_{3x}$ pin level directly when a framing error occurs |

— Synchronous mode

Serial data communication is synchronized with a clock. In his mode, serial data can be exchanged with another LSI that has a synchronous communication function.

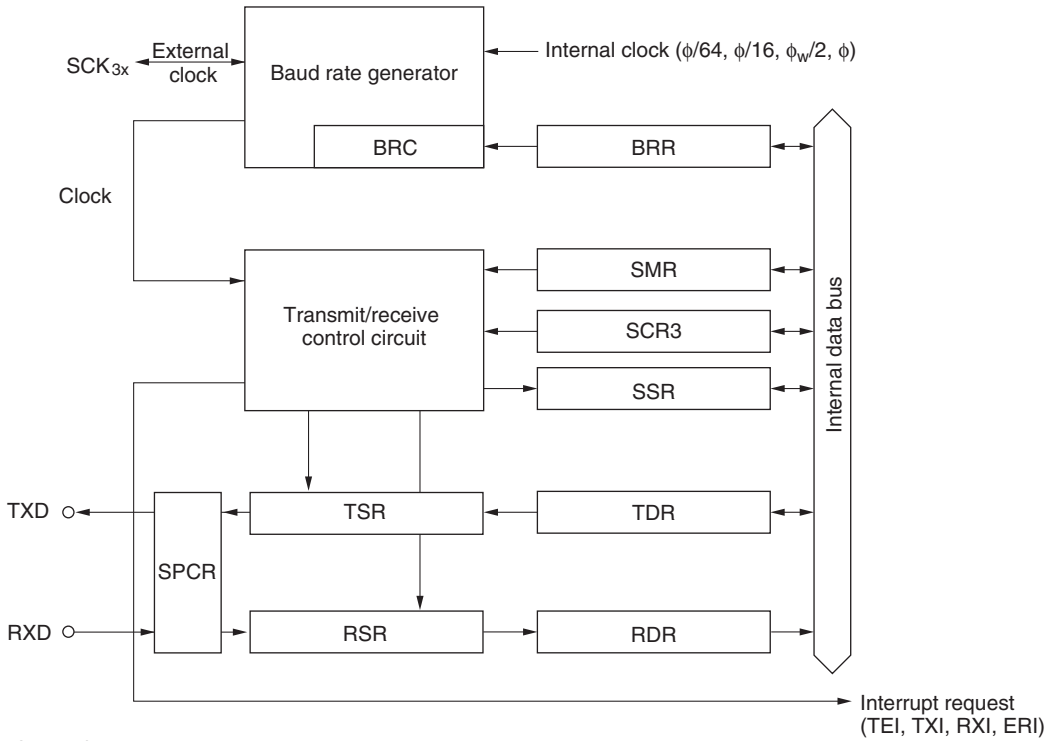
| | |
|---|---|
| Data length | 8 bits |
| Receive error detection | Overrun errors |

- Full-duplex communication

  Separate transmission and reception units are provided, enabling transmission and reception to be carried out simultaneously. The transmission and reception units are both double-buffered, allowing continuous transmission and reception.
- On-chip baud rate generator, allowing any desired bit rate to be selected
- Choice of an internal or external clock as the transmit/receive clock source
- Six interrupt sources: transmit end, transmit data empty, receive data full, overrun error, framing error, and parity error

### 10.1.2 Block Diagram

Figure 10.1 shows a block diagram of SCI3.




Legend:

| | |
|---|---|
| RSR: | Receive shift register |
| RDR: | Receive data register |
| TSR: | Transmit shift register |
| TDR: | Transmit data register |
| SMR: | Serial mode register |
| SCR3: | Serial control register 3 |
| SSR: | Serial status register |
| BRR: | Bit rate register |
| BRC: | Bit rate counter |
| SPCR: | Serial port control register |

**Figure 10.1 SCI3 Block Diagram**

### 10.1.3 Pin Configuration

Table 10.1 shows the SCI3 pin configuration.

**Table 10.1 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| SCI3 clock | $SCK_{3x}$ | I/O | SCI3 clock input/output |
| SCI3 receive data input | $RXD_{3x}$ | Input | SCI3 receive data input |
| SCI3 transmit data output | $TXD_{3x}$ | Output | SCI3 transmit data output |

### 10.1.4 Register Configuration

Table 10.2 shows the SCI3 register configuration.

**Table 10.2 Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Serial mode register | SMR | R/W | H'00 | H'FFA8/FF98 |
| Bit rate register | BRR | R/W | H'FF | H'FFA9/FF99 |
| Serial control register 3 | SCR3 | R/W | H'00 | H'FFAA/FF9A |
| Transmit data register | TDR | R/W | H'FF | H'FFAB/FF9B |
| Serial status register | SSR | R/W | H'84 | H'FFAC/FF9C |
| Receive data register | RDR | R | H'00 | H'FFAD/FF9D |
| Transmit shift register | TSR | Protected | — | — |
| Receive shift register | RSR | Protected | — | — |
| Bit rate counter | BRC | Protected | — | — |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |
| Serial port control register | SPCR | R/W | H'C0 | H'FF91 |

## 10.2 Register Descriptions

### 10.2.1 Receive Shift Register (RSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Read/Write | — | — | — | — | — | — | — | — |

RSR is a register used to receive serial data. Serial data input to RSR from the $RXD_{3x}$ pin is set in the order in which it is received, starting from the LSB (bit 0), and converted to parallel data. When one byte of data is received, it is transferred to RDR automatically.

RSR cannot be read or written directly by the CPU.

### 10.2.2 Receive Data Register (RDR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RDR7 | RDR6 | RDR5 | RDR4 | RDR3 | RDR2 | RDR1 | RDR0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

RDR is an 8-bit register that stores received serial data.

When reception of one byte of data is finished, the received data is transferred from RSR to RDR, and the receive operation is completed. RSR is then able to receive data. RSR and RDR are double-buffered, allowing consecutive receive operations.

RDR is a read-only register, and cannot be written by the CPU.

RDR is initialized to H'00 upon reset, and in standby, module standby or watch mode.

### 10.2.3 Transmit Shift Register (TSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Read/Write | — | — | — | — | — | — | — | — |

TSR is a register used to transmit serial data. Transmit data is first transferred from TDR to TSR, and serial data transmission is carried out by sending the data to the $TXD_{3x}$ pin in order, starting from the LSB (bit 0). When one byte of data is transmitted, the next byte of transmit data is transferred to TDR, and transmission started, automatically. Data transfer from TDR to TSR is not performed if no data has been written to TDR (if bit TDRE is set to 1 in the serial status register (SSR)).

TSR cannot be read or written directly by the CPU.

### 10.2.4 Transmit Data Register (TDR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDR7 | TDR6 | TDR5 | TDR4 | TDR3 | TDR2 | TDR1 | TDR0 |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

TDR is an 8-bit register that stores transmit data. When TSR is found to be empty, the transmit data written in TDR is transferred to TSR, and serial data transmission is started. Continuous transmission is possible by writing the next transmit data to TDR during TSR serial data transmission.

TDR can be read or written by the CPU at any time.

TDR is initialized to H'FF upon reset, and in standby, module standby, or watch mode.

### 10.2.5 Serial Mode Register (SMR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | COM | CHR | PE | PM | STOP | MP | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

SMR is an 8-bit register used to set the serial data transfer format and to select the clock source for the baud rate generator.

SMR can be read or written by the CPU at any time.

SMR is initialized to H'00 upon reset, and in standby, module standby, or watch mode.

**Bit 7:** Communication mode (COM)

Bit 7 selects whether SCI3 operates in asynchronous mode or synchronous mode.

| Bit 7<br>COM | Description | |
|---|---|---|
| 0 | Asynchronous mode | (initial value) |
| 1 | Synchronous mode | |

**Bit 6:** Character length (CHR)

Bit 6 selects either 7 or 8 bits as the data length to be used in asynchronous mode. In synchronous mode the data length is always 8 bits, irrespective of the bit 6 setting.

| Bit 6<br>CHR | Description | |
|---|---|---|
| 0 | 8-bit data/5-bit data*2 | (initial value) |
| 1 | 7-bit data*1/5-bit data*2 | |

Notes: 1. When 7-bit data is selected, the MSB (bit 7) of TDR is not transmitted.
2. When 5-bit data is selected, set both PE and MP to 1. The three most significant bits (bits 7, 6, and 5) of TDR are not transmitted.

**Bit 5:** Parity enable (PE)

Bit 5 selects whether a parity bit is to be added during transmission and checked during reception in asynchronous mode. In synchronous mode parity bit addition and checking is not performed, irrespective of the bit 5 setting.

| Bit 5<br>PE | Description | |
|---|---|---|
| 0 | Parity bit addition and checking disabled*2 | (initial value) |
| 1 | Parity bit addition and checking enabled*1*2 | |

Notes: 1. When PE is set to 1, even or odd parity, as designated by bit PM, is added to transmit data before it is sent, and the received parity bit is checked against the parity designated by bit PM.
2. For the case where 5-bit data is selected, see table 10.11.

**Bit 4:** Parity mode (PM)

Bit 4 selects whether even or odd parity is to be used for parity addition and checking. The PM bit setting is only valid in asynchronous mode when bit PE is set to 1, enabling parity bit addition and checking. The PM bit setting is invalid in synchronous mode, and in asynchronous mode if parity bit addition and checking is disabled.

| Bit 4<br>PM | Description | |
|---|---|---|
| 0 | Even parity*1 | (initial value) |
| 1 | Odd parity*2 | |

Notes: 1. When even parity is selected, a parity bit is added in transmission so that the total number of 1 bits in the transmit data plus the parity bit is an even number; in reception, a check is carried out to confirm that the number of 1 bits in the receive data plus the parity bit is an even number.
2. When odd parity is selected, a parity bit is added in transmission so that the total number of 1 bits in the transmit data plus the parity bit is an odd number; in reception, a check is carried out to confirm that the number of 1 bits in the receive data plus the parity bit is an odd number.

**Bit 3:** Stop bit length (STOP)

Bit 3 selects 1 bit or 2 bits as the stop bit length is asynchronous mode. The STOP bit setting is only valid in asynchronous mode. When synchronous mode is selected the STOP bit setting is invalid since stop bits are not added.

| Bit 3<br>STOP | Description | |
|---|---|---|
| 0 | 1 stop bit*1 | (initial value) |
| 1 | 2 stop bits*2 | |

Notes: 1. In transmission, a single 1 bit (stop bit) is added at the end of a transmit character.
2. In transmission, two 1 bits (stop bits) are added at the end of a transmit character.

In reception, only the first of the received stop bits is checked, irrespective of the STOP bit setting. If the second stop bit is 1 it is treated as a stop bit, but if 0, it is treated as the start bit of the next transmit character.

**Bit 2:** 5-Bit Communication (MP)

Bit 2 can set the 5-bit communication format. When this bit is set to 1, the 5-bit communication format is enabled. When writing 1 to this bit, always write 1 to bit 5 (RE) at the same time.

| Bit 2<br>MP | Description | |
|---|---|---|
| 0 | 7/8-bit communication format* | (initial value) |
| 1 | 5-bit communication format* | |

Note: * For the case where 5-bit data is selected, see table 10.11.

**Bits 1 and 0:** Clock select 1, 0 (CKS1, CKS0)

Bits 1 and 0 choose ϕ/64, ϕ/16, ϕw/2, or ϕ as the clock source for the baud rate generator.

For the relation between the clock source, bit rate register setting, and baud rate, see section 10.2.8, Bit rate register (BRR).

| Bit 1 CKS1 | Bit 0 CKS0 | Description |
|---|---|---|
| 0 | 0 | ϕ clock (initial value) |
| 0 | 1 | ϕ w/2 clock*1/ϕ w clock*2 |
| 1 | 0 | ϕ/16 clock |
| 1 | 1 | ϕ/64 clock |

Notes: 1. ϕ w/2 clock in active (medium-speed/high-speed) mode and sleep mode
2. ϕ w clock in subactive mode and subsleep mode
In subactive or subsleep mode, SCI3 can be operated when CPU clock is ϕw/2 only.

### 10.2.6 Serial Control Register 3 (SCR3)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

SCR3 is an 8-bit register for selecting transmit or receive operation, the asynchronous mode clock output, interrupt request enabling or disabling, and the transmit/receive clock source.

SCR3 can be read or written by the CPU at any time.

SCR3 is initialized to H'00 upon reset, and in standby, module standby or watch mode.

**Bit 7:** Transmit interrupt enable (TIE)

Bit 7 selects enabling or disabling of the transmit data empty interrupt request (TXI) when transmit data is transferred from the transmit data register (TDR) to the transmit shift register (TSR), and bit TDRE in the serial status register (SSR) is set to 1.

TXI can be released by clearing bit TDRE or bit TIE to 0.

| Bit 7<br>TIE | Description | |
|---|---|---|
| 0 | Transmit data empty interrupt request (TXI) disabled | (initial value) |
| 1 | Transmit data empty interrupt request (TXI) enabled | |

**Bit 6:** Receive interrupt enable (RIE)

Bit 6 selects enabling or disabling of the receive data full interrupt request (RXI) and the receive error interrupt request (ERI) when receive data is transferred from the receive shift register (RSR) to the receive data register (RDR), and bit RDRF in the serial status register (SSR) is set to 1. There are three kinds of receive error: overrun, framing, and parity.

RXI can be released by clearing bit RDRF or the FER, PER, or OER error flag to 0, or by clearing bit RIE to 0.

| Bit 6<br>RIE | Description | |
|---|---|---|
| 0 | Receive data full interrupt request (RXI) and receive error interrupt request (ERI) disabled | (initial value) |
| 1 | Receive data full interrupt request (RXI) and receive error interrupt request (ERI) enabled | |

**Bit 5:** Transmit enable (TE)

Bit 5 selects enabling or disabling of the start of transmit operation.

| Bit 5<br>TE | Description | |
|---|---|---|
| 0 | Transmit operation disabled*1 (TXD pin is I/O port) | (initial value) |
| 1 | Transmit operation enabled*2 (TXD pin is transmit data pin) | |

Notes: 1. Bit TDRE in SSR is fixed at 1.
2. When transmit data is written to TDR in this state, bit TDR in SSR is cleared to 0 and serial data transmission is started. Be sure to carry out serial mode register (SMR) settings, and setting of bit SPC31 or SPC32 in SPCR, to decide the transmission format before setting bit TE to 1.

**Bit 4:** Receive enable (RE)

Bit 4 selects enabling or disabling of the start of receive operation.

| Bit 4<br>RE | Description | |
|---|---|---|
| 0 | Receive operation disabled*1 (RXD pin is I/O port) | (initial value) |
| 1 | Receive operation enabled*2 (RXD pin is receive data pin) | |

Notes: 1. Note that the RDRF, FER, PER, and OER flags in SSR are not affected when bit RE is cleared to 0, and retain their previous state.
2. In this state, serial data reception is started when a start bit is detected in asynchronous mode or serial clock input is detected in synchronous mode. Be sure to carry out serial mode register (SMR) settings to decide the reception format before setting bit RE to 1.

**Bit 3:** Reserved (MPIE)

Bit 3 is reserved.

**Bit 2:** Transmit end interrupt enable (TEIE)

Bit 2 selects enabling or disabling of the transmit end interrupt request (TEI) if there is no valid transmit data in TDR when MSB data is to be sent.

| Bit 2 TEIE | Description | |
|---|---|---|
| 0 | Transmit end interrupt request (TEI) disabled | (initial value) |
| 1 | Transmit end interrupt request (TEI) enabled* | |

Note: * TEI can be released by clearing bit TDRE to 0 and clearing bit TEND to 0 in SSR, or by clearing bit TEIE to 0.

**Bits 1 and 0:** Clock enable 1 and 0 (CKE1, CKE0)

Bits 1 and 0 select the clock source and enabling or disabling of clock output from the $SCK_{3x}$ pin. The combination of CKE1 and CKE0 determines whether the $SCK_{3x}$ pin functions as an I/O port, a clock output pin, or a clock input pin.

The CKE0 bit setting is only valid in case of internal clock operation (CKE1 = 0) in asynchronous mode. In synchronous mode, or when external clock operation is used (CKE1 = 1), bit CKE0 should be cleared to 0.

After setting bits CKE1 and CKE0, set the operating mode in the serial mode register (SMR).

For details on clock source selection, see table 10.9 in 10.3.1, Overview.

| Bit 1 CKE1 | Bit 0 CKE0 | Description | | |
|---|---|---|---|---|
| | | **Communication Mode** | **Clock Source** | **$SCK_{3x}$ Pin Function** |
| 0 | 0 | Asynchronous | Internal clock | I/O port*1 |
| | | Synchronous | Internal clock | Serial clock output*1 |
| 0 | 1 | Asynchronous | Internal clock | Clock output*2 |
| | | Synchronous | Reserved | |
| 1 | 0 | Asynchronous | External clock | Clock input*3 |
| | | Synchronous | External clock | Serial clock input |
| 1 | 1 | Asynchronous | Reserved | |
| | | Synchronous | Reserved | |

Notes: 1. Initial value
2. A clock with the same frequency as the bit rate is output.
3. Input a clock with a frequency 16 times the bit rate.

### 10.2.7 Serial Status Register (SSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | OER | FER | PER | TEND | MPBR | MPBT |
| Initial value | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R | R | R/W |

Note: * Only a write of 0 for flag clearing is possible.

SSR is an 8-bit register containing status flags that indicate the operational status of SCI3.

SSR can be read or written by the CPU at any time, but only a write of 1 is possible to bits TDRE, RDRF, OER, PER, and FER. In order to clear these bits by writing 0, 1 must first be read.

Bits TEND and MPBR are read-only bits, and cannot be modified.

SSR is initialized to H'84 upon reset, and in standby, module standby, or watch mode.

**Bit 7:** Transmit data register empty (TDRE)

Bit 7 indicates that transmit data has been transferred from TDR to TSR.

| Bit 7<br>TDRE | Description |
|---|---|
| 0 | Transmit data written in TDR has not been transferred to TSR<br>Clearing conditions:<br>After reading TDRE = 1, cleared by writing 0 to TDRE<br>When data is written to TDR by an instruction |
| 1 | Transmit data has not been written to TDR, or transmit data written in TDR has been transferred to TSR<br>Setting conditions:<br>When bit TE in SCR3 is cleared to 0<br>When data is transferred from TDR to TSR (initial value) |

**Bit 6:** Receive data register full (RDRF)

Bit 6 indicates that received data is stored in RDR.

| Bit 6<br>RDRF | Description | |
|---|---|---|
| 0 | There is no receive data in RDR | (initial value) |
| | Clearing conditions: | |
| | After reading RDRF = 1, cleared by writing 0 to RDRF | |
| | When RDR data is read by an instruction | |
| 1 | There is receive data in RDR | |
| | Setting condition: | |
| | When reception ends normally and receive data is transferred from RSR to RDR | |

Note: If an error is detected in the receive data, or if the RE bit in SCR3 has been cleared to 0, RDR and bit RDRF are not affected and retain their previous state.
Note that if data reception is completed while bit RDRF is still set to 1, an overrun error (OER) will result and the receive data will be lost.

**Bit 5:** Overrun error (OER)

Bit 5 indicates that an overrun error has occurred during reception.

| Bit 5<br>OER | Description | |
|---|---|---|
| 0 | Reception in progress or completed*1 | (initial value) |
| | Clearing condition: | |
| | After reading OER = 1, cleared by writing 0 to OER | |
| 1 | An overrun error has occurred during reception*2 | |
| | Setting condition: | |
| | When reception is completed with RDRF set to 1 | |

Notes:
1. When bit RE in SCR3 is cleared to 0, bit OER is not affected and retains its previous state.
2. RDR retains the receive data it held before the overrun error occurred, and data received after the error is lost. Reception cannot be continued with bit OER set to 1, and in synchronous mode, transmission cannot be continued either.

**Bit 4:** Framing error (FER)

Bit 4 indicates that a framing error has occurred during reception in asynchronous mode.

| Bit 4<br>FER | Description |
|---|---|
| 0 | Reception in progress or completed*1 (initial value)<br>Clearing condition:<br>After reading FER = 1, cleared by writing 0 to FER |
| 1 | A framing error has occurred during reception<br>Setting condition:<br>When the stop bit at the end of the receive data is checked for a value of 1 at the end of reception, and the stop bit is 0*2 |

Notes: 1. When bit RE in SCR3 is cleared to 0, bit FER is not affected and retains its previous state.

2. Note that, in 2-stop-bit mode, only the first stop bit is checked for a value of 1, and the second stop bit is not checked. When a framing error occurs the receive data is transferred to RDR but bit RDRF is not set. Reception cannot be continued with bit FER set to 1. In synchronous mode, neither transmission nor reception is possible when bit FER is set to 1.

**Bit 3:** Parity error (PER)

Bit 3 indicates that a parity error has occurred during reception with parity added in asynchronous mode.

| Bit 3<br>PER | Description |
|---|---|
| 0 | Reception in progress or completed*1 (initial value)<br>Clearing condition:<br>After reading PER = 1, cleared by writing 0 to PER |
| 1 | A parity error has occurred during reception*2<br>Setting condition:<br>When the number of 1 bits in the receive data plus parity bit does not match the parity designated by bit PM in the serial mode register (SMR) |

Notes: 1. When bit RE in SCR3 is cleared to 0, bit PER is not affected and retains its previous state.
2. Receive data in which it a parity error has occurred is still transferred to RDR, but bit RDRF is not set. Reception cannot be continued with bit PER set to 1. In synchronous mode, neither transmission nor reception is possible when bit FER is set to 1.

**Bit 2:** Transmit end (TEND)

Bit 2 indicates that bit TDRE is set to 1 when the last bit of a transmit character is sent.

Bit 2 is a read-only bit and cannot be modified.

| Bit 2<br>TEND | Description |
|---|---|
| 0 | Transmission in progress<br>Clearing conditions:<br>After reading TDRE = 1, cleared by writing 0 to TDRE<br>When data is written to TDR by an instruction |
| 1 | Transmission ended (initial value)<br>Setting conditions:<br>When bit TE in SCR3 is cleared to 0<br>When bit TDRE is set to 1 when the last bit of a transmit character is sent |

**Bit 1:** Reserved (MPBR)

Bit 1 is reserved.

It is a read-only bit and cannot be modified.

**Bit 0:** Reserved (MPBT)

Bit 0 is reserved.

The write value should always be 0.

### 10.2.8 Bit Rate Register (BRR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BRR7 | BRR6 | BRR5 | BRR4 | BRR3 | BRR2 | BRR1 | BRR0 |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

BRR is an 8-bit register that designates the transmit/receive bit rate in accordance with the baud rate generator operating clock selected by bits CKS1 and CKS0 of the serial mode register (SMR).

BRR can be read or written by the CPU at any time.

BRR is initialized to H'FF upon reset, and in standby, module standby, or watch mode.

Table 10.3 shows examples of BRR settings in asynchronous mode. The values shown are for active (high-speed) mode.

**Table 10.3 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (1)**

| | OSC | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 32.8 kHz | | | 38.4 kHz | | | 2 MHz | | | 2.4576 MHz | | | 4 MHz | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | Cannot be used, as error exceeds 3% | | | — | — | — | — | — | — | 2 | 21 | –0.83 | — | — | — |
| 150 | | | | 0 | 3 | 0 | 2 | 12 | 0.16 | 3 | 3 | 0 | 2 | 25 | 0.16 |
| 200 | | | | 0 | 2 | 0 | 0 | 155 | 0.16 | 3 | 2 | 0 | — | — | — |
| 250 | | | | — | — | — | 0 | 124 | 0 | 0 | 153 | –0.26 | 0 | 249 | 0 |
| 300 | | | | 0 | 1 | 0 | 0 | 103 | 0.16 | 3 | 1 | 0 | 2 | 12 | 0.16 |
| 600 | | | | 0 | 0 | 0 | 0 | 51 | 0.16 | 3 | 0 | 0 | 0 | 103 | 0.16 |
| 1200 | | | | — | — | — | 0 | 25 | 0.16 | 2 | 1 | 0 | 0 | 51 | 0.16 |
| 2400 | | | | — | — | — | 0 | 12 | 0.16 | 2 | 0 | 0 | 0 | 25 | 0.16 |
| 4800 | | | | — | — | — | — | — | — | 0 | 7 | 0 | 0 | 12 | 0.16 |
| 9600 | | | | — | — | — | — | — | — | 0 | 3 | 0 | — | — | — |
| 19200 | | | | — | — | — | — | — | — | 0 | 1 | 0 | — | — | — |
| 31250 | | | | — | — | — | 0 | 0 | 0 | — | — | — | 0 | 1 | 0 |
| 38400 | | | | — | — | — | — | — | — | 0 | 0 | 0 | — | — | — |

**Table 10.3 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (2)**

| | OSC | | | | | |
|---|---|---|---|---|---|---|
| | 10 MHz | | | 16 MHz | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 88 | -0.25 | 2 | 141 | 0.03 |
| 150 | 2 | 64 | 0.16 | 2 | 103 | 0.16 |
| 200 | 2 | 48 | -0.35 | 2 | 77 | 0.16 |
| 250 | 2 | 38 | 0.16 | 2 | 62 | -0.79 |
| 300 | — | — | — | 2 | 51 | 0.16 |
| 600 | — | — | — | 2 | 25 | 0.16 |
| 1200 | 0 | 129 | 0.16 | 0 | 207 | 0.16 |
| 2400 | 0 | 64 | 0.16 | 0 | 103 | 0.16 |
| 4800 | — | — | — | 0 | 51 | 0.16 |
| 9600 | — | — | — | 0 | 25 | 0.16 |
| 19200 | — | — | — | 0 | 12 | 0.16 |
| 31250 | 0 | 4 | 0 | 0 | 7 | 0 |
| 38400 | — | — | — | — | — | — |

Notes: 1. The setting should be made so that the error is not more than 1%.

2. The value set in BRR is given by the following equation:

$$N = \frac{OSC}{(64 \times 2^{2n} \times B)} - 1$$

where B: Bit rate (bit/s)

N: Baud rate generator BRR setting ($0 \leq N \leq 255$)

OSC: Value of $\phi_{OSC}$ (Hz)

n: Baud rate generator input clock number (n = 0, 2, or 3)

(The relation between n and the clock is shown in table 10.4.)

3. The error in table 10.3 is the value obtained from the following equation, rounded to two decimal places.

$$\text{Error (\%)} = \frac{\text{B (rate obtained from n, N, OSC)} - \text{R(bit rate in left-hand column in table 10.3.)}}{\text{R (bit rate in left-hand column in table 10.3.)}} \times 100$$

**Table 10.4 Relation between n and Clock**

| | | SMR Setting | |
|---|---|---|---|
| **n** | **Clock** | **CKS1** | **CKS0** |
| 0 | $\phi$ | 0 | 0 |
| 0 | $\phi_w/2$ *1 / $\phi_w$ *2 | 0 | 1 |
| 2 | $\phi/16$ | 1 | 0 |
| 3 | $\phi/64$ | 1 | 1 |

Notes: 1. $\phi$ w/2 clock in active (medium-speed/high-speed) mode and sleep mode
2. $\phi$ w clock in subactive mode and subsleep mode
In subactive or subsleep mode, SCI3 can be operated when CPU clock is $\phi$w/2 only.

Table 10.5 shows the maximum bit rate for each frequency. The values shown are for active (high-speed) mode.

**Table 10.5 Maximum Bit Rate for Each Frequency (Asynchronous Mode)**

| | | Setting | |
|---|---|---|---|
| **OSC (MHz)** | **Maximum Bit Rate (bit/s)** | **n** | **N** |
| 0.0384* | 600 | 0 | 0 |
| 2 | 31250 | 0 | 0 |
| 2.4576 | 38400 | 0 | 0 |
| 4 | 62500 | 0 | 0 |
| 10 | 156250 | 0 | 0 |
| 16 | 250000 | 0 | 0 |

*: When SMR is set up to CKS1 = “0”, CKS0 = “1”.

Table 10.6 shows examples of BRR settings in synchronous mode. The values shown are for active (high-speed) mode.

**Table 10.6 Examples of BRR Settings for Various Bit Rates (Synchronous Mode)**

| Bit Rate (bit/s) | OSC | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 38.4 kHz | | | 2 MHz | | | 4 MHz | | | 10 MHz | | | 16 MHz | | |
| | n | N | Error | n | N | Error | n | N | Error | n | N | Error | n | N | Error |
| 200 | 0 | 23 | 0 | — | — | — | — | — | — | — | — | — | — | — | — |
| 250 | — | — | — | — | — | — | 2 | 124 | 0 | — | — | — | 3 | 124 | 0 |
| 300 | 2 | 0 | 0 | — | — | — | — | — | — | — | — | — | — | — | — |
| 500 | | | | — | — | — | — | — | — | — | — | — | 2 | 249 | 0 |
| 1K | | | | 0 | 249 | 0 | — | — | — | — | — | — | 2 | 124 | 0 |
| 2.5K | | | | 0 | 99 | 0 | 0 | 199 | 0 | — | — | — | 2 | 49 | 0 |
| 5K | | | | 0 | 49 | 0 | 0 | 99 | 0 | 0 | 249 | 0 | 2 | 24 | 0 |
| 10K | | | | 0 | 24 | 0 | 0 | 49 | 0 | 0 | 124 | 0 | 0 | 199 | 0 |
| 25K | | | | 0 | 9 | 0 | 0 | 19 | 0 | 0 | 49 | 0 | 0 | 79 | 0 |
| 50K | | | | 0 | 4 | 0 | 0 | 9 | 0 | 0 | 24 | 0 | 0 | 39 | 0 |
| 100K | | | | — | — | — | 0 | 4 | 0 | — | — | — | 0 | 19 | 0 |
| 250K | | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 0 | 0 | 7 | 0 |
| 500K | | | | | | | 0 | 0 | 0 | — | — | — | 0 | 3 | 0 |
| 1M | | | | | | | | | | — | — | — | 0 | 1 | 0 |

Blank: Cannot be set.

—: A setting can be made, but an error will result.

Notes: The value set in BRR is given by the following equation:

$$N = \frac{OSC}{(8 \times 2^{2n} \times B)} - 1$$

where B: Bit rate (bit/s)

N: Baud rate generator BRR setting ($0 \leq N \leq 255$)

OSC: Value of $\phi_{OSC}$ (Hz)

n: Baud rate generator input clock number (n = 0, 2, or 3)

(The relation between n and the clock is shown in table 10.7.)

**Table 10.7 Relation between n and Clock**

| | | SMR Setting | |
|---|---|---|---|
| **n** | **Clock** | **CKS1** | **CKS0** |
| 0 | $\phi$ | 0 | 0 |
| 0 | $\phi_w/2^{*1}/\phi_w^{*2}$ | 0 | 1 |
| 2 | $\phi/16$ | 1 | 0 |
| 3 | $\phi/64$ | 1 | 1 |

Notes: 1. $\phi$w/2 clock in active (medium-speed/high-speed) mode and sleep mode
2. $\phi$w clock in subactive mode and subsleep mode
In subactive or subsleep mode, SCI3 can be operated when CPU clock is $\phi$w/2 only.

### 10.2.9 Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bits relating to SCI3 are described here. For details of the other bits, see the sections on the relevant modules.

**Bit 6:** SCI31 module standby mode control (S31CKSTP)

Bit 6 controls setting and clearing of module standby mode for SCI31.

| **S31CKSTP** | **Description** | |
|---|---|---|
| 0 | SCI31 is set to module standby mode | |
| 1 | SCI31 module standby mode is cleared | (initial value) |

Note: All SCI31 register is initialized in module standby mode.

**Bit 5:** SCI32 module standby mode control (S32CKSTP)

Bit 5 controls setting and clearing of module standby mode for SCI32.

| S32CKSTP | Description | |
|---|---|---|
| 0 | SCI32 is set to module standby mode | |
| 1 | SCI32 module standby mode is cleared | (initial value) |

Note: All SCI32 register is initialized in module standby mode.

### 10.2.10 Serial Port Control Register (SPCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | SPC32 | SPC31 | SCINV3 | SCINV2 | SCINV1 | SCINV0 |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | R/W | R/W | R/W | R/W | R/W | R/W |

SPCR is an 8-bit readable/writable register that performs $RXD_{31}$, $RXD_{32}$, $TXD_{31}$, and $TXD_{32}$ pin input/output data inversion switching. SPCR is initialized to H'C0 by a reset.

**Bits 7 and 6:** Reserved bits

Bits 7 and 6 are reserved; they are always read as 1 and cannot be modified.

**Bit 5:** $P4_2$/$TXD_{32}$ pin function switch (SPC32)

This bit selects whether pin $P4_2$/$TXD_{32}$ is used as $P4_2$ or as $TXD_{32}$.

| Bit 5<br>SPC32 | Description | |
|---|---|---|
| 0 | Functions as $P4_2$ I/O pin | (initial value) |
| 1 | Functions as $TXD_{32}$ output pin* | |

Note: * Set the TE bit in SCR3 after setting this bit to 1.

**Bit 4:** $P3_5/TXD_{31}$ pin function switch (SPC31)

This bit selects whether pin $P3_5/TXD_{31}$ is used as $P3_5$ or as $TXD_{31}$.

| Bit 4<br>SPC31 | Description | |
|---|---|---|
| 0 | Functions as $P3_5$ I/O pin | (initial value) |
| 1 | Functions as $TXD_{31}$ output pin* | |

Note: * Set the TE bit in SCR3 after setting this bit to 1.

**Bit 3:** $TXD_{32}$ pin output data inversion switch

Bit 3 specifies whether or not $TXD_{32}$ pin output data is to be inverted.

| Bit 3<br>SCINV3 | Description | |
|---|---|---|
| 0 | $TXD_{32}$ output data is not inverted | (initial value) |
| 1 | $TXD_{32}$ output data is inverted | |

**Bit 2:** $RXD_{32}$ pin input data inversion switch

Bit 2 specifies whether or not $RXD_{32}$ pin input data is to be inverted.

| Bit 2<br>SCINV2 | Description | |
|---|---|---|
| 0 | $RXD_{32}$ input data is not inverted | (initial value) |
| 1 | $RXD_{32}$ input data is inverted | |

**Bit 1:** $TXD_{31}$ pin output data inversion switch

Bit 1 specifies whether or not $TXD_{31}$ pin output data is to be inverted.

| Bit 1<br>SCINV1 | Description | |
|---|---|---|
| 0 | $TXD_{31}$ output data is not inverted | (initial value) |
| 1 | $TXD_{31}$ output data is inverted | |

**Bit 0:** $RXD_{31}$ pin input data inversion switch

Bit 0 specifies whether or not $RXD_{31}$ pin input data is to be inverted.

| Bit 0<br>SCINV0 | Description | |
|---|---|---|
| 0 | $RXD_{31}$ input data is not inverted | (initial value) |
| 1 | $RXD_{31}$ input data is inverted | |

## 10.3 Operation

### 10.3.1 Overview

SCI3 can perform serial communication in two modes: asynchronous mode in which synchronization is provided character by character, and synchronous mode in which synchronization is provided by clock pulses. The serial mode register (SMR) is used to select asynchronous or synchronous mode and the data transfer format, as shown in table 10.8.

The clock source for SCI3 is determined by bit COM in SMR and bits CKE1 and CKE0 in SCR3, as shown in table 10.9.

**(1) Synchronous Mode**

- Choice of 5-, 7-, or 8-bit data length
- Choice of parity addition, and addition of 1 or 2 stop bits. (The combination of these parameters determines the data transfer format and the character length.)
- Framing error (FER), parity error (PER), overrun error (OER), and break detection during reception
- Choice of internal or external clock as the clock source

  When internal clock is selected: SCI3 operates on the baud rate generator clock, and a clock with the same frequency as the bit rate can be output.

  When external clock is selected: A clock with a frequency 16 times the bit rate must be input. (The on-chip baud rate generator is not used.)

**(2) Synchronous Mode**

- Data transfer format: Fixed 8-bit data length
- Overrun error (OER) detection during reception
- Choice of internal or external clock as the clock source

  When internal clock is selected: SCI3 operates on the baud rate generator clock, and a serial clock is output.

  When external clock is selected: The on-chip baud rate generator is not used, and SCI3 operates on the input serial clock.

**Table 10.8 SMR Settings and Corresponding Data Transfer Formats**

| SMR | | | | | | Data Transfer Format | | |
|---|---|---|---|---|---|---|---|---|
| Bit 7 COM | Bit 6 CHR | Bit 2 MP | Bit 5 PE | Bit 3 STOP | Mode | Data Length | Parity Bit | Stop Bit Length |
| 0 | 0 | 0 | 0 | 0 | Asynchronous mode | 8-bit data | No | 1 bit |
| | | | | 1 | | | | 2 bits |
| | | | 1 | 0 | | | Yes | 1 bit |
| | | | | 1 | | | | 2 bits |
| | 1 | | 0 | 0 | | 7-bit data | No | 1 bit |
| | | | | 1 | | | | 2 bits |
| | | | 1 | 0 | | | Yes | 1 bit |
| | | | | 1 | | | | 2 bits |
| | 0 | 1 | 0 | 0 | Setting prohibited | | | |
| | | | | 1 | | | | |
| | | | 1 | 0 | Asynchronous mode | 5-bit data | No | 1 bit |
| | | | | 1 | | | | 2 bits |
| | 1 | | 0 | 0 | Setting prohibited | | | |
| | | | | 1 | | | | |
| | | | 1 | 0 | Asynchronous mode | 5-bit data | Yes | 1 bit |
| | | | | 1 | | | | 2 bits |
| 1 | * | 0 | * | * | Synchronous mode | 8-bit data | No | No |

*: Don’t care

**Table 10.9 SMR and SCR3 Settings and Clock Source Selection**

| SMR | SCR3 | | | Transmit/Receive Clock | |
|---|---|---|---|---|---|
| Bit 7 | Bit 1 | Bit 0 | | | |
| COM | CKE1 | CKE0 | Mode | Clock Source | $SCK_{3x}$ Pin Function |
| 0 | 0 | 0 | Asynchronous mode | Internal | I/O port ($SCK_{3x}$ pin not used) |
| | | 1 | | | Outputs clock with same frequency as bit rate |
| | 1 | 0 | | External | Outputs clock with frequency 16 times bit rate |
| 1 | 0 | 0 | Synchronous mode | Internal | Outputs serial clock |
| | 1 | 0 | | External | Inputs serial clock |
| 0 | 1 | 1 | Reserved (Do not specify these combinations) | | |
| 1 | 0 | 1 | | | |
| 1 | 1 | 1 | | | |

### (3) Interrupts and Continuous Transmission/Reception

SCI3 can carry out continuous reception using RXI and continuous transmission using TXI. These interrupts are shown in table 10.10.

**Table 10.10 Transmit/Receive Interrupts**

| Interrupt | Flags | Interrupt Request Conditions | Notes |
|---|---|---|---|
| RXI | RDRF<br>RIE | When serial reception is performed normally and receive data is transferred from RSR to RDR, bit RDRF is set to 1, and if bit RIE is set to 1 at this time, RXI is enabled and an interrupt is requested. (See figure 10.2(a).) | The RXI interrupt routine reads the receive data transferred to RDR and clears bit RDRF to 0. Continuous reception can be performed by repeating the above operations until reception of the next RSR data is completed. |
| TXI | TDRE<br>TIE | When TSR is found to be empty (on completion of the previous transmission) and the transmit data placed in TDR is transferred to TSR, bit TDRE is set to 1. If bit TIE is set to 1 at this time, TXI is enabled and an interrupt is requested. (See figure 10.2(b).) | The TXI interrupt routine writes the next transmit data to TDR and clears bit TDRE to 0. Continuous transmission can be performed by repeating the above operations until the data transferred to TSR has been transmitted. |
| TEI | TEND<br>TEIE | When the last bit of the character in TSR is transmitted, if bit TDRE is set to 1, bit TEND is set to 1. If bit TEIE is set to 1 at this time, TEI is enabled and an interrupt is requested. (See figure 10.2(c).) | TEI indicates that the next transmit data has not been written to TDR when the last bit of the transmit character in TSR is sent. |

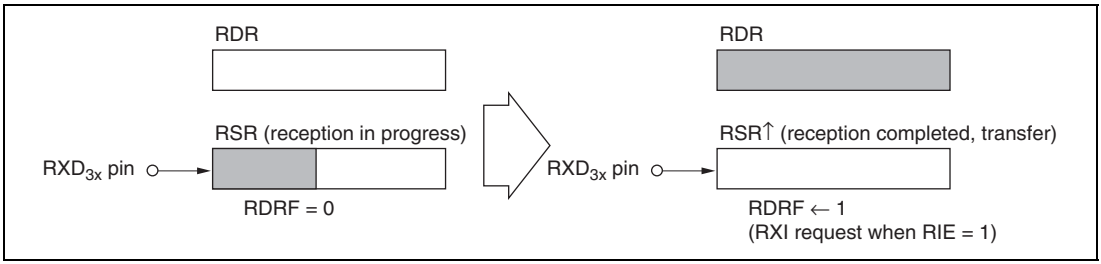



**Figure 10.2 (a) RDRF Setting and RXI Interrupt**

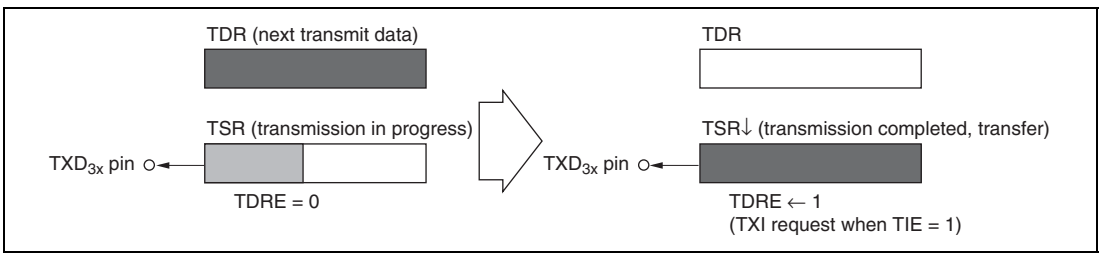



**Figure 10.2 (b) TDRE Setting and TXI Interrupt**

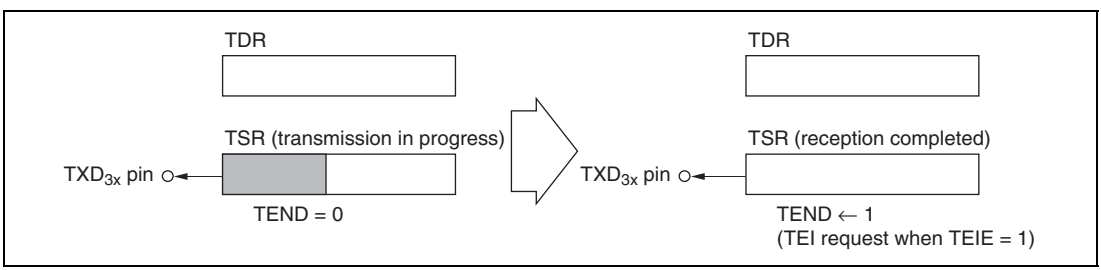



**Figure 10.2 (c) TEND Setting and TEI Interrupt**

### 10.3.2 Operation in Asynchronous Mode

In asynchronous mode, serial communication is performed with synchronization provided character by character. A start bit indicating the start of communication and one or two stop bits indicating the end of communication are added to each character before it is sent.

SCI3 has separate transmission and reception units, allowing full-duplex communication. As the transmission and reception units are both double-buffered, data can be written during transmission and read during reception, making possible continuous transmission and reception.

#### (1) Data Transfer Format

The general data transfer format in asynchronous communication is shown in figure 10.3.

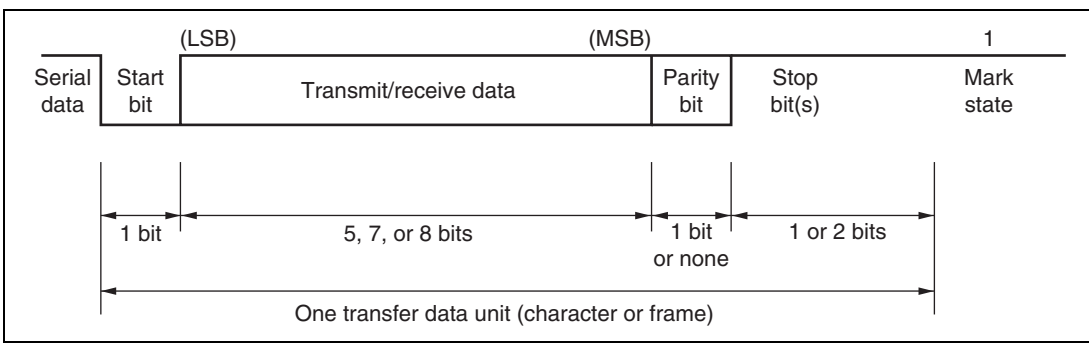



**Figure 10.3 Data Format in Asynchronous Communication**

In asynchronous communication, the communication line is normally in the mark state (high level). SCI3 monitors the communication line and when it detects a space (low level), identifies this as a start bit and begins serial data communication.

One transfer data character consists of a start bit (low level), followed by transmit/receive data (LSB-first format, starting from the least significant bit), a parity bit (high or low level), and finally one or two stop bits (high level).

In asynchronous mode, synchronization is performed by the falling edge of the start bit during reception. The data is sampled on the 8th pulse of a clock with a frequency 16 times the bit period, so that the transfer data is latched at the center of each bit.

Table 10.11 shows the 16 data transfer formats that can be set in asynchronous mode. The format is selected by the settings in the serial mode register (SMR).

## Table 10.11 Data Transfer Formats (Asynchronous Mode)

| SMR | | | | Serial Data Transfer Format and Frame Length | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHR | PE | MP | STOP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | 0 | S | 8-bit data | | | | | | | | STOP | | |
| 0 | 0 | 0 | 1 | S | 8-bit data | | | | | | | | STOP | STOP | |
| 0 | 0 | 1 | 0 | Setting prohibited | | | | | | | | | | | |
| 0 | 0 | 1 | 1 | Setting prohibited | | | | | | | | | | | |
| 0 | 1 | 0 | 0 | S | 8-bit data | | | | | | | | P | STOP | |
| 0 | 1 | 0 | 1 | S | 8-bit data | | | | | | | | P | STOP | STOP |
| 0 | 1 | 1 | 0 | S | 5-bit data | | | | | STOP | | | | | |
| 0 | 1 | 1 | 1 | S | 5-bit data | | | | | STOP | STOP | | | | |
| 1 | 0 | 0 | 0 | S | 7-bit data | | | | | | | STOP | | | |
| 1 | 0 | 0 | 1 | S | 7-bit data | | | | | | | STOP | STOP | | |
| 1 | 0 | 1 | 0 | Setting prohibited | | | | | | | | | | | |
| 1 | 0 | 1 | 1 | Setting prohibited | | | | | | | | | | | |
| 1 | 1 | 0 | 0 | S | 7-bit data | | | | | | | P | STOP | | |
| 1 | 1 | 0 | 1 | S | 7-bit data | | | | | | | P | STOP | STOP | |
| 1 | 1 | 1 | 0 | S | 5-bit data | | | | | P | STOP | | | | |
| 1 | 1 | 1 | 1 | S | 5-bit data | | | | | P | STOP | STOP | | | |

Legend:
S: Start bit
STOP: Stop bit
P: Parity bit

### (2) Clock

Either an internal clock generated by the baud rate generator or an external clock input at the $SCK_{3x}$ pin can be selected as the SCI3 transmit/receive clock. The selection is made by means of bit COM in SMR and bits SCE1 and CKE0 in SCR3. See table 10.9 for details on clock source selection.

When an external clock is input at the $SCK_{3x}$ pin, the clock frequency should be 16 times the bit rate.

When SCI3 operates on an internal clock, the clock can be output at the $SCK_{3x}$ pin. In this case the frequency of the output clock is the same as the bit rate, and the phase is such that the clock rises at the center of each bit of transmit/receive data, as shown in figure 10.4.

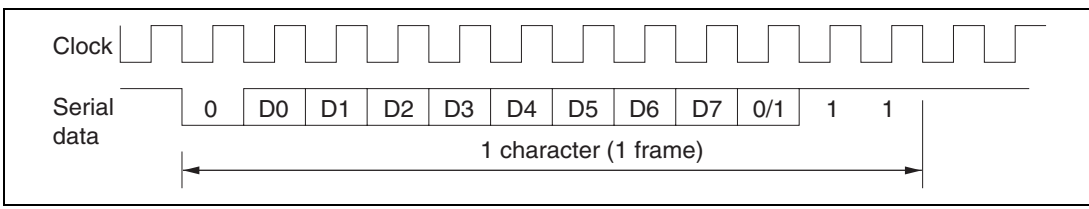


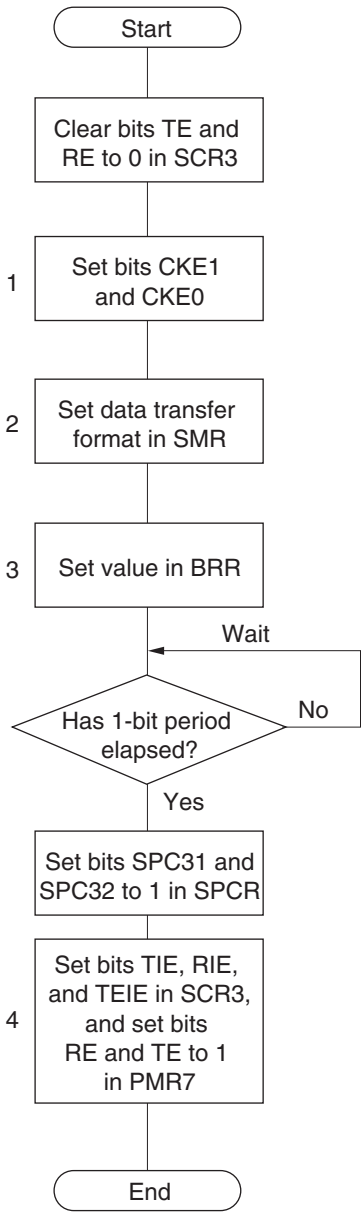

**Figure 10.4 Phase Relationship between Output Clock and Transfer Data (Asynchronous Mode) (8-bit data, parity, 2 stop bits)**

### (3) Data Transfer Operations

**SCI3 initialization:** Before data is transferred on SCI3, bits TE and RE in SCR3 must first be cleared to 0, and then SCI3 must be initialized as follows.

Note: If the operation mode or data transfer format is changed, bits TE and RE must first be cleared to 0.
When bit TE is cleared to 0, bit TDRE is set to 1.
Note that the RDRF, PER, FER, and OER flags and the contents of RDR are retained when RE is cleared to 0.
When an external clock is used in asynchronous mode, the clock should not be stopped during operation, including initialization. When an external clock is used in synchronous mode, the clock should not be supplied during operation, including initialization.

Figure 10.5 shows an example of a flowchart for initializing SCI3.




1. Set clock selection in SCR3. Be sure to clear the other bits to 0. If clock output is selected in asynchronous mode, the clock is output immediately after setting bits CKE1 and CKE0. If clock output is selected for reception in synchronous mode, the clock is output immediately after bits CKE1, CKE0, and RE are set to 1.

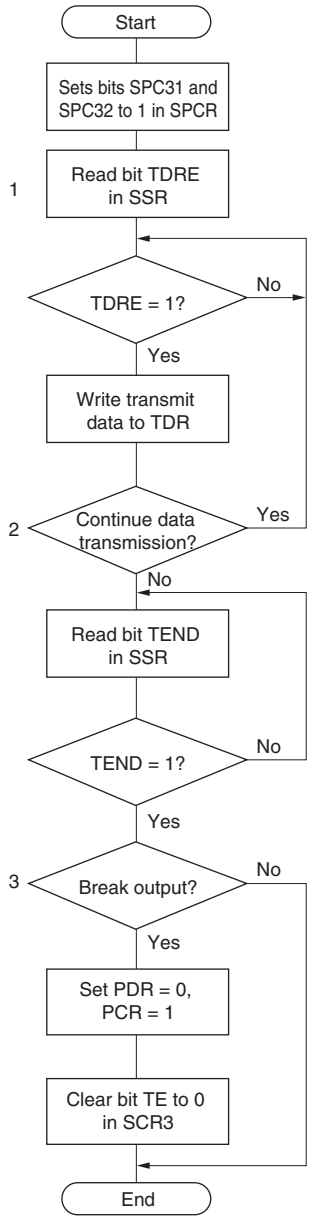
2. Set the data transfer format in the serial mode register (SMR).

3. Write the value corresponding to the transfer rate in BRR. This operation is not necessary when an external clock is selected.

4. Wait for at least one bit period, then set bits TIE, RIE, and TEIE in SCR3, and set bits RE and TE to 1 in PMR7. Setting bits TE and RE enables the TXD3x and RXD3x pins to be used. In asynchronous mode the mark state is established when transmitting, and the idle state waiting for a start bit when receiving.

**Figure 10.5 Example of SCI3 Initialization Flowchart**

**Transmitting:** Figure 10.6 shows an example of a flowchart for data transmission. This procedure should be followed for data transmission after initializing SCI3.




1. Read the serial status register (SSR) and check that bit TDRE is set to 1, then write transmit data to the transmit data register (TDR). When data is written to TDR, bit TDRE is cleared to 0 automatically.
   (After the TE bit is set to 1, one frame of 1s is output, then transmission is possible.)

2. When continuing data transmission, be sure to read TDRE = 1 to confirm that a write can be performed before writing data to TDR. When data is written to TDR, bit TDRE is cleared to 0 automatically.

3. If a break is to be output when data transmission ends, set the port PCR to 1 and clear the port PDR to 0, then clear bit TE in SCR3 to 0.

**Figure 10.6 Example of Data Transmission Flowchart (Asynchronous Mode)**

SCI3 operates as follows when transmitting data.

SCI3 monitors bit TDRE in SSR, and when it is cleared to 0, recognizes that data has been written to TDR and transfers data from TDR to TSR. It then sets bit TDRE to 1 and starts transmitting. If bit TIE in SCR3 is set to 1 at this time, a TXI request is made.

Serial data is transmitted from the TXD3x pin using the relevant data transfer format in table 10.11. When the stop bit is sent, SCI3 checks bit TDRE. If bit TDRE is cleared to 0, SCI3 transfers data from TDR to TSR, and when the stop bit has been sent, starts transmission of the next frame. If bit TDRE is set to 1, bit TEND in SSR bit is set to 1the mark state, in which 1s are transmitted, is established after the stop bit has been sent. If bit TEIE in SCR3 is set to 1 at this time, a TEI request is made.

Figure 10.12 shows an example of the operation when transmitting in asynchronous mode.

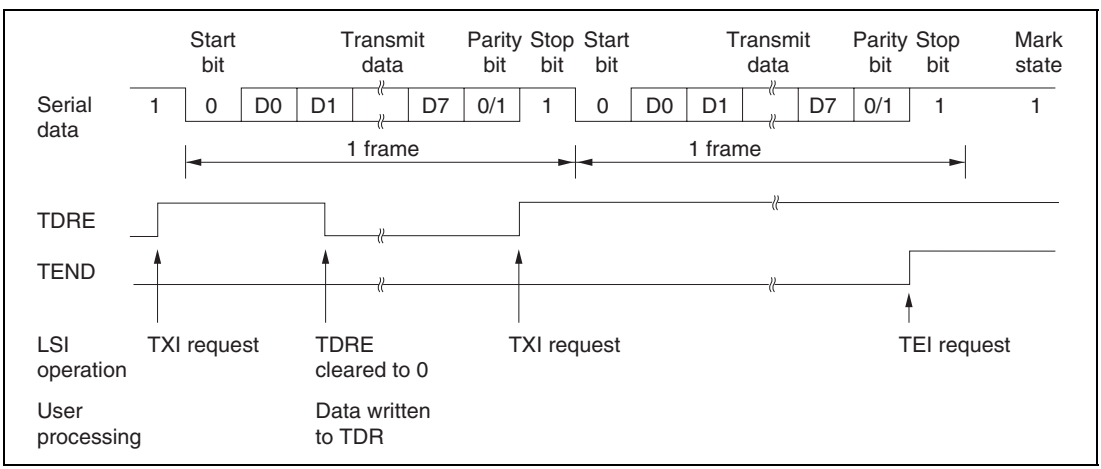



**Figure 10.7 Example of Operation when Transmitting in Asynchronous Mode (8-bit data, parity, 1 stop bit)**

**Receiving:** Figure 10.8 shows an example of a flowchart for data reception. This procedure should be followed for data reception after initializing SCI3.

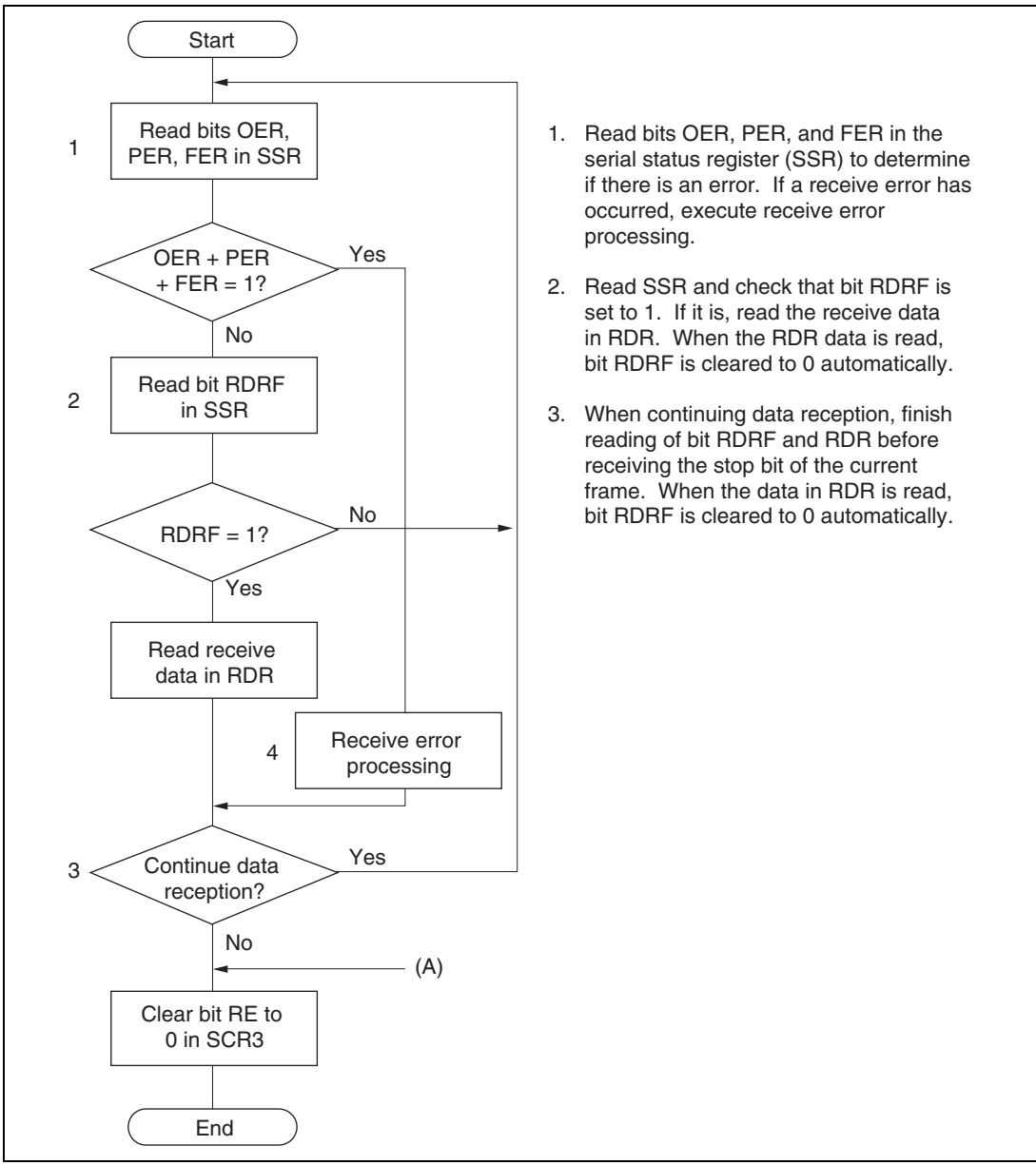



**Figure 10.8 Example of Data Reception Flowchart (Asynchronous Mode)**

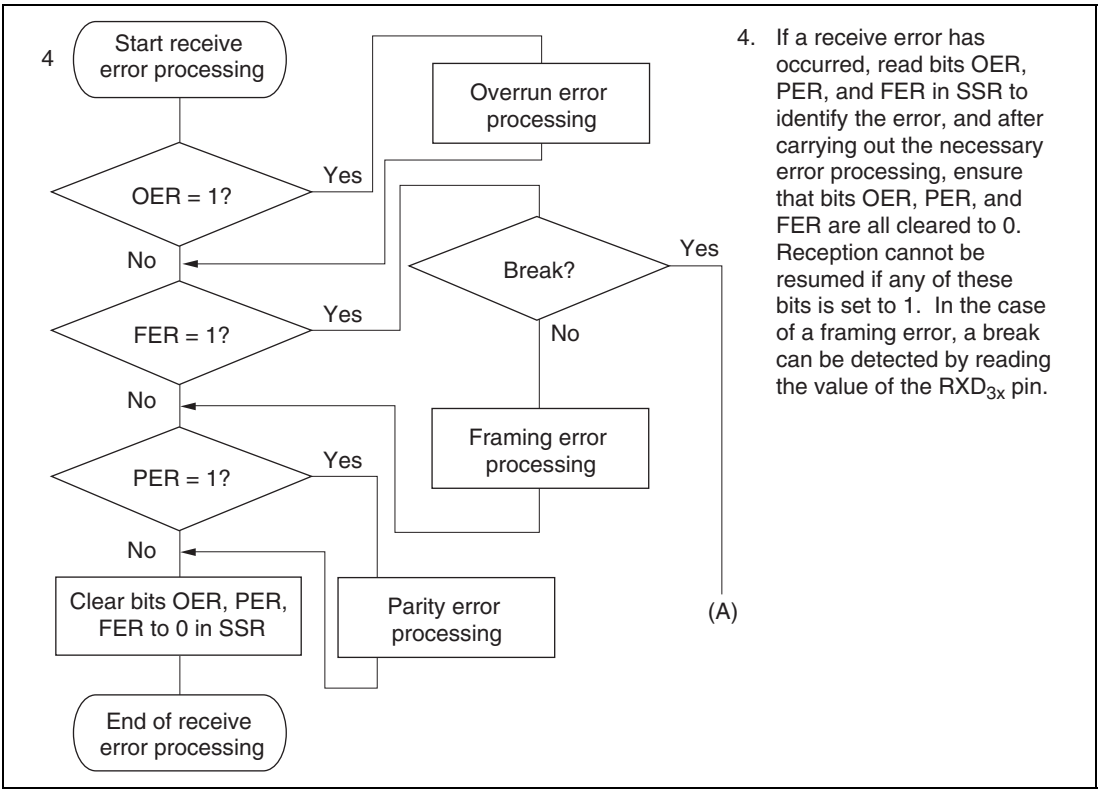



**Figure 10.8 Example of Data Reception Flowchart (Asynchronous Mode) (cont)**

SCI3 operates as follows when receiving data.

SCI3 monitors the communication line, and when it detects a 0 start bit, performs internal synchronization and begins reception. Reception is carried out in accordance with the relevant data transfer format in table 10.11. The received data is first placed in RSR in LSB-to-MSB order, and then the parity bit and stop bit(s) are received. SCI3 then carries out the following checks.

- Parity check
  SCI3 checks that the number of 1 bits in the receive data conforms to the parity (odd or even) set in bit PM in the serial mode register (SMR).
- Stop bit check
  SCI3 checks that the stop bit is 1. If two stop bits are used, only the first is checked.
- Status check
  SCI3 checks that bit RDRF is set to 0, indicating that the receive data can be transferred from RSR to RDR.

If no receive error is found in the above checks, bit RDRF is set to 1, and the receive data is stored in RDR. If bit RIE is set to 1 in SCR3, an RXI interrupt is requested. If the error checks identify a receive error, bit OER, PER, or FER is set to 1 depending on the kind of error. Bit RDRF retains its state prior to receiving the data. If bit RIE is set to 1 in SCR3, an ERI interrupt is requested.

Table 10.12 shows the conditions for detecting a receive error, and receive data processing.

Note: No further receive operations are possible while a receive error flag is set. Bits OER, FER, PER, and RDRF must therefore be cleared to 0 before resuming reception.

**Table 10.12 Receive Error Detection Conditions and Receive Data Processing**

| Receive Error | Abbr. | Detection Conditions | Receive Data Processing |
|---|---|---|---|
| Overrun error | OER | When the next date receive operation is completed while bit RDRF is still set to 1 in SSR | Receive data is not transferred from RSR to RDR |
| Framing error | FER | When the stop bit is 0 | Receive data is transferred from RSR to RDR |
| Parity error | PER | When the parity (odd or even) set in SMR is different from that of the received data | Receive data is transferred from RSR to RDR |

Figure 10.9 shows an example of the operation when receiving in asynchronous mode.

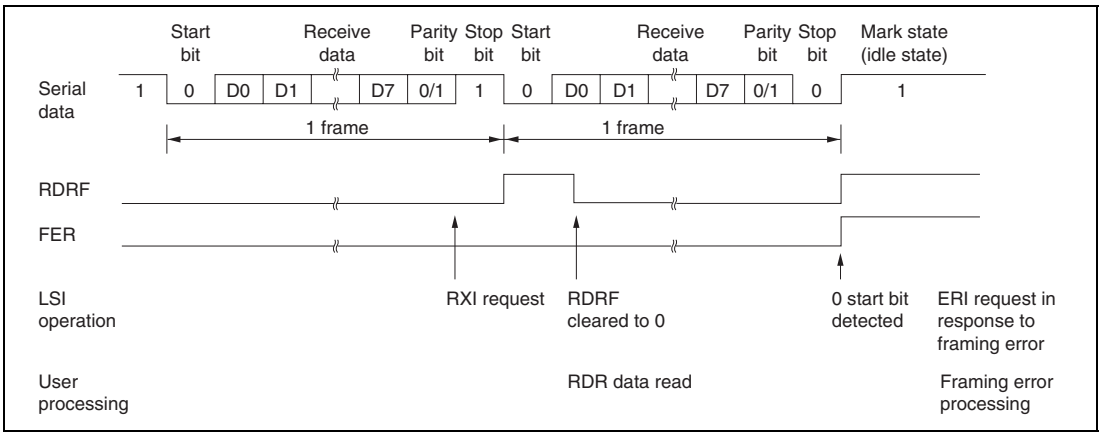



**Figure 10.9 Example of Operation when Receiving in Asynchronous Mode (8-bit data, parity, 1 stop bit)**

### 10.3.3 Operation in Synchronous Mode

In synchronous mode, SCI3 transmits and receives data in synchronization with clock pulses. This mode is suitable for high-speed serial communication.

SCI3 has separate transmission and reception units, allowing full-duplex communication with a shared clock.

As the transmission and reception units are both double-buffered, data can be written during transmission and read during reception, making possible continuous transmission and reception.

### (1) Data Transfer Format

The general data transfer format in asynchronous communication is shown in figure 10.10.

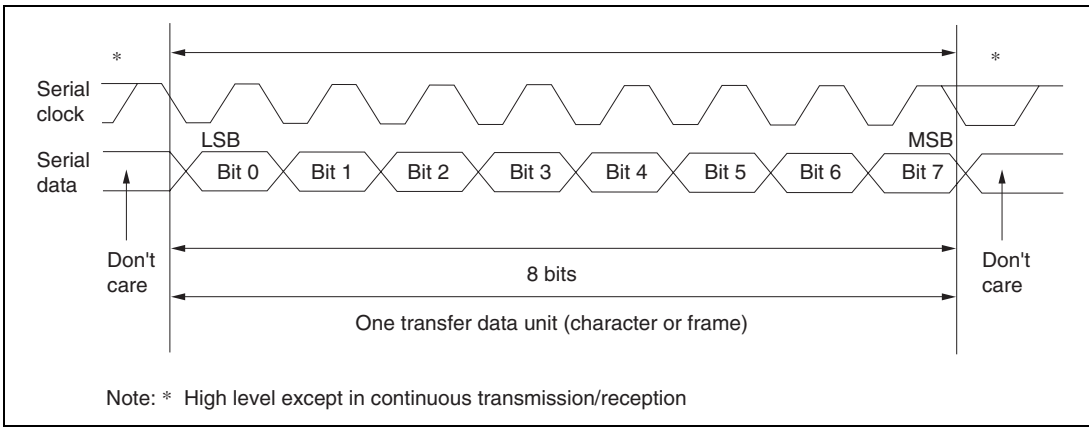



**Figure 10.10 Data Format in Synchronous Communication**

In synchronous communication, data on the communication line is output from one falling edge of the serial clock until the next falling edge. Data confirmation is guaranteed at the rising edge of the serial clock.

One transfer data character begins with the LSB and ends with the MSB. After output of the MSB, the communication line retains the MSB state.

When receiving in synchronous mode, SCI3 latches receive data at the rising edge of the serial clock.

The data transfer format uses a fixed 8-bit data length.

Parity bits cannot be added.

### (2) Clock

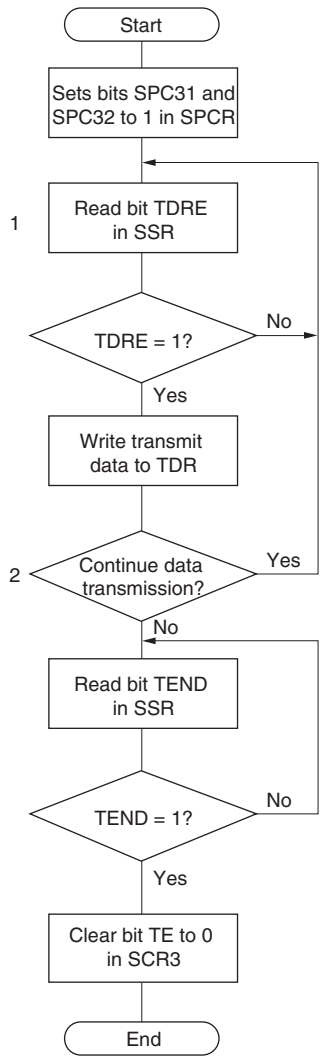
Either an internal clock generated by the baud rate generator or an external clock input at the $SCK_{3x}$ pin can be selected as the SCI3 serial clock. The selection is made by means of bit COM in SMR and bits CKE1 and CKE0 in SCR3. See table 10.9 for details on clock source selection.

When SCI3 operates on an internal clock, the serial clock is output at the $SCK_{3x}$ pin. Eight pulses of the serial clock are output in transmission or reception of one character, and when SCI3 is not transmitting or receiving, the clock is fixed at the high level.

### (3) Data Transfer Operations

**SCI3 initialization:** Data transfer on SCI3 first of all requires that SCI3 be initialized as described in 10.3.2.3. SCI3 initialization, and shown in figure 10.5.

**Transmitting:** Figure 10.11 shows an example of a flowchart for data transmission. This procedure should be followed for data transmission after initializing SCI3.




1. Read the serial status register (SSR) and check that bit TDRE is set to 1, then write transmit data to the transmit data register (TDR). When data is written to TDR, bit TDRE is cleared to 0 automatically, the clock is output, and data transmission is started. When clock output is selected, the clock is output and data transmission started when data is written to TDR.

2. When continuing data transmission, be sure to read TDRE = 1 to confirm that a write can be performed before writing data to TDR. When data is written to TDR, bit TDRE is cleared to 0 automatically.

**Figure 10.11 Example of Data Transmission Flowchart (Synchronous Mode)**

SCI3 operates as follows when transmitting data.

SCI3 monitors bit TDRE in SSR, and when it is cleared to 0, recognizes that data has been written to TDR and transfers data from TDR to TSR. It then sets bit TDRE to 1 and starts transmitting. If bit TIE in SCR3 is set to 1 at this time, a TXI request is made.

When clock output mode is selected, SCI3 outputs 8 serial clock pulses. When an external clock is selected, data is output in synchronization with the input clock.

Serial data is transmitted from the TXD3x pin in order from the LSB (bit 0) to the MSB (bit 7). When the MSB (bit 7) is sent, checks bit TDRE. If bit TDRE is cleared to 0, SCI3 transfers data from TDR to TSR, and starts transmission of the next frame. If bit TDRE is set to 1, SCI3 sets bit TEND to 1 in SSR, and after sending the MSB (bit 7), retains the MSB state. If bit TEIE in SCR3 is set to 1 at this time, a TEI request is made.

After transmission ends, the SCK pin is fixed at the high level.

Note: Transmission is not possible if an error flag (OER, FER, or PER) that indicates the data reception status is set to 1. Check that these error flags are all cleared to 0 before a transmit operation.

Figure 10.12 shows an example of the operation when transmitting in synchronous mode.

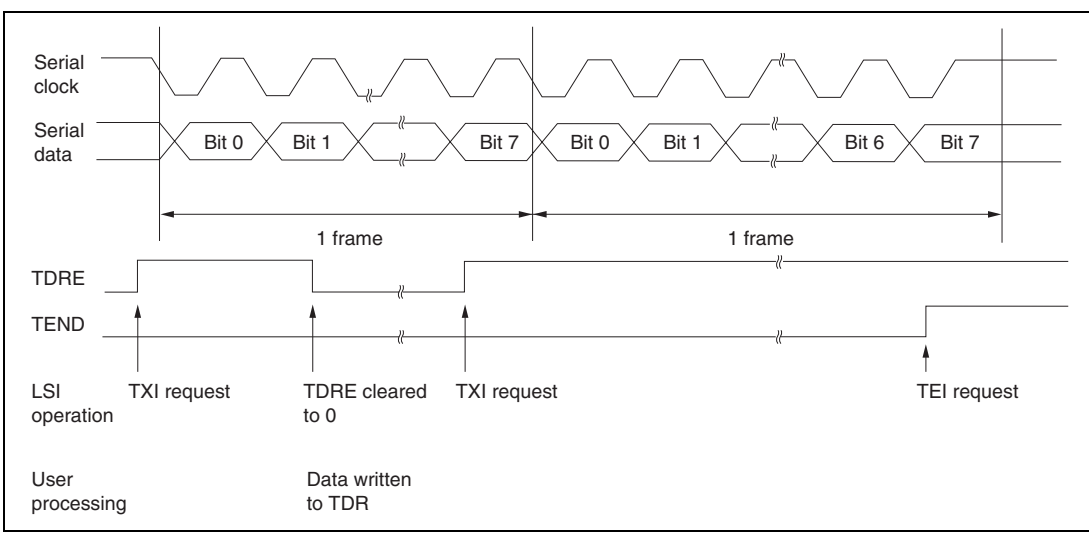



**Figure 10.12 Example of Operation when Transmitting in Synchronous Mode**

**Receiving:** Figure 10.13 shows an example of a flowchart for data reception. This procedure should be followed for data reception after initializing SCI3.

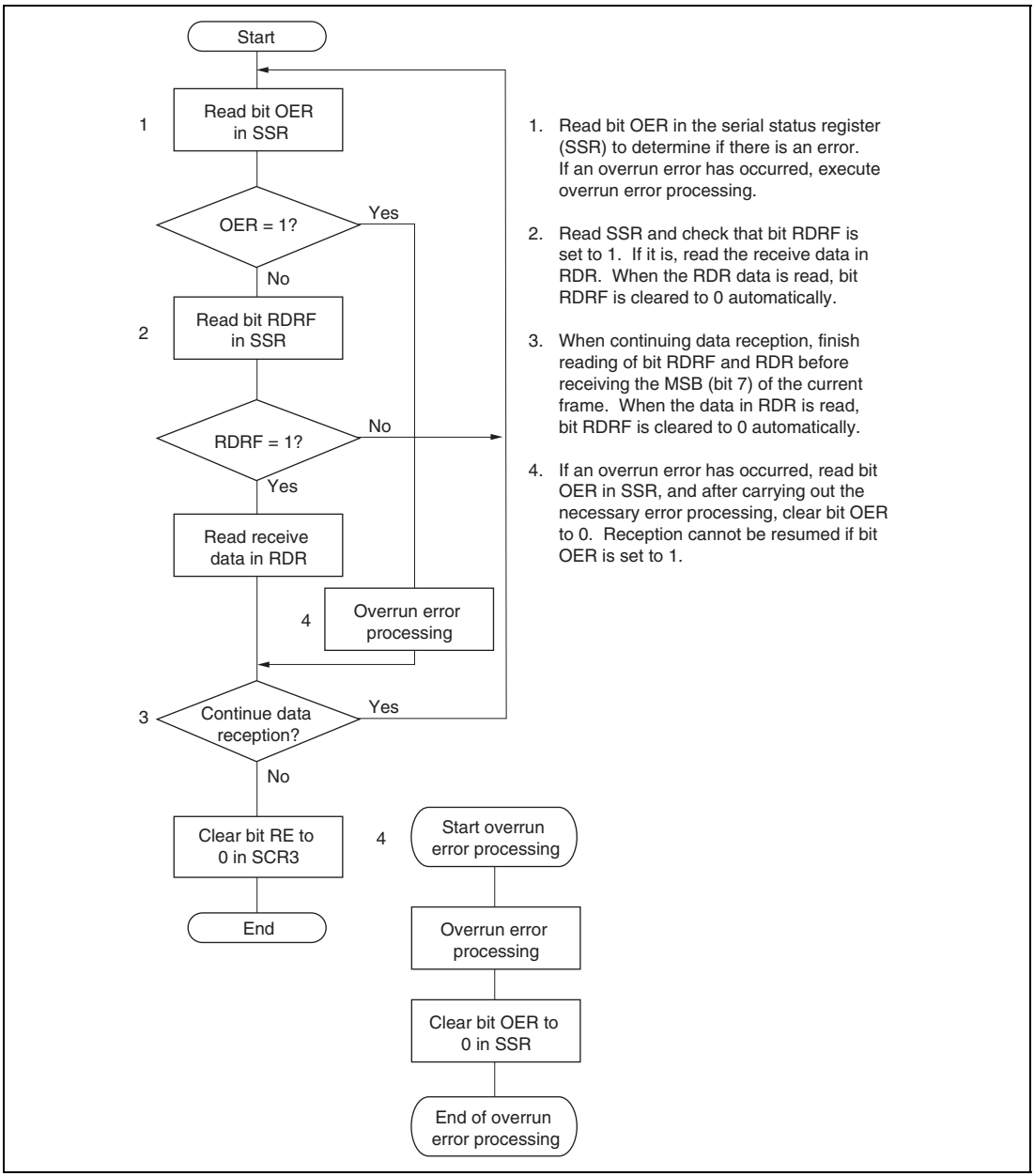



**Figure 10.13 Example of Data Reception Flowchart (Synchronous Mode)**

SCI3 operates as follows when receiving data.

SCI3 performs internal synchronization and begins reception in synchronization with the serial clock input or output.

The received data is placed in RSR in LSB-to-MSB order.

After the data has been received, SCI3 checks that bit RDRF is set to 0, indicating that the receive data can be transferred from RSR to RDR.

If this check shows that there is no overrun error, bit RDRF is set to 1, and the receive data is stored in RDR. If bit RIE is set to 1 in SCR3, an RXI interrupt is requested. If the check identifies an overrun error, bit OER is set to 1.

Bit RDRF remains set to 1. If bit RIE is set to 1 in SCR3, an ERI interrupt is requested.

See table 10.12 for the conditions for detecting a receive error, and receive data processing.

Note: No further receive operations are possible while a receive error flag is set. Bits OER, FER, PER, and RDRF must therefore be cleared to 0 before resuming reception.

Figure 10.14 shows an example of the operation when receiving in synchronous mode.

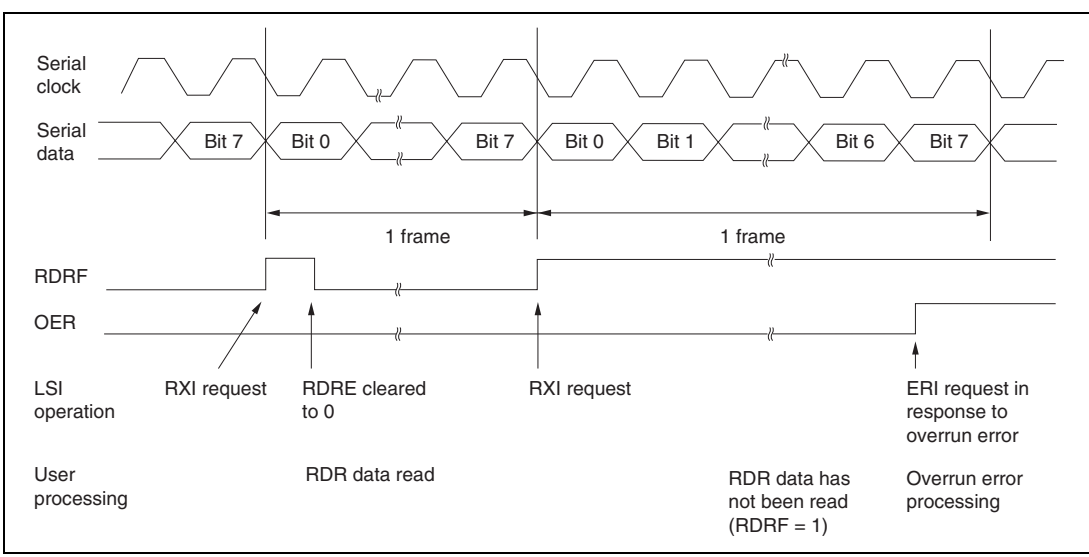
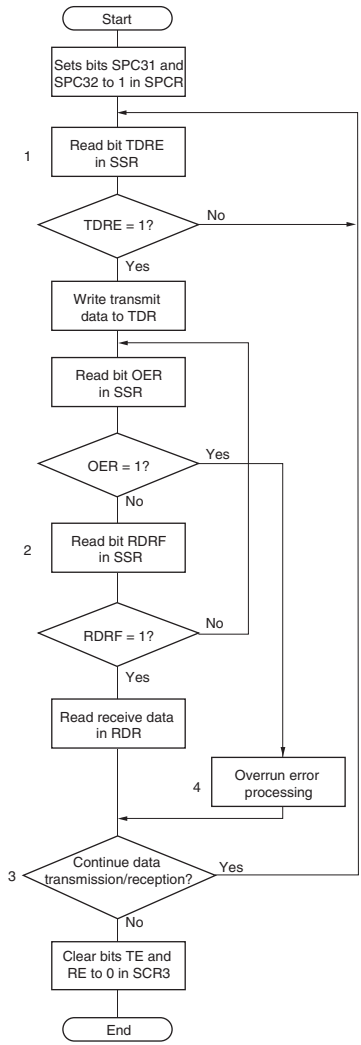



**Figure 10.14 Example of Operation when Receiving in Synchronous Mode**

**Simultaneous transmit/receive:** Figure 10.15 shows an example of a flowchart for a simultaneous transmit/receive operation. This procedure should be followed for simultaneous transmission/reception after initializing SCI3.




1. Read the serial status register (SSR) and check that bit TDRE is set to 1, then write transmit data to the transmit data register (TDR). When data is written to TDR, bit TDRE is cleared to 0 automatically.
2. Read SSR and check that bit RDRF is set to 1. If it is, read the receive data in RDR. When the RDR data is read, bit RDRF is cleared to 0 automatically.
3. When continuing data transmission/reception, finish reading of bit RDRF and RDR before receiving the MSB (bit 7) of the current frame. Before receiving the MSB (bit 7) of the current frame, also read TDRE = 1 to confirm that a write can be performed, then write data to TDR. When data is written to TDR, bit TDRE is cleared to 0 automatically, and when the data in RDR is read, bit RDRF is cleared to 0 automatically.
4. If an overrun error has occurred, read bit OER in SSR, and after carrying out the necessary error processing, clear bit OER to 0. Transmission and reception cannot be resumed if bit OER is set to 1.
   See figure 10.13 for details on overrun error processing.

**Figure 10.15 Example of Simultaneous Data Transmission/Reception Flowchart (Synchronous Mode)**

Notes: 1. When switching from transmission to simultaneous transmission/reception, check that SCI3 has finished transmitting and that bits TDRE and TEND are set to 1, clear bit TE to 0, and then set bits TE and RE to 1 simultaneously.

2. When switching from reception to simultaneous transmission/reception, check that SCI3 has finished receiving, clear bit RE to 0, then check that bit RDRF and the error flags (OER, FER, and PER) are cleared to 0, and finally set bits TE and RE to 1 simultaneously.

## 10.4 Interrupts

SCI3 can generate six kinds of interrupts: transmit end, transmit data empty, receive data full, and three receive error interrupts (overrun error, framing error, and parity error). These interrupts have the same vector address.

The various interrupt requests are shown in table 10.13.

**Table 10.13 SCI3 Interrupt Requests**

| Interrupt Abbr. | Interrupt Request | Vector Address |
|---|---|---|
| RXI | Interrupt request initiated by receive data full flag (RDRF) | H'0022/H'0024 |
| TXI | Interrupt request initiated by transmit data empty flag (TDRE) | |
| TEI | Interrupt request initiated by transmit end flag (TEND) | |
| ERI | Interrupt request initiated by receive error flag (OER, FER, PER) | |

Each interrupt request can be enabled or disabled by means of bits TIE and RIE in SCR3.

When bit TDRE is set to 1 in SSR, a TXI interrupt is requested. When bit TEND is set to 1 in SSR, a TEI interrupt is requested. These two interrupts are generated during transmission.

The initial value of bit TDRE in SSR is 1. Therefore, if the transmit data empty interrupt request (TXI) is enabled by setting bit TIE to 1 in SCR3 before transmit data is transferred to TDR, a TXI interrupt will be requested even if the transmit data is not ready.

Also, the initial value of bit TEND in SSR is 1. Therefore, if the transmit end interrupt request (TEI) is enabled by setting bit TEIE to 1 in SCR3 before transmit data is transferred to TDR, a TEI interrupt will be requested even if the transmit data has not been sent.

Effective use of these interrupt requests can be made by having processing that transfers transmit data to TDR carried out in the interrupt service routine.

To prevent the generation of these interrupt requests (TXI and TEI), on the other hand, the enable bits for these interrupt requests (bits TIE and TEIE) should be set to 1 after transmit data has been transferred to TDR.

When bit RDRF is set to 1 in SSR, an RXI interrupt is requested, and if any of bits OER, PER, and FER is set to 1, an ERI interrupt is requested. These two interrupt requests are generated during reception.

For further details, see section 3.3, Interrupts.

## 10.5 Application Notes

The following points should be noted when using SCI3.

### (1) Relation between Writes to TDR and Bit TDRE

Bit TDRE in the serial status register (SSR) is a status flag that indicates that data for serial transmission has not been prepared in TDR. When data is written to TDR, bit TDRE is cleared to 0 automatically. When SCI3 transfers data from TDR to TSR, bit TDRE is set to 1.

Data can be written to TDR irrespective of the state of bit TDRE, but if new data is written to TDR while bit TDRE is cleared to 0, the data previously stored in TDR will be lost of it has not yet been transferred to TSR. Accordingly, to ensure that serial transmission is performed dependably, you should first check that bit TDRE is set to 1, then write the transmit data to TDR once only (not two or more times).

### (2) Operation when a Number of Receive Errors Occur Simultaneously

If a number of receive errors are detected simultaneously, the status flags in SSR will be set to the states shown in table 10.14. If an overrun error is detected, data transfer from RSR to RDR will not be performed, and the receive data will be lost.

**Table 10.14 SSR Status Flag States and Receive Data Transfer**

| SSR Status Flags | | | | Receive Data Transfer | |
|---|---|---|---|---|---|
| RDRF* | OER | FER | PER | RSR → RDR | Receive Error Status |
| 1 | 1 | 0 | 0 | X | Overrun error |
| 0 | 0 | 1 | 0 | O | Framing error |
| 0 | 0 | 0 | 1 | O | Parity error |
| 1 | 1 | 1 | 0 | X | Overrun error + framing error |
| 1 | 1 | 0 | 1 | X | Overrun error + parity error |
| 0 | 0 | 1 | 1 | O | Framing error + parity error |
| 1 | 1 | 1 | 1 | X | Overrun error + framing error + parity error |

O : Receive data is transferred from RSR to RDR.

X : Receive data is not transferred from RSR to RDR.

Note: * Bit RDRF retains its state prior to data reception. However, note that if RDR is read after an overrun error has occurred in a frame because reading of the receive data in the previous frame was delayed, RDRF will be cleared to 0.

### (3) Break Detection and Processing

When a framing error is detected, a break can be detected by reading the value of the $RXD_{3x}$ pin directly. In a break, the input from the $RXD_{3x}$ pin becomes all 0s, with the result that bit FER is set and bit PER may also be set.

SCI3 continues the receive operation even after receiving a break. Note, therefore, that even though bit FER is cleared to 0 it will be set to 1 again.

### (4) Mark State and Break Detection

When bit TE is cleared to 0, the $TXD_{3x}$ pin functions as an I/O port whose input/output direction and level are determined by PDR and PCR. This fact can be used to set the $TXD_{3x}$ pin to the mark state, or to detect a break during transmission.

To keep the communication line in the mark state (1 state) until bit TE is set to 1, set PCR = 1 and PDR = 1. Since bit TE is cleared to 0 at this time, the $TXD_{3x}$ pin functions as an I/O port and 1 is output.

To detect a break, clear bit TE to 0 after setting PCR = 1 and PDR = 0.

When bit TE is cleared to 0, the transmission unit is initialized regardless of the current transmission state, the $TXD_{3x}$ pin functions as an I/O port, and 0 is output from the $TXD_{3x}$ pin.

### (5) Receive Error Flags and Transmit Operation (Synchronous Mode Only)

When a receive error flag (OER, PER, or FER) is set to 1, transmission cannot be started even if bit TDRE is cleared to 0. The receive error flags must be cleared to 0 before starting transmission.

Note also that receive error flags cannot be cleared to 0 even if bit RE is cleared to 0.

### (6) Receive Data Sampling Timing and Receive Margin in Asynchronous Mode

In asynchronous mode, SCI3 operates on a basic clock with a frequency 16 times the transfer rate. When receiving, SCI3 performs internal synchronization by sampling the falling edge of the start bit with the basic clock. Receive data is latched internally at the 8th rising edge of the basic clock. This is illustrated in figure 10.16.

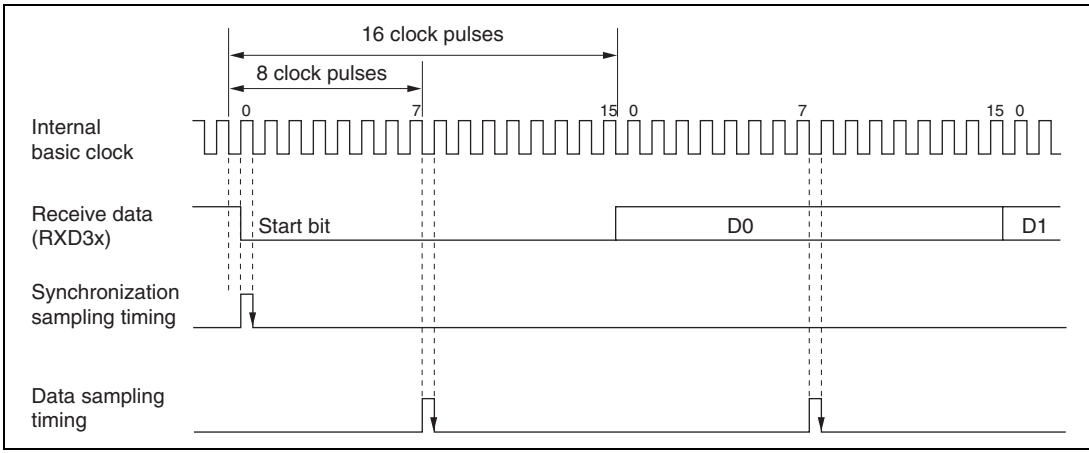



**Figure 10.16 Receive Data Sampling Timing in Asynchronous Mode**

Consequently, the receive margin in asynchronous mode can be expressed as shown in equation (1).

$$M = \{(0.5 - \frac{1}{2N}) - \frac{D - 0.5}{N} - (L - 0.5)\,F\} \times 100\ [\%] \qquad \text{..... Equation (1)}$$

where M: Receive margin (%)
N: Ratio of bit rate to clock (N = 16)
D: Clock duty (D = 0.5 to 1.0)
L: Frame length (L = 9 to 12)
F: Absolute value of clock frequency deviation

Substituting 0 for F (absolute value of clock frequency deviation) and 0.5 for D (clock duty) in equation (1), a receive margin of 46.875% is given by equation (2).

When D = 0.5 and F = 0,

$$M = \{0.5 — 1/(2 \times 16)\} \times 100\ [\%]$$
$$= 46.875\% \qquad \text{..... Equation (2)}$$

However, this is only a computed value, and a margin of 20% to 30% should be allowed when carrying out system design.

### (7) Relation between RDR Reads and Bit RDRF

In a receive operation, SCI3 continually checks the RDRF flag. If bit RDRF is cleared to 0 when reception of one frame ends, normal data reception is completed. If bit RDRF is set to 1, this indicates that an overrun error has occurred.

When the contents of RDR are read, bit RDRF is cleared to 0 automatically. Therefore, if bit RDR is read more than once, the second and subsequent read operations will be performed while bit RDRF is cleared to 0. Note that, when an RDR read is performed while bit RDRF is cleared to 0, if the read operation coincides with completion of reception of a frame, the next frame of data may be read. This is illustrated in figure 10.17.

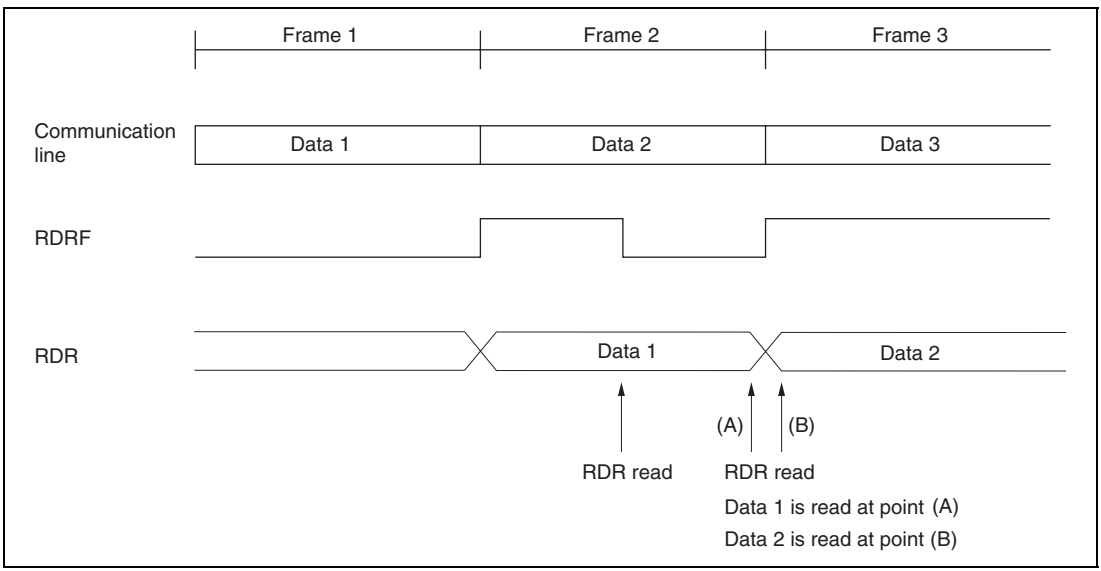



**Figure 10.17 Relation between RDR Read Timing and Data**

In this case, only a single RDR read operation (not two or more) should be performed after first checking that bit RDRF is set to 1. If two or more reads are performed, the data read the first time should be transferred to RAM, etc., and the RAM contents used. Also, ensure that there is sufficient margin in an RDR read operation before reception of the next frame is completed. To be precise in terms of timing, the RDR read should be completed before bit 7 is transferred in synchronous mode, or before the STOP bit is transferred in asynchronous mode.

### (8) Transmit and Receive Operations when Making a State Transition

Make sure that transmit and receive operations have completely finished before carrying out state transition processing.

### (9) Switching $SCK_{3x}$ Function

If pin $SCK_{3x}$ is used as a clock output pin by SCI3 in synchronous mode and is then switched to a general input/output pin (a pin with a different function), the pin outputs a low level signal for half a system clock (ϕ) cycle immediately after it is switched.

This can be prevented by either of the following methods according to the situation.

a. When an $SCK_{3x}$ function is switched from clock output to non clock-output
   When stopping data transfer, issue one instruction to clear bits TE and RE to 0 and to set bits CKE1 and CKE0 in SCR3 to 1 and 0, respectively. In this case, bit COM in SMR should be left 1. The above prevents $SCK_{3x}$ from being used as a general input/output pin. To avoid an intermediate level of voltage from being applied to $SCK_{3x}$, the line connected to $SCK_{3x}$ should be pulled up to the $V_{CC}$ level via a resistor, or supplied with output from an external device.
b. When an $SCK_{3x}$ function is switched from clock output to general input/output
   When stopping data transfer,
   (i) Issue one instruction to clear bits TE and RE to 0 and to set bits CKE1 and CKE0 in SCR3 to 1 and 0, respectively.
   (ii) Clear bit COM in SMR to 0
   (iii) Clear bits CKE1 and CKE0 in SCR3 to 0
   Note that special care is also needed here to avoid an intermediate level of voltage from being applied to $SCK_{3x}$.

### (10) Setup at Subactive or Subsleep Mode

At subactive or subsleep mode, SCI3 becomes possible use only at CPU clock is ϕw/2.

# Section 11 14-Bit PWM

## 11.1 Overview

This LSI is provided with a 14-bit PWM (pulse width modulator) on-chip, which can be used as a D/A converter by connecting a low-pass filter.

### 11.1.1 Features

Features of the 14-bit PWM are as follows.

- Choice of two conversion periods
  Any of the following four conversion periods can be chosen:
  — 131,072/ϕ, with a minimum modulation width of 8/ϕ (PWCR1 = 1, PWCR0 = 1)
  — 65,536/ϕ, with a minimum modulation width of 4/ϕ (PWCR1 = 1, PWCR0 = 0)
  — 32,768/ϕ, with a minimum modulation width of 2/ϕ (PWCR1 = 0, PWCR0 = 1)
  — 16,384/ϕ, with a minimum modulation width of 1/ϕ (PWCR1 = 0, PWCR0 = 0)
- Pulse division method for less ripple
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### 11.1.2 Block Diagram

Figure 11.1 shows a block diagram of the 14-bit PWM.

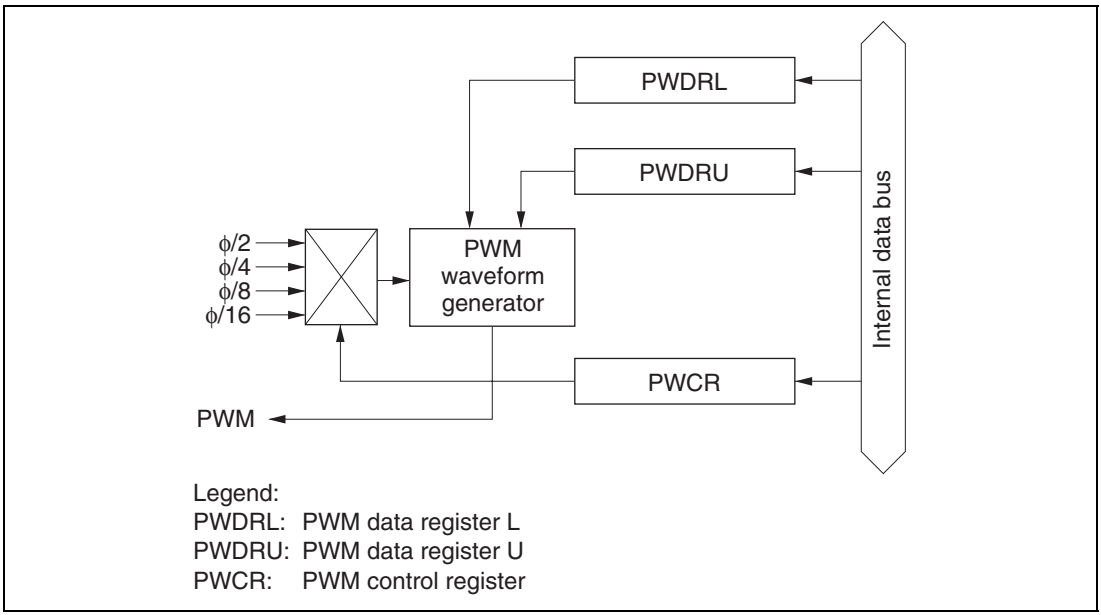



**Figure 11.1 Block Diagram of the 14 Bit PWM**

### 11.1.3 Pin Configuration

Table 11.1 shows the output pin assigned to the 14-bit PWM.

**Table 11.1 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| PWM output pin | PWM | Output | Pulse-division PWM waveform output |

### 11.1.4 Register Configuration

Table 11.2 shows the register configuration of the 14-bit PWM.

**Table 11.2 Register Configuration**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| PWM control register | PWCR | W | H'FC | H'FFD0 |
| PWM data register U | PWDRU | W | H'C0 | H'FFD1 |
| PWM data register L | PWDRL | W | H'00 | H'FFD2 |
| Clock stop register 2 | CKSTPR2 | R/W | H'FF | H'FFFB |

## 11.2 Register Descriptions

### 11.2.1 PWM Control Register (PWCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PWCR1 | PWCR0 |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Read/Write | — | — | — | — | — | — | W | W |

PWCR is an 8-bit write-only register for input clock selection.

Upon reset, PWCR is initialized to H'FC.

**Bits 7 to 2:** Reserved bits

Bits 7 to 2 are reserved; they are always read as 1, and cannot be modified.

**Bits 1 and 0:** Clock select 1 and 0 (PWCR1, PWCR0)

Bits 1 and 0 select the clock supplied to the 14-bit PWM. These bits are write-only bits; they are always read as 1.

| Bit 1 PWCR1 | Bit 0 PWCR0 | Description |
|---|---|---|
| 0 | 0 | The input clock is $\phi/2$ ($t\phi^* = 2/\phi$) (initial value)<br>The conversion period is $16{,}384/\phi$, with a minimum modulation width of $1/\phi$ |
| 0 | 1 | The input clock is $\phi/4$ ($t\phi^* = 4/\phi$)<br>The conversion period is $32{,}768/\phi$, with a minimum modulation width of $2/\phi$ |
| 1 | 0 | The input clock is $\phi/8$ ($t\phi^* = 8/\phi$)<br>The conversion period is $65{,}536/\phi$, with a minimum modulation width of $4/\phi$ |
| 1 | 1 | The input clock is $\phi/16$ ($t\phi^* = 16/\phi$)<br>The conversion period is $131{,}072/\phi$, with a minimum modulation width of $8/\phi$ |

Note: * Period of PWM input clock.

### 11.2.2 PWM Data Registers U and L (PWDRU, PWDRL)

PWDRU

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | PWDRU5 | PWDRU4 | PWDRU3 | PWDRU2 | PWDRU1 | PWDRU0 |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | W | W | W | W | W | W |

PWDRL

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWDRL7 | PWDRL6 | PWDRL5 | PWDRL4 | PWDRL3 | PWDRL2 | PWDRL1 | PWDRL0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

PWDRU and PWDRL form a 14-bit write-only register, with the upper 6 bits assigned to PWDRU and the lower 8 bits to PWDRL. The value written to PWDRU and PWDRL gives the total high-level width of one PWM waveform cycle.

When 14-bit data is written to PWDRU and PWDRL, the register contents are latched in the PWM waveform generator, updating the PWM waveform generation data. The 14-bit data should always be written in the following sequence:

1. Write the lower 8 bits to PWDRL.
2. Write the upper 6 bits to PWDRU.

PWDRU and PWDRL are write-only registers. If they are read, all bits are read as 1.

Upon reset, PWDRU and PWDRL are initialized to H'C000.

### 11.2.3 Clock Stop Register 2 (CKSTPR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | AECKSTP | WDCKSTP | PWCKSTP | LDCKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

CKSTPR2 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to the PWM is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 1:** PWM module standby mode control (PWCKSTP)

Bit 1 controls setting and clearing of module standby mode for the PWM.

| PWCKSTP | Description | |
|---|---|---|
| 0 | PWM is set to module standby mode | |
| 1 | PWM module standby mode is cleared | (initial value) |

## 11.3 Operation

### 11.3.1 Operation

When using the 14-bit PWM, set the registers in the following sequence.

1. Set bit PWM in port mode register 3 (PMR3) to 1 so that pin $P3_0$/PWM is designated for PWM output.
2. Set bits PWCR1 and PWCR0 in the PWM control register (PWCR) to select a conversion period of 131,072/ϕ (PWCR1 = 1, PWCR0 = 1), 65,536/ϕ (PWCR1 = 1, PWCR0 = 0), 32,768/ϕ (PWCR1 = 0, PWCR0 = 1), or 16,384/ϕ (PWCR1 = 0, PWCR0 = 0).
3. Set the output waveform data in PWM data registers U and L (PWDRU/L). Be sure to write in the correct sequence, first PWDRL then PWDRU. When data is written to PWDRU, the data in these registers will be latched in the PWM waveform generator, updating the PWM waveform generation in synchronization with internal signals.

One conversion period consists of 64 pulses, as shown in figure 11.2. The total of the high-level pulse widths during this period ($T_H$) corresponds to the data in PWDRU and PWDRL. This relation can be represented as follows.

$$T_H = (\text{data value in PWDRU and PWDRL} + 64) \times t_\phi/2$$

where tϕ is the PWM input clock period: 2/ϕ (PWCR = H'0), 4/ϕ (PWCR = H'1), 8/ϕ (PWCR = H'2), or 16/ϕ (PWCR = H'3).

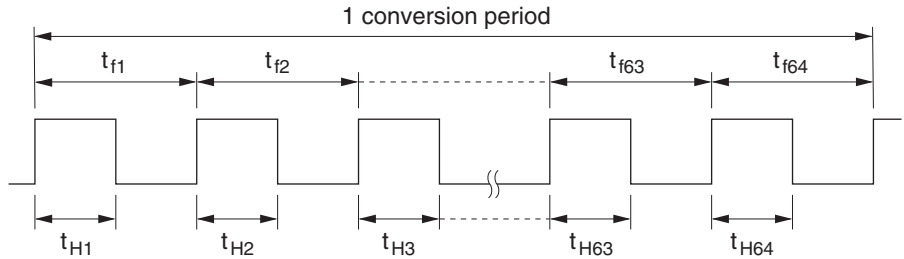
Example: Settings in order to obtain a conversion period of 32,768 μs:

When PWCR1 = 0 and PWCR0 = 0, the conversion period is 16,384/ϕ, so ϕ must be 0.5 MHz. In this case, tfn = 512 μs, with 1/ϕ (resolution) = 2.0 μs.

When PWCR1 = 0 and PWCR0 = 1, the conversion period is 32,768/ϕ, so ϕ must be 1 MHz. In this case, tfn = 512 μs, with 2/ϕ (resolution) = 2.0 μs.

When PWCR1 = 1 and PWCR0 = 0, the conversion period is 65,536/ϕ , so ϕ must be 2 MHz. In this case, tfn = 512 μs, with 4/ϕ (resolution) = 2.0 μs.

Accordingly, for a conversion period of 32,768 μs, the system clock frequency (ϕ) must be 0.5 MHz, 1 MHz, or 2 MHz.




$T_H = t_{H1} + t_{H2} + t_{H3} + \ldots\ldots t_{H64}$

$t_{f1} = t_{f2} = t_{f3} \ldots\ldots = t_{f64}$

**Figure 11.2 PWM Output Waveform**

### 11.3.2 PWM Operation Modes

PWM operation modes are shown in table 11.3.

**Table 11.3 PWM Operation Modes**

| Operation Mode | Reset | Active | Sleep | Watch | Subactive | Subsleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|
| PWCR | Reset | Functions | Functions | Held | Held | Held | Held | Held |
| PWDRU | Reset | Functions | Functions | Held | Held | Held | Held | Held |
| PWDRL | Reset | Functions | Functions | Held | Held | Held | Held | Held |

# Section 12 A/D Converter

## 12.1 Overview

This LSI includes on-chip a resistance-ladder-based successive-approximation analog-to-digital converter, and can convert up to 8 channels of analog input.

### 12.1.1 Features

The A/D converter has the following features.

- 10-bit resolution
- Eight input channels
- Conversion time: approx. 12.4 μs per channel (at 5 MHz operation)
- Built-in sample-and-hold function
- Interrupt requested on completion of A/D conversion
- A/D conversion can be started by external trigger input
- Use of module standby mode enables this module to be placed in standby mode independently when not used.

### 12.1.2 Block Diagram

Figure 12.1 shows a block diagram of the A/D converter.

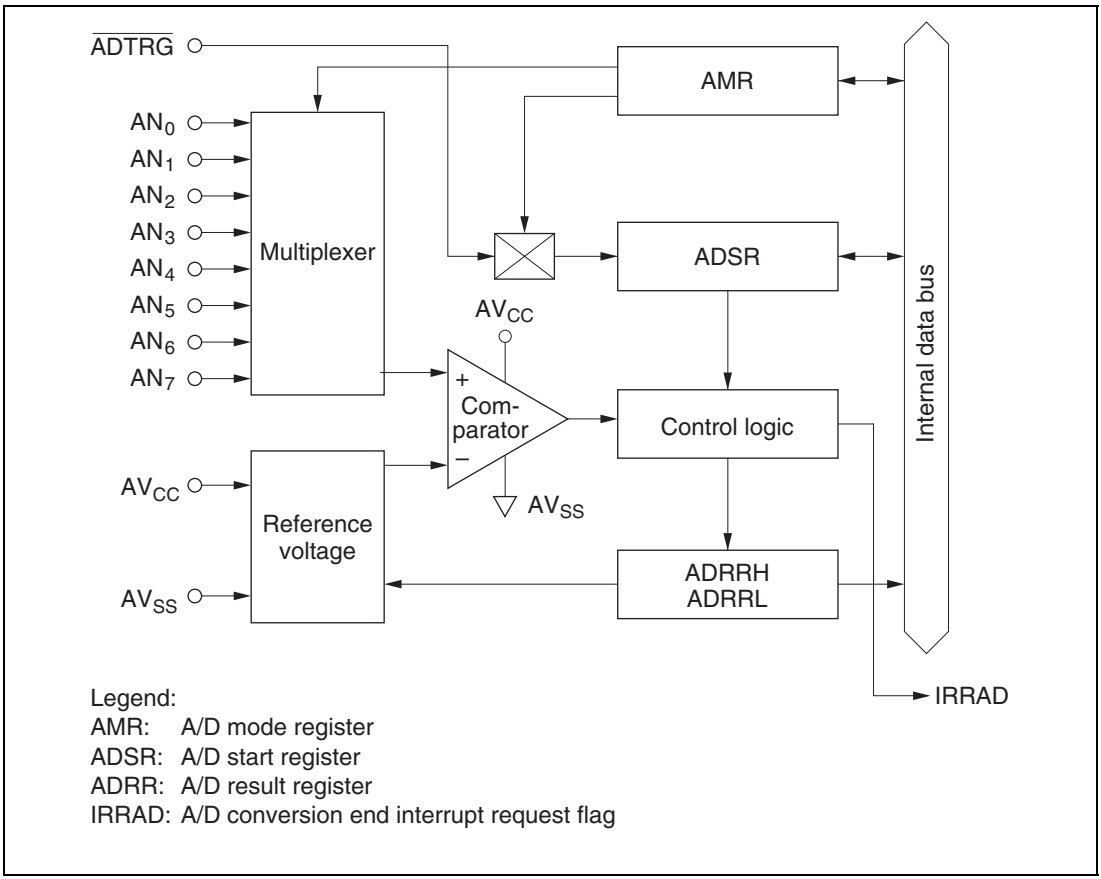



**Figure 12.1 Block Diagram of the A/D Converter**

### 12.1.3 Pin Configuration

Table 12.1 shows the A/D converter pin configuration.

**Table 12.1 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Analog power supply | $AV_{CC}$ | Input | Power supply and reference voltage of analog part |
| Analog ground | $AV_{SS}$ | Input | Ground and reference voltage of analog part |
| Analog input 0 | $AN_0$ | Input | Analog input channel 0 |
| Analog input 1 | $AN_1$ | Input | Analog input channel 1 |
| Analog input 2 | $AN_2$ | Input | Analog input channel 2 |
| Analog input 3 | $AN_3$ | Input | Analog input channel 3 |
| Analog input 4 | $AN_4$ | Input | Analog input channel 4 |
| Analog input 5 | $AN_5$ | Input | Analog input channel 5 |
| Analog input 6 | $AN_6$ | Input | Analog input channel 6 |
| Analog input 7 | $AN_7$ | Input | Analog input channel 7 |
| External trigger input | $\overline{ADTRG}$ | Input | External trigger input for starting A/D conversion |

### 12.1.4 Register Configuration

Table 12.2 shows the A/D converter register configuration.

**Table 12.2 Register Configuration**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| A/D mode register | AMR | R/W | H'30 | H'FFC6 |
| A/D start register | ADSR | R/W | H'7F | H'FFC7 |
| A/D result register H | ADRRH | R | Not fixed | H'FFC4 |
| A/D result register L | ADRRL | R | Not fixed | H'FFC5 |
| Clock stop register 1 | CKSTPR1 | R/W | H'FF | H'FFFA |

## 12.2 Register Descriptions

### 12.2.1 A/D Result Registers (ADRRH, ADRRL)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADR9 | ADR8 | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 | — | — | — | — | — | — |
| Initial value | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | Not fixed | — | — | — | — | — | — |
| Read/Write | R | R | R | R | R | R | R | R | R | R | — | — | — | — | — | — |
| | ADRRH | | | | | | | | ADRRL | | | | | | | |

ADRRH and ADRRL together comprise a 16-bit read-only register for holding the results of analog-to-digital conversion. The upper 8 bits of the data are held in ADRRH, and the lower 2 bits in ADRRL.

ADRRH and ADRRL can be read by the CPU at any time, but the ADRRH and ADRRL values during A/D conversion are not fixed. After A/D conversion is complete, the conversion result is stored as 10-bit data, and this data is held until the next conversion operation starts.

ADRRH and ADRRL are not cleared on reset.

### 12.2.2 A/D Mode Register (AMR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKS | TRGE | — | — | CH3 | CH2 | CH1 | CH0 |
| Initial value | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | — | — | R/W | R/W | R/W | R/W |

AMR is an 8-bit read/write register for specifying the A/D conversion speed, external trigger option, and the analog input pins.

Upon reset, AMR is initialized to H'30.

**Bit 7:** Clock select (CKS)

Bit 7 sets the A/D conversion speed.

| Bit 7 | | Conversion Time (Active (High-Speed) Mode)* | |
|---|---|---|---|
| **CKS** | **Conversion Period** | **ϕ = 1 MHz** | **ϕ = 5 MHz** |
| 0 | 62/ϕ (initial value) | 62 μs | 12.4 μs |
| 1 | 31/ϕ | 31 μs | — |

Note: * For information on conversion time settings for which operation is guaranteed, see section 16, Electrical Characteristics.

**Bit 6:** External trigger select (TRGE)

Bit 6 enables or disables the start of A/D conversion by external trigger input.

| Bit 6 TRGE | Description | |
|---|---|---|
| 0 | Disables start of A/D conversion by external trigger | (initial value) |
| 1 | Enables start of A/D conversion by rising or falling edge of external trigger at pin $\overline{\text{ADTRG}}$* | |

Note: * The external trigger ($\overline{\text{ADTRG}}$) edge is selected by bit IEG4 of IEGR. See (1) IRQ Edge Select Register (IEGR) in section 3.3.2, Interrupt Control Registers, for details.

**Bits 5 and 4:** Reserved bits

Bits 5 and 4 are reserved; they are always read as 1, and cannot be modified.

**Bits 3 to 0:** Channel select (CH3 to CH0)

Bits 3 to 0 select the analog input channel.

The channel selection should be made while bit ADSF is cleared to 0.

| Bit 3<br>CH3 | Bit 2<br>CH2 | Bit 1<br>CH1 | Bit 0<br>CH0 | Analog Input Channel | |
|---|---|---|---|---|---|
| 0 | 0 | * | * | No channel selected | (initial value) |
| 0 | 1 | 0 | 0 | AN0 | |
| 0 | 1 | 0 | 1 | AN1 | |
| 0 | 1 | 1 | 0 | AN2 | |
| 0 | 1 | 1 | 1 | AN3 | |
| 1 | 0 | 0 | 0 | AN4 | |
| 1 | 0 | 0 | 1 | AN5 | |
| 1 | 0 | 1 | 0 | AN6 | |
| 1 | 0 | 1 | 1 | AN7 | |
| 1 | 1 | * | * | Setting prohibited | |

Note: * Don’t care

### 12.2.3 A/D Start Register (ADSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADSF | — | — | — | — | — | — | — |
| Initial value | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | — | — | — | — | — | — | — |

The A/D start register (ADSR) is an 8-bit read/write register for starting and stopping A/D conversion.

A/D conversion is started by writing 1 to the A/D start flag (ADSF) or by input of the designated edge of the external trigger signal, which also sets ADSF to 1. When conversion is complete, the converted data is set in ADRRH and ADRRL, and at the same time ADSF is cleared to 0.

**Bit 7:** A/D start flag (ADSF)

Bit 7 controls and indicates the start and end of A/D conversion.

| Bit 7<br>ADSF | Description | |
|---|---|---|
| 0 | Read: Indicates the completion of A/D conversion | (initial value) |
| | Write: Stops A/D conversion | |
| 1 | Read: Indicates A/D conversion in progress | |
| | Write: Starts A/D conversion | |

**Bits 6 to 0:** Reserved bits

Bits 6 to 0 are reserved; they are always read as 1, and cannot be modified.

### 12.2.4 Clock Stop Register 1 (CKSTPR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CKSTPR1 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to the A/D converter is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 4:** A/D converter module standby mode control (ADCKSTP)

Bit 4 controls setting and clearing of module standby mode for the A/D converter.

| ADCKSTP | Description | |
|---|---|---|
| 0 | A/D converter is set to module standby mode | |
| 1 | A/D converter module standby mode is cleared | (initial value) |

## 12.3 Operation

### 12.3.1 A/D Conversion Operation

The A/D converter operates by successive approximations, and yields its conversion result as 10-bit data.

A/D conversion begins when software sets the A/D start flag (bit ADSF) to 1. Bit ADSF keeps a value of 1 during A/D conversion, and is cleared to 0 automatically when conversion is complete.

The completion of conversion also sets bit IRRAD in interrupt request register 2 (IRR2) to 1. An A/D conversion end interrupt is requested if bit IENAD in interrupt enable register 2 (IENR2) is set to 1.

If the conversion time or input channel needs to be changed in the A/D mode register (AMR) during A/D conversion, bit ADSF should first be cleared to 0, stopping the conversion operation, in order to avoid malfunction.

### 12.3.2 Start of A/D Conversion by External Trigger Input

The A/D converter can be made to start A/D conversion by input of an external trigger signal. External trigger input is enabled at pin $\overline{\text{ADTRG}}$ when bit IRQ4 in PMR1 is set to 1 and bit TRGE in AMR is set to 1. Then when the input signal edge designated in bit IEG4 of interrupt edge select register (IEGR) is detected at pin $\overline{\text{ADTRG}}$, bit ADSF in ADSR will be set to 1, starting A/D conversion.

Figure 12.2 shows the timing.

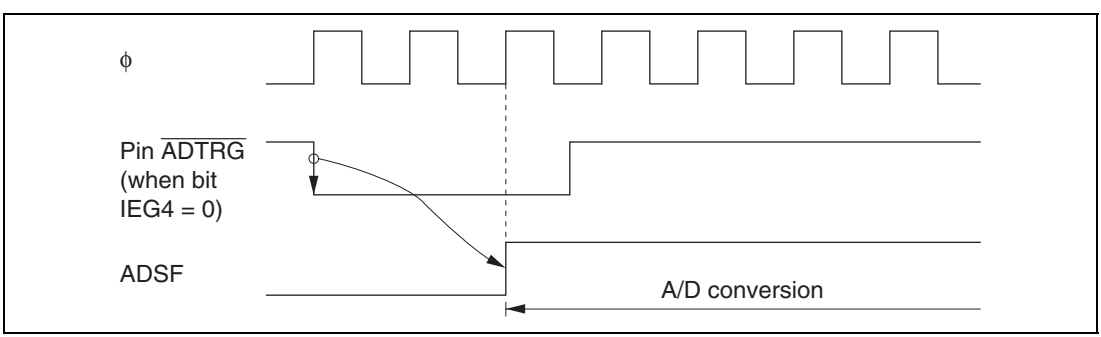



**Figure 12.2 External Trigger Input Timing**

### 12.3.3 A/D Converter Operation Modes

A/D converter operation modes are shown in table 12.3.

**Table 12.3 A/D Converter Operation Modes**

| Operation Mode | Reset | Active | Sleep | Watch | Subactive | Subsleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|
| AMR | Reset | Functions | Functions | Held | Held | Held | Held | Held |
| ADSR | Reset | Functions | Functions | Held | Held | Held | Held | Held |
| ADRRH | Held* | Functions | Functions | Held | Held | Held | Held | Held |
| ADRRL | Held* | Functions | Functions | Held | Held | Held | Held | Held |

Note: * Undefined in a power-on reset.

## 12.4 Interrupts

When A/D conversion ends (ADSF changes from 1 to 0), bit IRRAD in interrupt request register 2 (IRR2) is set to 1.

A/D conversion end interrupts can be enabled or disabled by means of bit IENAD in interrupt enable register 2 (IENR2).

For further details see section 3.3, Interrupts.

## 12.5 Typical Use

An example of how the A/D converter can be used is given below, using channel 1 (pin AN1) as the analog input channel. Figure 12.3 shows the operation timing.

1. Bits CH3 to CH0 of the A/D mode register (AMR) are set to 0101, making pin $AN_1$ the analog input channel. A/D interrupts are enabled by setting bit IENAD to 1, and A/D conversion is started by setting bit ADSF to 1.
2. When A/D conversion is complete, bit IRRAD is set to 1, and the A/D conversion result is stored is stored in ADRRH and ADRRL. At the same time ADSF is cleared to 0, and the A/D converter goes to the idle state.
3. Bit IENAD = 1, so an A/D conversion end interrupt is requested.
4. The A/D interrupt handling routine starts.
5. The A/D conversion result is read and processed.
6. The A/D interrupt handling routine ends.

If ADSF is set to 1 again afterward, A/D conversion starts and steps 2 to 6 take place.

Figures 12.4 and 12.5 show flow charts of procedures for using the A/D converter.

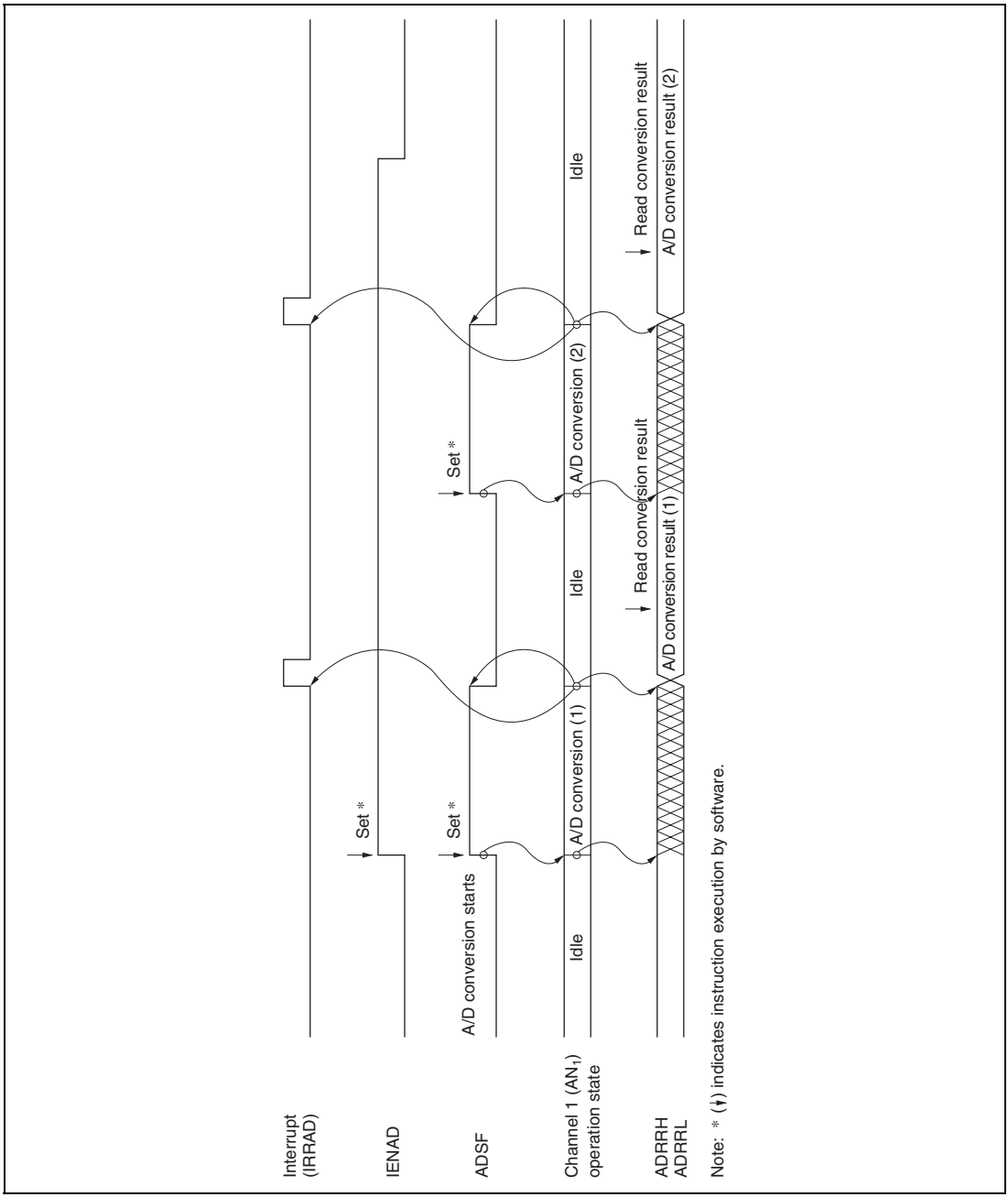



**Figure 12.3 Typical A/D Converter Operation Timing**

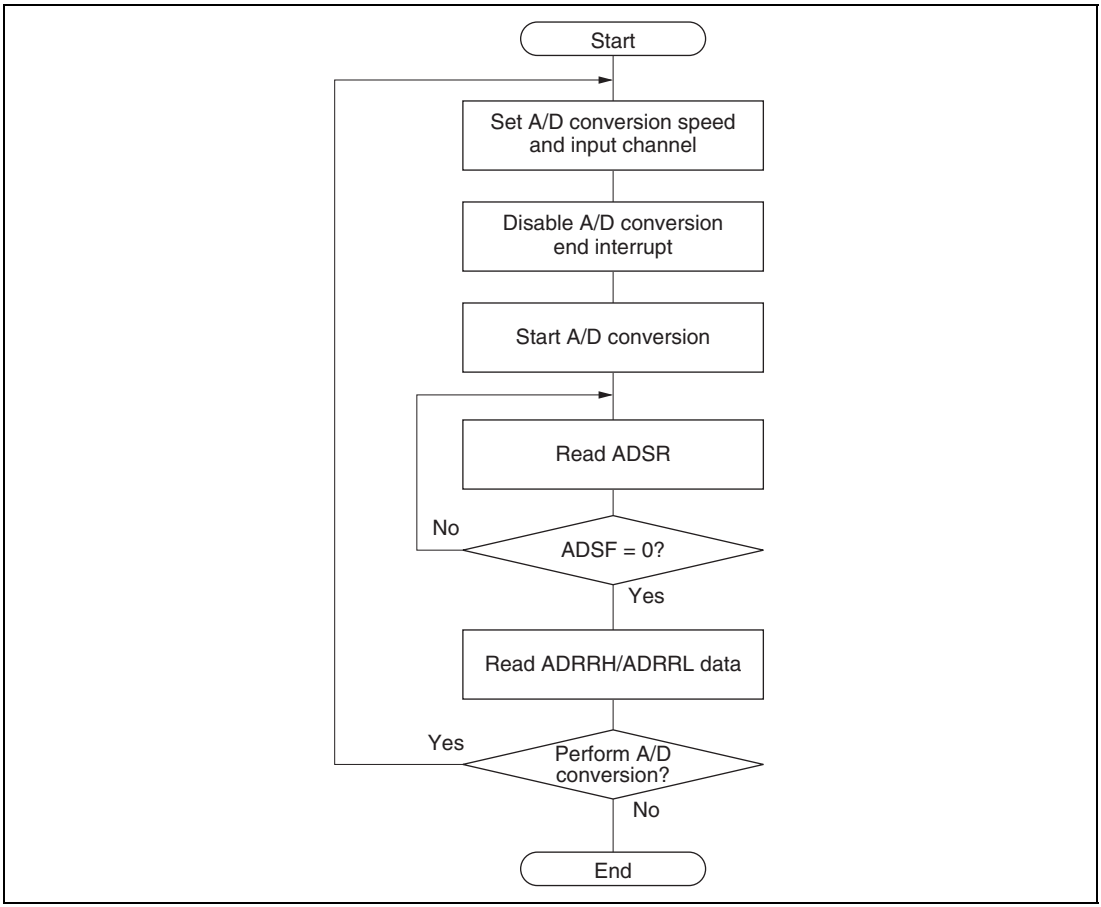



**Figure 12.4 Flow Chart of Procedure for Using A/D Converter (Polling by Software)**

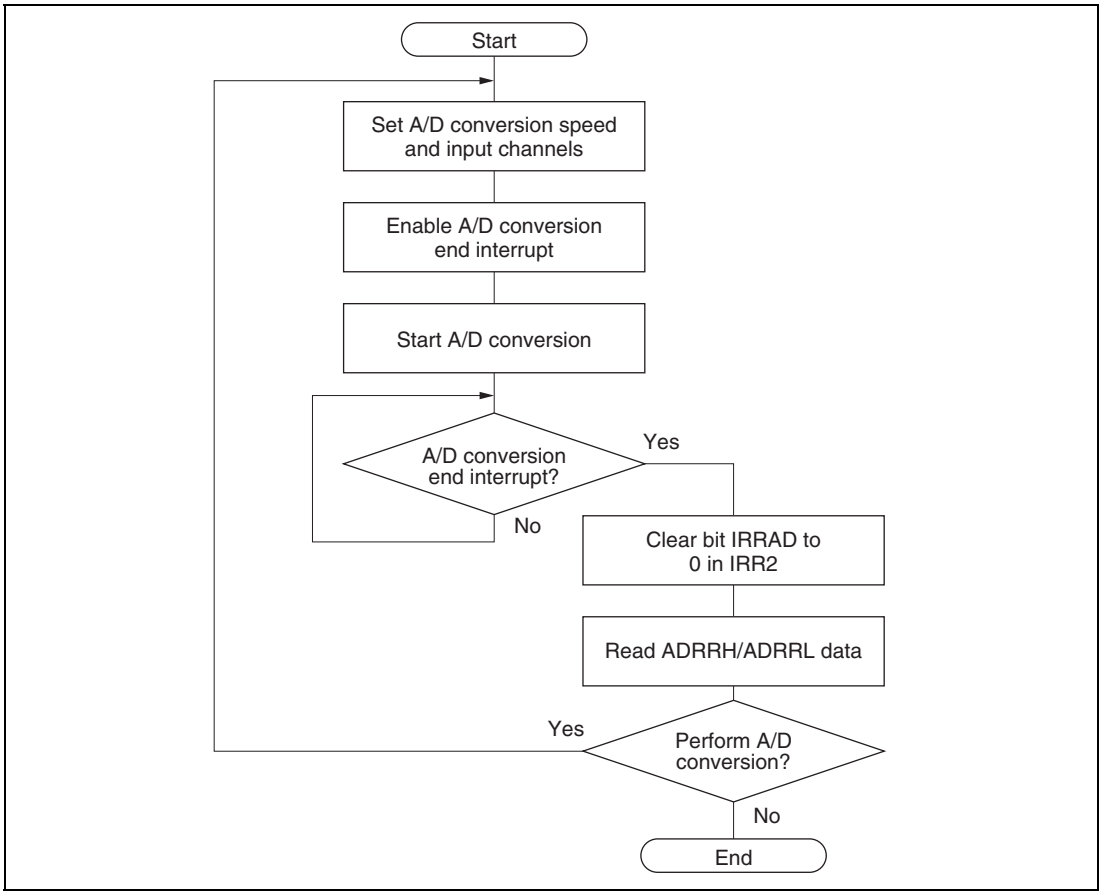



**Figure 12.5 Flow Chart of Procedure for Using A/D Converter (Interrupts Used)**

## 12.6 Application Notes

### 12.6.1 Application Notes

- Data in ADRRH and ADRRL should be read only when the A/D start flag (ADSF) in the A/D start register (ADSR) is cleared to 0.
- Changing the digital input signal at an adjacent pin during A/D conversion may adversely affect conversion accuracy.
- When A/D conversion is started after clearing module standby mode, wait for 10 $\phi$ clock cycles before starting.
- In active mode or sleep mode, analog power supply current ($AI_{STOP1}$) flows into the ladder resistance even when the A/D converter is not operating. Therefore, if the A/D converter is not used, it is recommended that $AV_{CC}$ be connected to the system power supply and the ADCKSTP(A/D converter module standby mode control) bit be cleared to 0 in clock stop register 1 (CKSTPR1).

### 12.6.2 Permissible Signal Source Impedance

This LSI's analog input is designed such that conversion precision is guaranteed for an input signal for which the signal source impedance is 10 kΩ or less. This specification is provided to enable the A/D converter's sample-and-hold circuit input capacitance to be charged within the sampling time; if the sensor output impedance exceeds 10 kΩ, charging may be insufficient and it may not be possible to guarantee A/D conversion precision. However, a large capacitance provided externally, the input load will essentially comprise only the internal input resistance of 10 kΩ, and the signal source impedance is ignored. However, as a low-pass filter effect is obtained in this case, it may not be possible to follow an analog signal with a large differential coefficient (e.g., 5 mV/μs or greater) (see figure 12.6). When converting a high-speed analog signal, a low-impedance buffer should be inserted.

### 12.6.3 Influences on Absolute Precision

Adding capacitance results in coupling with GND, and therefore noise in GND may adversely affect absolute precision. Be sure to make the connection to an electrically stable GND.

Care is also required to ensure that filter circuits do not interfere with digital signals or act as antennas on the mounting board.

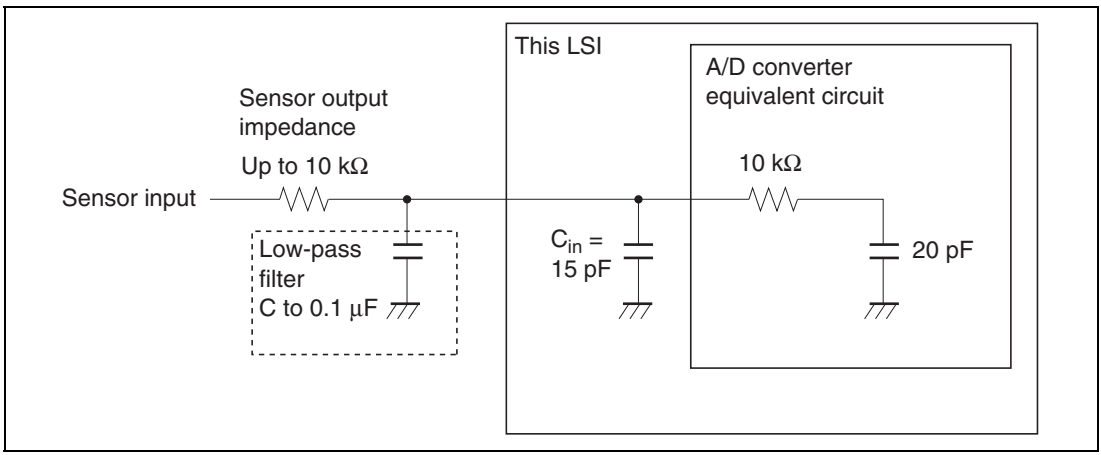



**Figure 12.6 Analog Input Circuit Example**

# Section 13 LCD Controller/Driver

## 13.1 Overview

This LSI has an on-chip segment type LCD control circuit, LCD driver, and power supply circuit, enabling it to directly drive an LCD panel.

### 13.1.1 Features

#### (1) Features

Features of the LCD controller/driver are given below.

- Display capacity

| Duty Cycle | Internal Driver |
|---|---|
| Static | 32 seg |
| 1/2 | 32 seg |
| 1/3 | 32 seg |
| 1/4 | 32 seg |

- LCD RAM capacity
  8 bits × 32 bytes (256 bits)
- Word access to LCD RAM
- All eight segment output pins can be used individually as port pins.
- Common output pins not used because of the duty cycle can be used for common double-buffering (parallel connection).
- Display possible in operating modes other than standby mode
- Choice of 11 frame frequencies
- Built-in power supply split-resistance, supplying LCD drive power
- Use of module standby mode enables this module to be placed in standby mode independently when not used.
- A or B waveform selectable by software

### 13.1.2 Block Diagram

Figure 13.1 shows a block diagram of the LCD controller/driver.

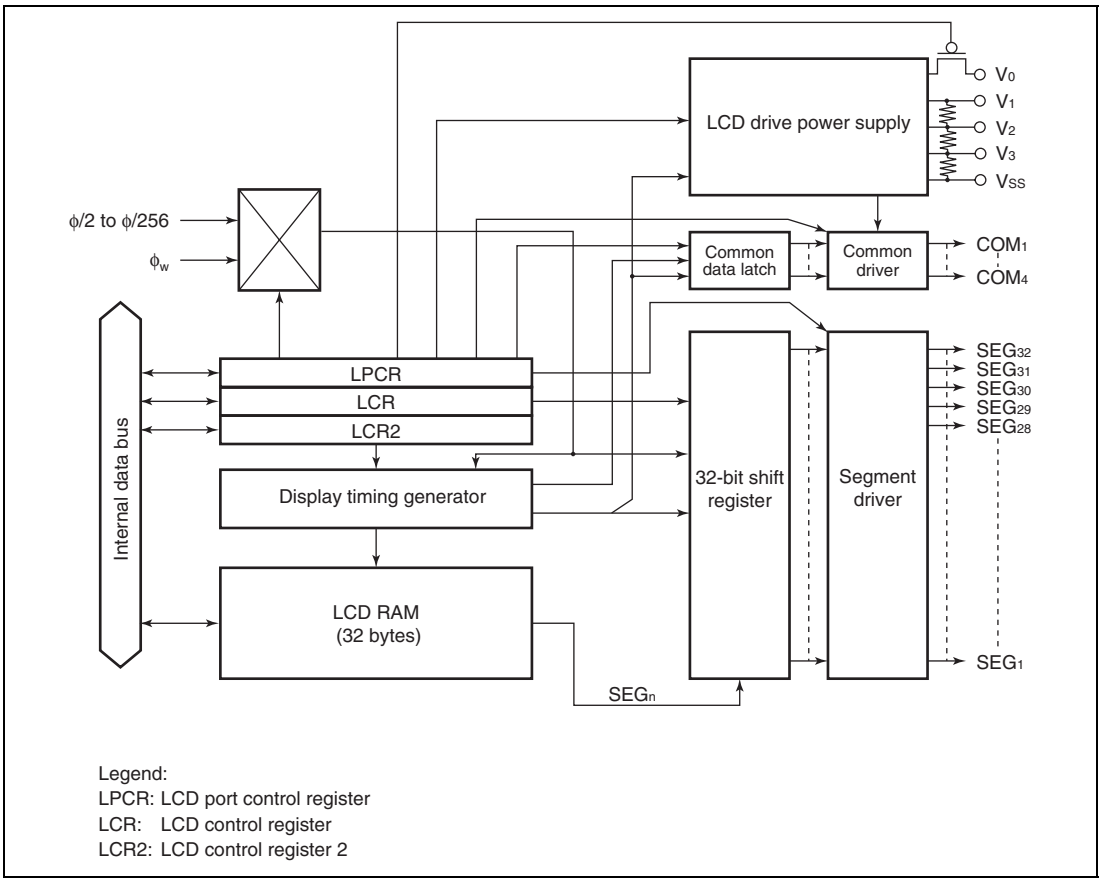



**Figure 13.1 Block Diagram of LCD Controller/Driver**

### 13.1.3 Pin Configuration

Table 13.1 shows the LCD controller/driver pin configuration.

**Table 13.1 Pin Configuration**

| Name | Abbr. | I/O | Function |
|---|---|---|---|
| Segment output pins | $SEG_{32}$ to $SEG_1$ | Output | LCD segment drive pins<br>All pins are multiplexed as port pins (setting programmable) |
| Common output pins | $COM_4$ to $COM_1$ | Output | LCD common drive pins<br>Pins can be used in parallel with static or 1/2 duty |
| LCD power supply pins | $V_0$, $V_1$, $V_2$, $V_3$ | — | Used when a bypass capacitor is connected externally, and when an external power supply circuit is used |

### 13.1.4 Register Configuration

Table 13.2 shows the register configuration of the LCD controller/driver.

**Table 13.2 LCD Controller/Driver Registers**

| Name | Abbr. | R/W | Initial Value | Address |
|---|---|---|---|---|
| LCD port control register | LPCR | R/W | H'00 | H'FFC0 |
| LCD control register | LCR | R/W | H'80 | H'FFC1 |
| LCD control register 2 | LCR2 | R/W | H'60 | H'FFC2 |
| LCD RAM | — | R/W | Undefined | H'F740, H'F75F |
| Clock stop register 2 | CKSTPR2 | R/W | H'FF | H'FFFB |

## 13.2 Register Descriptions

### 13.2.1 LCD Port Control Register (LPCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DTS1 | DTS0 | CMX | — | SGS3 | SGS2 | SGS1 | SGS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | W | R/W | R/W | R/W | R/W |

LPCR is an 8-bit read/write register which selects the duty cycle and LCD driver pin functions.

LPCR is initialized to H'00 upon reset.

**Bits 7 to 5:** Duty cycle select 1 and 0 (DTS1, DTS0), common function select (CMX)

The combination of DTS1 and DTS0 selects static, 1/2, 1/3, or 1/4 duty. CMX specifies whether or not the same waveform is to be output from multiple pins to increase the common drive power when not all common pins are used because of the duty setting.

| Bit 7 DTS1 | Bit 6 DTS0 | Bit 5 CMX | Duty Cycle | Common Drivers | Notes |
|---|---|---|---|---|---|
| 0 | 0 | 0 | Static | $COM_1$ (initial value) | Do not use $COM_4$, $COM_3$, and $COM_2$. |
| | | 1 | | $COM_4$ to $COM_1$ | $COM_4$, $COM_3$, and $COM_2$ output the same waveform as $COM_1$. |
| 0 | 1 | 0 | 1/2 duty | $COM_2$ to $COM_1$ | Do not use $COM_4$ and $COM_3$. |
| | | 1 | | $COM_4$ to $COM_1$ | $COM_4$ outputs the same waveform as $COM_3$, and $COM_2$ outputs the same waveform as $COM_1$. |
| 1 | 0 | 0 | 1/3 duty | $COM_3$ to $COM_1$ | Do not use $COM_4$. |
| | | 1 | | $COM_4$ to $COM_1$ | Do not use $COM_4$. |
| 1 | 1 | 0 | 1/4 duty | $COM_4$ to $COM_1$ | — |
| | | 1 | | | |

**Bit 4:** Reserved

Bit 4 is reserved. It can only be written with 0.

**Bits 3 to 0:** Segment driver select 3 to 0 (SGS3 to SGS0)

Bits 3 to 0 select the segment drivers to be used.

| | | | | Function of Pins $SEG_{32}$ to $SEG_1$ | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit 3 SGS3** | **Bit 2 SGS2** | **Bit 1 SGS1** | **Bit 0 SGS0** | **$SEG_{32}$ to $SEG_{25}$** | **$SEG_{24}$ to $SEG_{17}$** | **$SEG_{16}$ to $SEG_9$** | **$SEG_8$ to $SEG_1$** | **Notes** |
| 0 | 0 | 0 | 0 | Port | Port | Port | Port | (initial value) |
| 0 | 0 | 0 | 1 | Port | Port | Port | Port | |
| 0 | 0 | 1 | * | SEG | Port | Port | Port | |
| 0 | 1 | 0 | * | SEG | SEG | Port | Port | |
| 0 | 1 | 1 | * | SEG | SEG | SEG | Port | |
| 1 | * | * | * | SEG | SEG | SEG | SEG | |

*: Don't care

### 13.2.2 LCD Control Register (LCR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PSW | ACT | DISP | CKS3 | CKS2 | CKS1 | CKS0 |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

LCR is an 8-bit read/write register which performs LCD drive power supply on/off control and display data control, and selects the frame frequency.

LCR is initialized to H'80 upon reset.

**Bit 7:** Reserved bit

Bit 7 is reserved; it is always read as 1 and cannot be modified.

**Bit 6:** LCD drive power supply on/off control (PSW)

Bit 6 can be used to turn the LCD drive power supply off when LCD display is not required in a power-down mode, or when an external power supply is used. When the ACT bit is cleared to 0, or in standby mode, the LCD drive power supply is turned off regardless of the setting of this bit.

| Bit 6<br>PSW | Description | |
|---|---|---|
| 0 | LCD drive power supply off | (initial value) |
| 1 | LCD drive power supply on | |

**Bit 5:** Display function activate (ACT)

Bit 5 specifies whether or not the LCD controller/driver is used. Clearing this bit to 0 halts operation of the LCD controller/driver. The LCD drive power supply is also turned off, regardless of the setting of the PSW bit. However, register contents are retained.

| Bit 5<br>ACT | Description | |
|---|---|---|
| 0 | LCD controller/driver operation halted | (initial value) |
| 1 | LCD controller/driver operates | |

**Bit 4:** Display data control (DISP)

Bit 4 specifies whether the LCD RAM contents are displayed or blank data is displayed regardless of the LCD RAM contents.

| Bit 4<br>DISP | Description | |
|---|---|---|
| 0 | Blank data is displayed | (initial value) |
| 1 | LCD RAM data is display | |

**Bits 3 to 0:** Frame frequency select 3 to 0 (CKS3 to CKS0)

Bits 3 to 0 select the operating clock and the frame frequency. In subactive mode, watch mode, and subsleep mode, the system clock (ϕ) is halted, and therefore display operations are not performed if one of the clocks from ϕ/2 to ϕ/256 is selected. If LCD display is required in these modes, ϕw, ϕw/2, or ϕw/4 must be selected as the operating clock.

| Bit 3<br>CKS3 | Bit 2<br>CKS2 | Bit 1<br>CKS1 | Bit 0<br>CKS0 | Operating Clock | Frame Frequency*2 | |
|---|---|---|---|---|---|---|
| | | | | | ϕ = 2 MHz | ϕ = 250 kHz*1 |
| 0 | * | 0 | 0 | ϕw | 128 Hz*3 (initial value) | |
| 0 | * | 0 | 1 | ϕw/2 | 64 Hz*3 | |
| 0 | * | 1 | * | ϕw/4 | 32 Hz*3 | |
| 1 | 0 | 0 | 0 | ϕ/2 | — | 244 Hz |
| 1 | 0 | 0 | 1 | ϕ/4 | 977 Hz | 122 Hz |
| 1 | 0 | 1 | 0 | ϕ/8 | 488 Hz | 61 Hz |
| 1 | 0 | 1 | 1 | ϕ/16 | 244 Hz | 30.5 Hz |
| 1 | 1 | 0 | 0 | ϕ/32 | 122 Hz | — |
| 1 | 1 | 0 | 1 | ϕ/64 | 61 Hz | — |
| 1 | 1 | 1 | 0 | ϕ/128 | 30.5 Hz | — |
| 1 | 1 | 1 | 1 | ϕ/256 | — | — |

*: Don't care

Notes: 1. This is the frame frequency in active (medium-speed, ϕosc/16) mode when ϕ = 2 MHz.
2. When 1/3 duty is selected, the frame frequency is 4/3 times the value shown.
3. This is the frame frequency when ϕw = 32.768 kHz.

### 13.2.3 LCD Control Register 2 (LCR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LCDAB | — | — | — | CDS3 | CDS2 | CDS1 | CDS0 |
| Initial value | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | — | — | R/W | R/W | R/W | R/W | R/W |

LCR2 is an 8-bit read/write register which controls switching between the A waveform and B waveform, and selects the duty cycle of the charge/discharge pulses which control disconnection of the power supply split-resistance from the power supply circuit.

LCR2 is initialized to H'60 upon reset.

**Bit 7:** A waveform/B waveform switching control (LCDAB)

Bit 7 specifies whether the A waveform or B waveform is used as the LCD drive waveform.

| Bit 7<br>LCDAB | Description | |
|---|---|---|
| 0 | Drive using A waveform | (initial value) |
| 1 | Drive using B waveform | |

**Bits 6 and 5:** Reserved bits

Bits 6 and 5 are reserved; they are always read as 1 and cannot be modified.

**Bit 4:** Reserved bit

Bit 4 is reserved; it is always read as 0 and must not be written with 1.

**Bits 3 to 0:** Charge/discharge pulse duty cycle select (CDS3 to CDS0)

| Bit 3 CDS3 | Bit 2 CDS2 | Bit 1 CDS1 | Bit 0 CDS0 | Duty Cycle | Notes | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | Fixed high | (initial value) |
| 0 | 0 | 0 | 1 | 1/8 | | |
| 0 | 0 | 1 | 0 | 2/8 | | |
| 0 | 0 | 1 | 1 | 3/8 | | |
| 0 | 1 | 0 | 0 | 4/8 | | |
| 0 | 1 | 0 | 1 | 5/8 | | |
| 0 | 1 | 1 | 0 | 6/8 | | |
| 0 | 1 | 1 | 1 | 0 | Fixed low | |
| 1 | 0 | * | * | 1/16 | | |
| 1 | 1 | * | * | 1/32 | | |

*: Don't care

Bits 3 to 0 select the duty cycle while the power supply split-resistance is connected to the power supply circuit.

When a 0 duty cycle is selected, the power supply split-resistance is permanently disconnected from the power supply circuit, so power should be supplied to pins $V_1$, $V_2$, and $V_3$ by an external circuit.

Figure 13.2 shows the waveform of the charge/discharge pulses. The duty cycle is Tc/Tw.

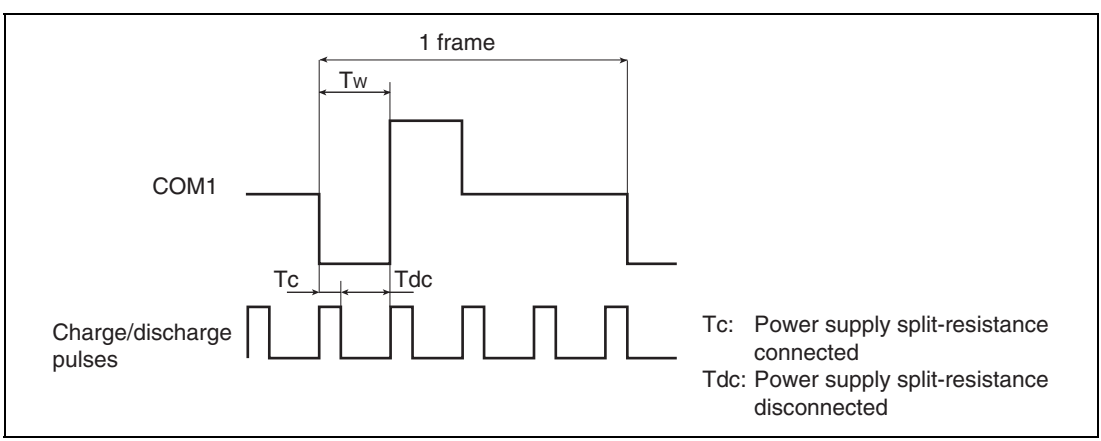



**Figure 13.2 Example of A Waveform with 1/2 Duty and 1/2 Bias**

### 13.2.4 Clock Stop Register 2 (CKSTPR2)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | AECKSTP | WDCKSTP | PWCKSTP | LDCKSTP |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | R/W | R/W | R/W | R/W |

CKSTPR2 is an 8-bit read/write register that performs module standby mode control for peripheral modules. Only the bit relating to the LCD controller/driver is described here. For details of the other bits, see the sections on the relevant modules.

**Bit 0:** LCD controller/driver module standby mode control (LDCKSTP)

Bit 0 controls setting and clearing of module standby mode for the LCD controller/driver.

| Bit 0<br>LDCKSTP | Description | |
|---|---|---|
| 0 | LCD controller/driver is set to module standby mode | |
| 1 | LCD controller/driver module standby mode is cleared | (initial value) |

## 13.3 Operation

### 13.3.1 Settings Up to LCD Display

To perform LCD display, the hardware and software related items described below must first be determined.

#### (1) Hardware Settings

a. Using 1/2 duty

When 1/2 duty is used, interconnect pins $V_2$ and $V_3$ as shown in figure 13.3.

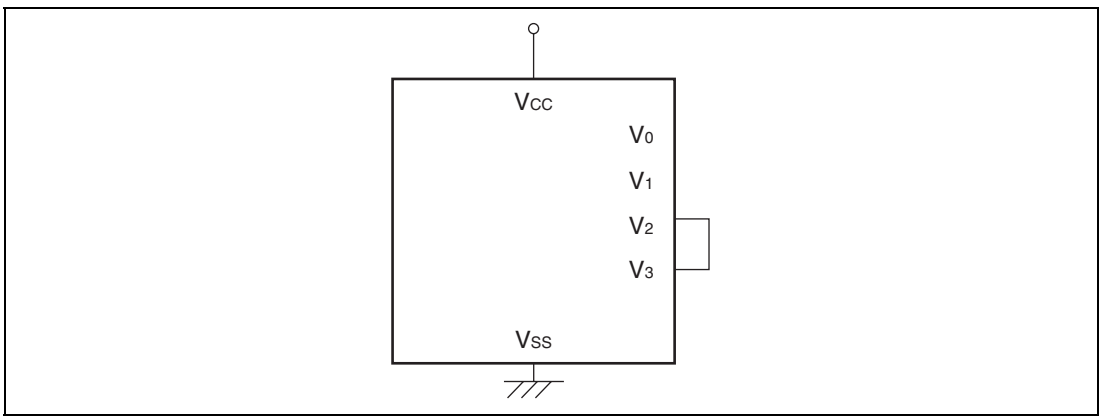



**Figure 13.3 Handling of LCD Drive Power Supply when Using 1/2 Duty**

b. Large-panel display

As the impedance of the built-in power supply split-resistance is large, it may not be suitable for driving a large panel. If the display lacks sharpness when using a large panel, refer to section 13.3.6, Boosting the LCD Drive Power Supply. When static or 1/2 duty is selected, the common output drive capability can be increased. Set CMX to 1 when selecting the duty cycle. In this mode, with a static duty cycle pins $COM_4$ to $COM_1$ output the same waveform, and with 1/2 duty the $COM_1$ waveform is output from pins $COM_2$ and $COM_1$, and the $COM_2$ waveform is output from pins $COM_4$ and $COM_3$.

c. Luminance adjustment function ($V_0$ pin)

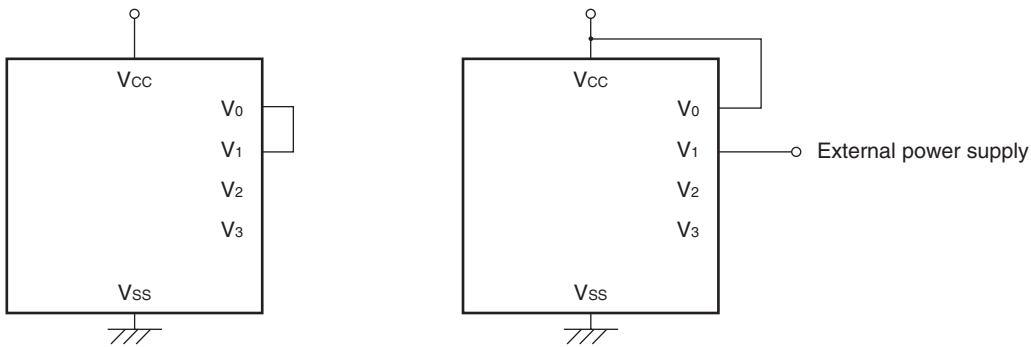
Connecting a resistance between the $V_0$ and $V_1$ pins enables the luminance to be adjusted. For details, see section 13.3.3, Luminance Adjustment Function ($V_0$ Pin).

d. LCD drive power supply setting

With this LSI, there are two ways of providing LCD power: by using the on-chip power supply circuit, or by using an external circuit.

When the on-chip power supply circuit is used for the LCD drive power supply, the $V_0$ and $V_1$ pins should be interconnected externally, as shown in figure 13.4 (a).

When an external power supply circuit is used for the LCD drive power supply, connect the external power supply to the $V_1$ pin, and short the $V_0$ pin to $V_{CC}$ externally, as shown in figure 13.4 (b).




(a) Using on-chip power supply circuit (b) Using external power supply circuit

**Figure 13.4 Examples of LCD Power Supply Pin Connections**

e. Low-power-consumption LCD drive system

Use of a low-power-consumption LCD drive system enables the power consumption required for LCD drive to be optimized. For details, see section 13.3.4, Low-Power-Consumption LCD Drive System.

### (2) Software settings

a. Duty selection

Any of four duty cycles—static, 1/2 duty, 1/3 duty, or 1/4 duty—can be selected with bits DTS1 and DTS0.

b. Segment selection

The segment drivers to be used can be selected with bits $SGS_3$ to $SGS_0$.

c. Frame frequency selection

The frame frequency can be selected by setting bits $CKS_3$ to $CKS_0$. The frame frequency should be selected in accordance with the LCD panel specification. For the clock selection method in watch mode, subactive mode, and subsleep mode, see section 13.3.5, Operation in Power-Down Modes.

d. A or B waveform selection

Either the A or B waveform can be selected as the LCD waveform to be used by means of LCDAB.

### 13.3.2 Relationship between LCD RAM and Display

The relationship between the LCD RAM and the display segments differs according to the duty cycle. LCD RAM maps for the different duty cycles are shown in figures 13.5 to 13.8.

After setting the registers required for display, data is written to the part corresponding to the duty using the same kind of instruction as for ordinary RAM, and display is started automatically when turned on. Word- or byte-access instructions can be used for RAM setting.

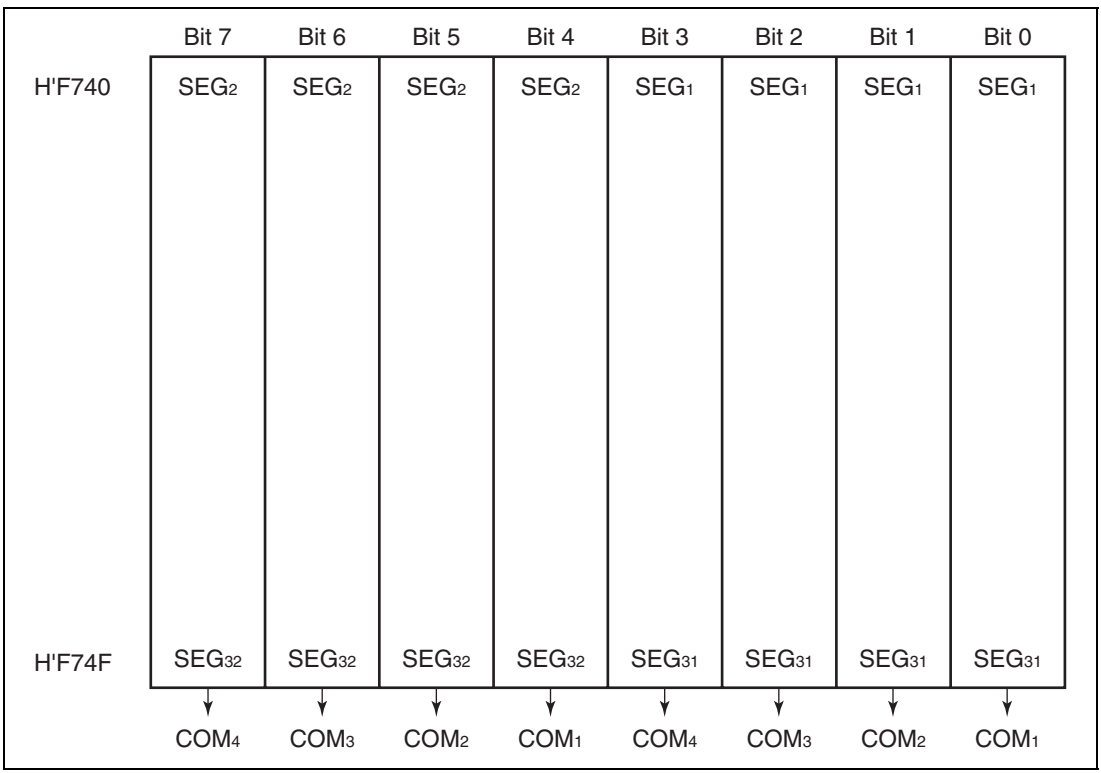



**Figure 13.5 LCD RAM Map (1/4 Duty)**

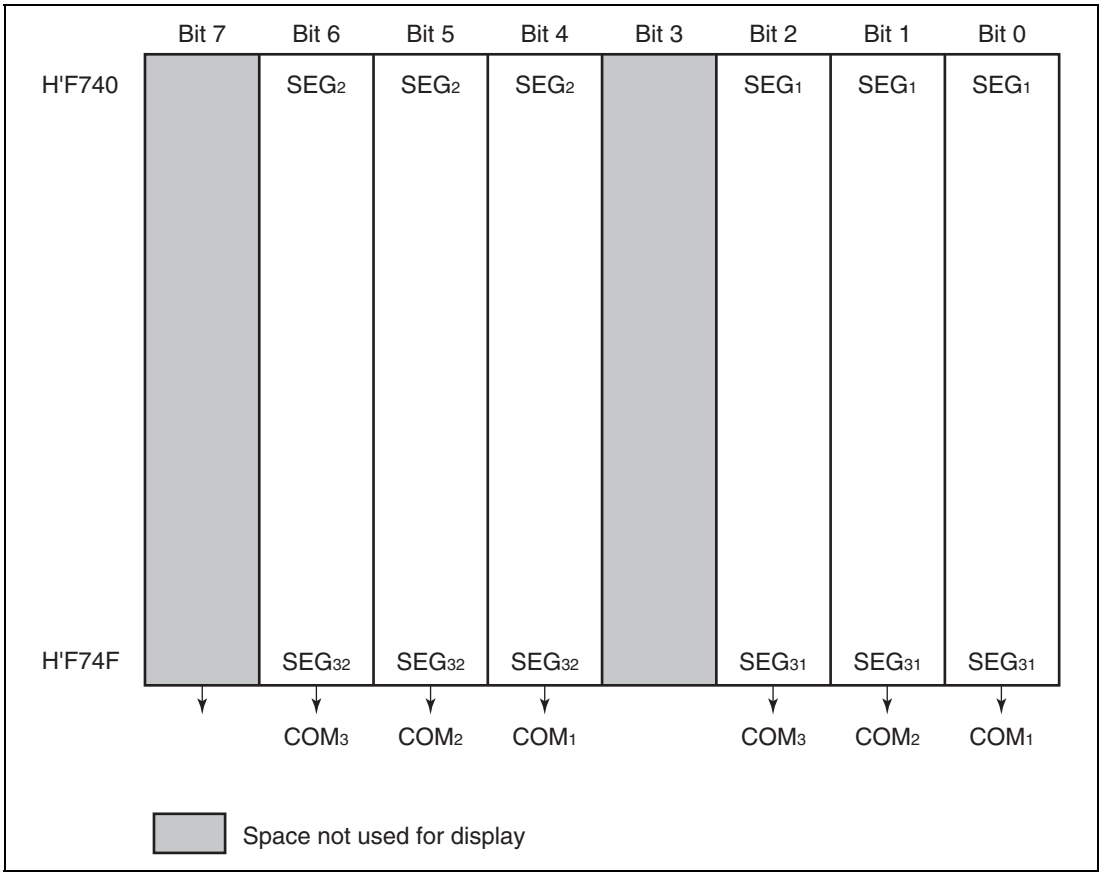



**Figure 13.6 LCD RAM Map (1/3 Duty)**

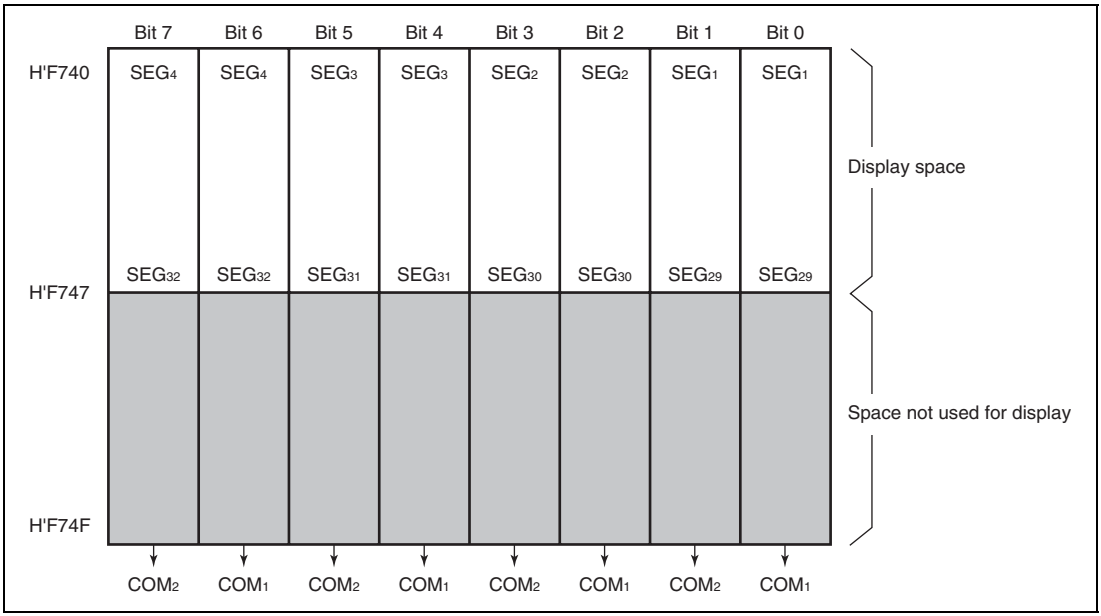



**Figure 13.7 LCD RAM Map (1/2 Duty)**

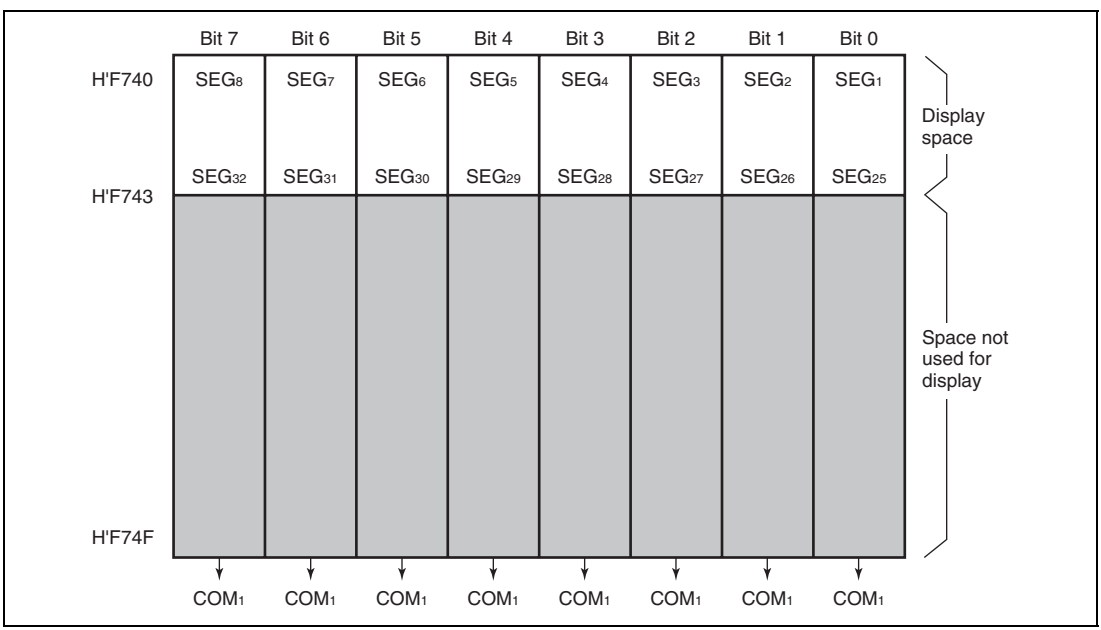



**Figure 13.8 LCD RAM Map (Static Mode)**

### 13.3.3 Luminance Adjustment Function ($V_0$ Pin)

Figure 13.9 shows a detailed block diagram of the LCD drive power supply unit.

The voltage output to the $V_0$ pin is $V_{CC}$. When either of these voltages is used directly as the LCD drive power supply, the $V_0$ and $V_1$ pins should be shorted. Also, connecting a variable resistance, R, between the $V_0$ and $V_1$ pins makes it possible to adjust the voltage applied to the $V_1$ pin, and so to provide luminance adjustment for the LCD panel.

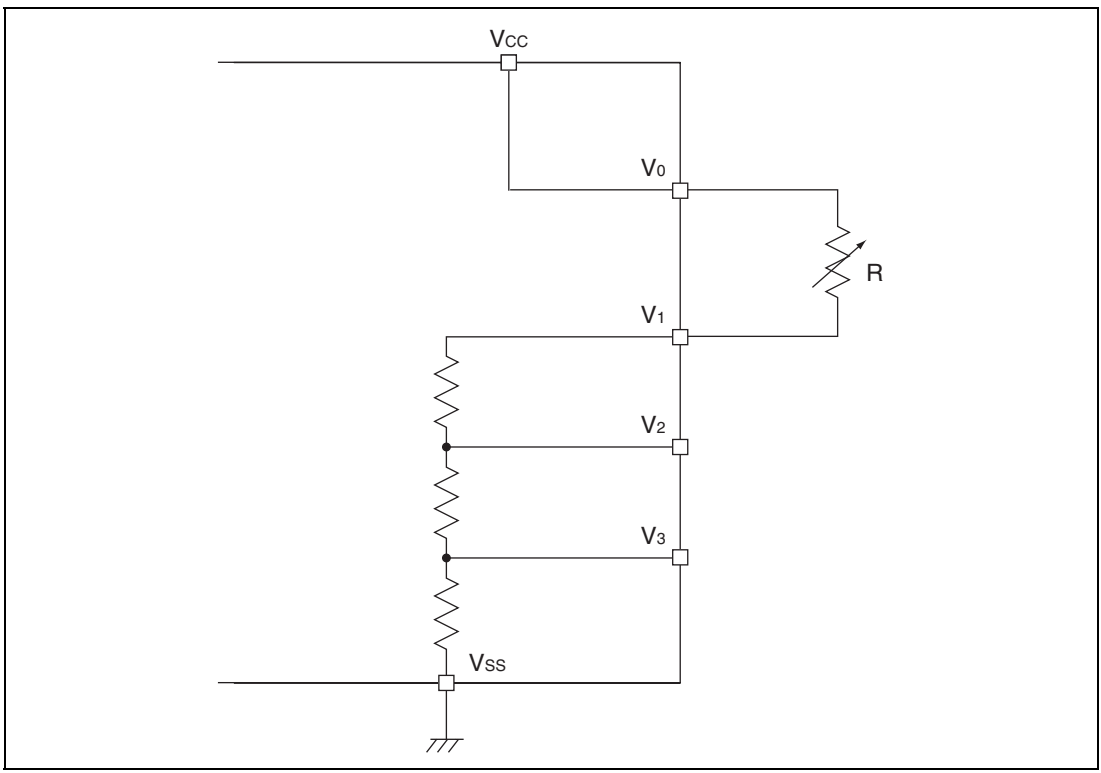



**Figure 13.9 LCD Drive Power Supply Unit**

### 13.3.4 Low-Power-Consumption LCD Drive System

The use of the built-in split-resistance is normally the easiest method for implementing the LCD power supply circuit, but since the built-in resistance is fixed, a certain direct current flows constantly from the built-in resistance's $V_{CC}$ to $V_{SS}$. As this current does not depend on the current dissipation of the LCD panel, if an LCD panel with a small current dissipation is used, a wasteful amount of power will be consumed. This LSI is equipped with a function to minimize this waste of power. Use of this function makes it possible to achieve the optimum power supply circuit for the LCD panel's current dissipation.

#### (1) Principles

a. Capacitors are connected as external circuits to LCD power supply pins V1, V2, and V3, as shown in figure 13.10.
b. The capacitors connected to V1, V2, and V3 are repeatedly charged and discharged in the cycle shown in figure 13.10, maintaining the potentials.
c. At this time, the charged potential is a potential corresponding to the V1, V2, and V3 pins, respectively. (For example, with 1/3 bias drive, the charge for V2 is 2/3 that of V1, and that for V3 is 1/3 that of V1.)
d. Power is supplied to the LCD panel by means of the charges accumulated in these capacitors.
e. The capacitances and charging/discharging periods of these capacitors are therefore determined by the current dissipation of the LCD panel.
f. The charging and discharging periods can be selected by software.

#### (2) Example of Operation (with 1/3 bias drive)

a. During charging period Tc in the figure, the potential is divided among pins V1, V2, and V3 by the built-in split-resistance (the potential of V2 being 2/3 that of V1, and that of V3 being 1/3 that of V1), as shown in figure 13.10, and external capacitors C1, C2, and C3 are charged. The LCD panel is continues to be driven during this time.
b. In the following discharging period, Tdc, charging is halted and the charge accumulated in each capacitor is discharged, driving the LCD panel.
c. At this time, a slight voltage drop occurs due to the discharging; optimum values must be selected for the charging period and the capacitor capacitances to ensure that this does not affect the driving of the LCD panel.
d. In this way, the capacitors connected to V1, V2, and V3 are repeatedly charged and discharged in the cycle shown in figure 13.10, maintaining the potentials and continuously driving the LCD panel.

e. As can be seen from the above description, the capacitances and charging/discharging periods of the capacitors are determined by the current dissipation of the LCD panel used. The charging/discharging periods can be selected with bits CDS3 to CDS0.

f. The actual capacitor capacitances and charging/discharging periods must be determined experimentally in accordance with the current dissipation requirements of the LCD panel. An optimum current value can be selected, in contrast to the case in which a direct current flows constantly in the built-in split-resistance.

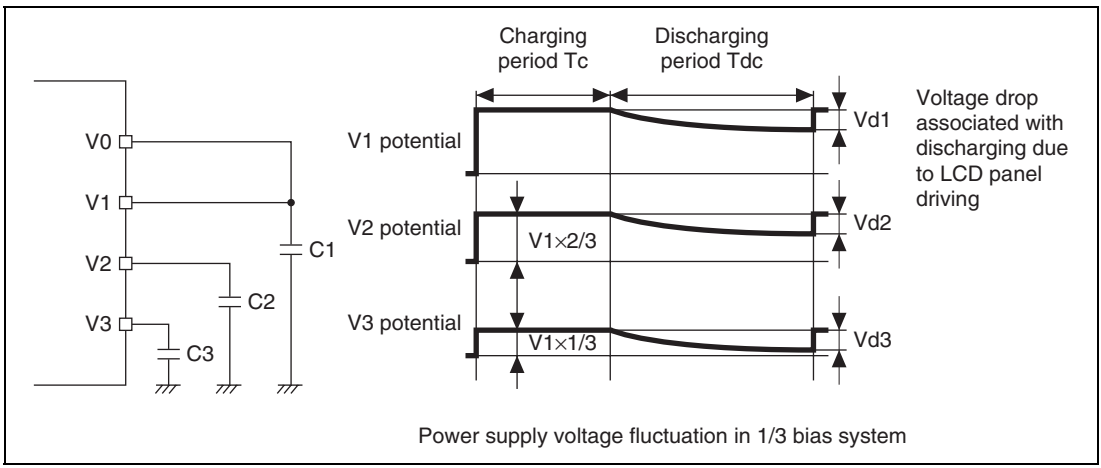



**Figure 13.10 Example of Low-Power-Consumption LCD Drive Operation**

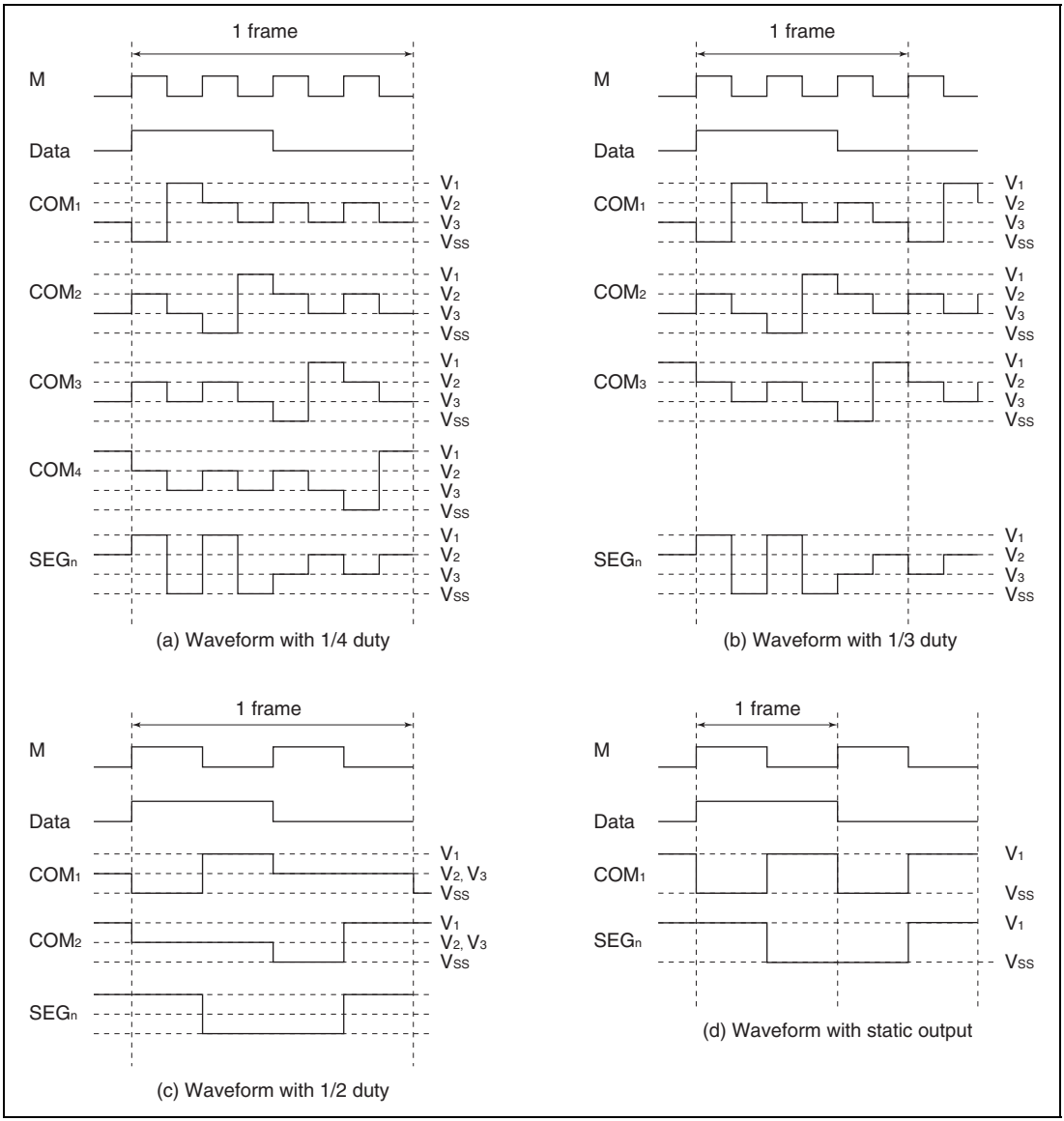



**Figure 13.11 Output Waveforms for Each Duty Cycle (A Waveform)**

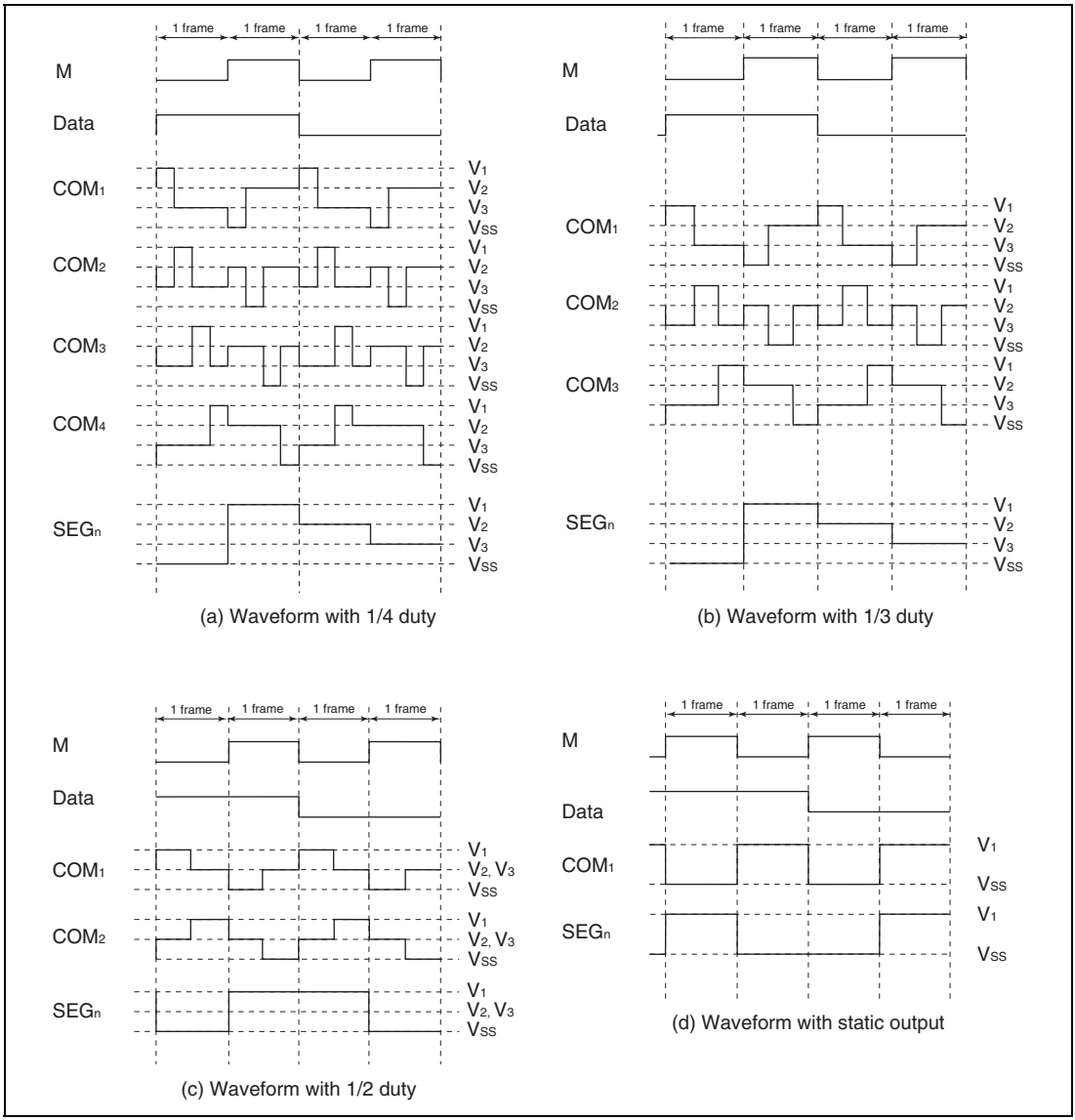



**Figure 13.12 Output Waveforms for Each Duty Cycle (B Waveform)**

**Table 13.3 Output Levels**

| Data | | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|
| **M** | | **0** | **1** | **0** | **1** |
| Static | Common output | $V_1$ | $V_{SS}$ | $V_1$ | $V_{SS}$ |
| | Segment output | $V_1$ | $V_{SS}$ | $V_{SS}$ | $V_1$ |
| 1/2 duty | Common output | $V_2$, $V_3$ | $V_2$, $V_3$ | $V_1$ | $V_{SS}$ |
| | Segment output | $V_1$ | $V_{SS}$ | $V_{SS}$ | $V_1$ |
| 1/3 duty | Common output | $V_3$ | $V_2$ | $V_1$ | $V_{SS}$ |
| | Segment output | $V_2$ | $V_3$ | $V_{SS}$ | $V_1$ |
| 1/4 duty | Common output | $V_3$ | $V_2$ | $V_1$ | $V_{SS}$ |
| | Segment output | $V_2$ | $V_3$ | $V_{SS}$ | $V_1$ |

### 13.3.5 Operation in Power-Down Modes

In this LSI, the LCD controller/driver can be operated even in the power-down modes. The operating state of the LCD controller/driver in the power-down modes is summarized in table 13.4.

In subactive mode, watch mode, and subsleep mode, the system clock oscillator stops, and therefore, unless ϕw, ϕw/2, or ϕw/4 has been selected by bits CKS3 to CKS0, the clock will not be supplied and display will halt. Since there is a possibility that a direct current will be applied to the LCD panel in this case, it is essential to ensure that ϕw, ϕw/2, or ϕw/4 is selected. In active (medium-speed) mode, the system clock is switched, and therefore CKS3 to CKS0 must be modified to ensure that the frame frequency does not change.

**Table 13.4 Power-Down Modes and Display Operation**

| Mode | | Reset | Active | Sleep | Watch | Sub-active | Sub-sleep | Standby | Module Standby |
|---|---|---|---|---|---|---|---|---|---|
| Clock | ϕ | Runs | Runs | Runs | Stops | Stops | Stops | Stops | Stops*4 |
| | ϕw | Runs | Runs | Runs | Runs | Runs | Runs | Stops*1 | Stops*4 |
| Display operation | ACT = "0" | Stops | Stops | Stops | Stops | Stops | Stops | Stops*2 | Stops |
| | ACT = "1" | Stops | Functions | Functions | Functions *3 | Functions *3 | Functions *3 | Stops*2 | Stops |

Notes: 1. The subclock oscillator does not stop, but clock supply is halted.
2. The LCD drive power supply is turned off regardless of the setting of the PSW bit.
3. Display operation is performed only if ϕw, ϕw/2, or ϕw/4 is selected as the operating clock.
4. The clock supplied to the LCD stops.

### 13.3.6 Boosting the LCD Drive Power Supply

When a large panel is driven, the on-chip power supply capacity may be insufficient. If the power supply capacity is insufficient when $V_{CC}$ is used as the power supply, the power supply impedance must be reduced. This can be done by connecting bypass capacitors of around 0.1 to 0.3 μF to pins $V_1$ to $V_3$, as shown in figure 13.13, or by adding a split-resistance externally.

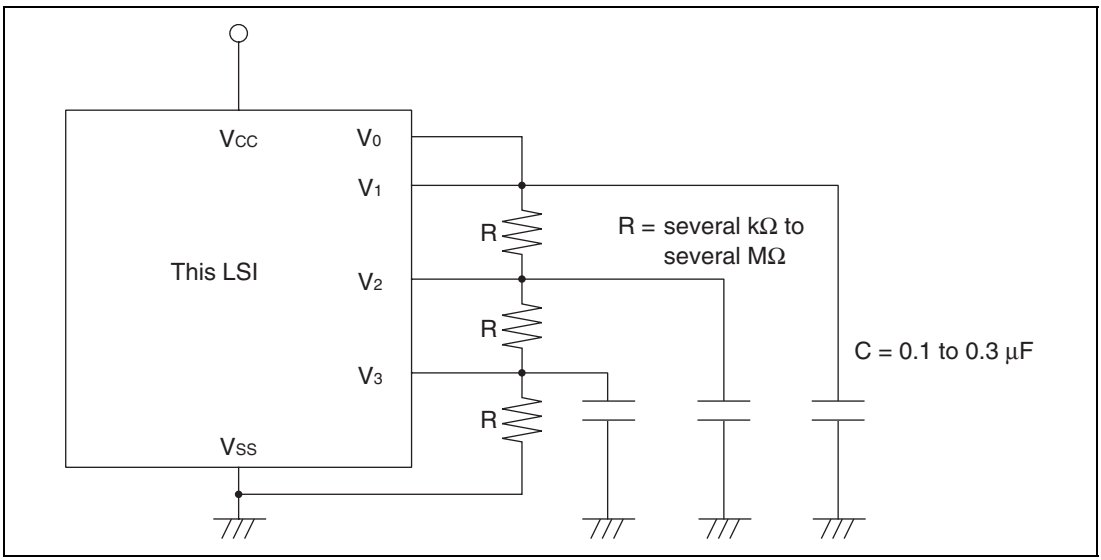



**Figure 13.13 Connection of External Split-Resistance**

# Section 14 Power Supply Circuit

## 14.1 Overview

This LSI incorporates an internal power supply step-down circuit. Use of this circuit enables the internal power supply to be fixed at a constant level of approximately 3.0 V to 3.2 V, independently of the voltage of the power supply connected to the external $V_{cc}$ pin. As a result, the current consumed when an external power supply is used at 3.0 V or above can be held down to virtually the same low level as when used at approximately 3.0 V. If the external power supply is 3.0 V or below, the internal voltage will be practically the same as the external voltage. It is, of course, also possible to use the same level of external power supply voltage and internal power supply voltage without using the internal power supply step-down circuit.

## 14.2 When Using Internal Power Supply Step-Down Circuit

Connect the external power supply to the $V_{cc}$ pin, and connect a capacitance of approximately 0.33 µF between $CV_{cc}$ and $V_{ss}$, as shown in figure 14.1. The internal step-down circuit is made effective simply by adding this external circuit. In the external circuit interface, the external power supply voltage connected to $V_{cc}$ and the GND potential connected to $V_{ss}$ are the reference levels. For example, for port input/output levels, the $V_{cc}$ level is the reference for the high level, and the $V_{ss}$ level is that for the low level. The LCD power supply and A/D converter analog power supply are not affected by the internal step-down current.

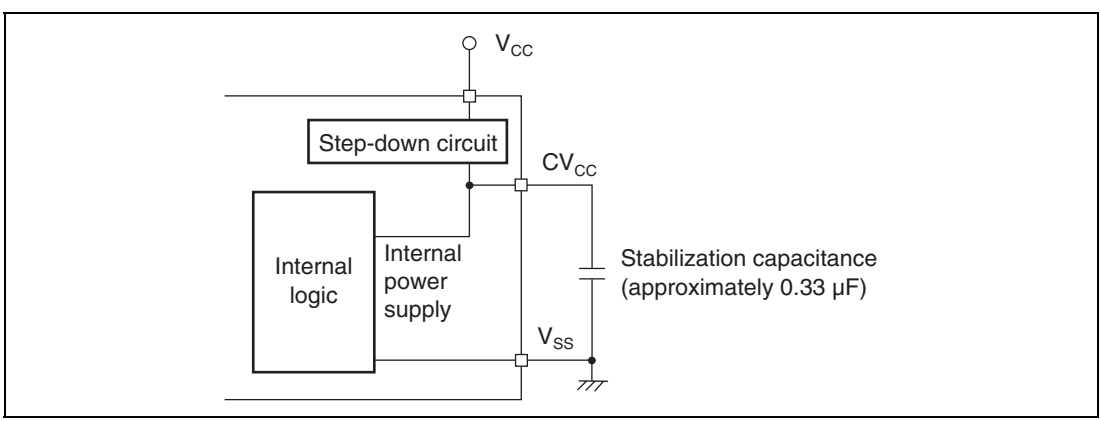



**Figure 14.1 Power Supply Connection when Internal Step-Down Circuit is Used**

## 14.3 When Not Using Internal Power Supply Step-Down Circuit

When the internal power supply step-down circuit is not used, connect the external power supply to the $CV_{cc}$ pin and $V_{cc}$ pin, as shown in figure 14.2. The external power supply is then input directly to the internal power supply. The permissible range for the power supply voltage is 2.7 V to 3.6 V. Normally, however, the internal power supply step-down circuit should be used. Operation cannot be guaranteed if a voltage outside this range is input.

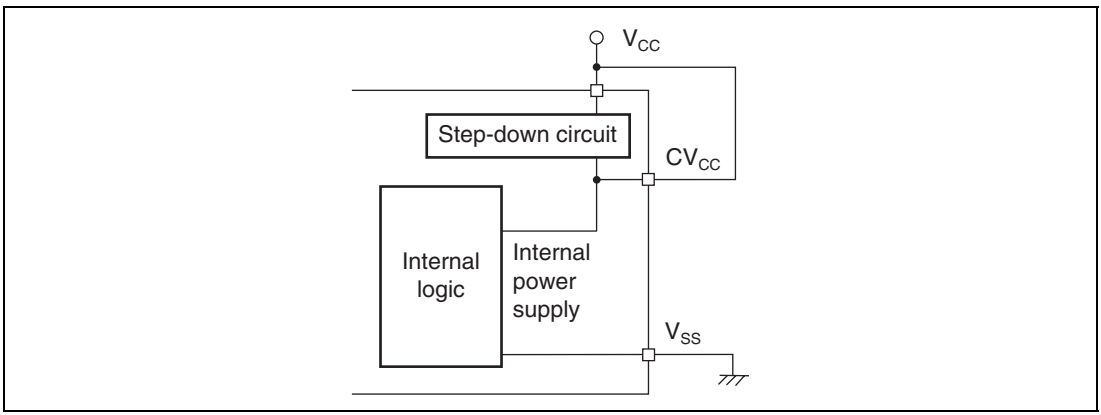



**Figure 14.2 Power Supply Connection when Internal Step-Down Circuit is Not Used**

# Section 15 List of Registers

The register list gives information on the on-chip I/O register addresses, how the register bits are configured, and the register states in each operating mode. The information is given as shown below.

1. Register addresses (address order)

- Registers are listed from the lower allocation addresses.
- Registers are classified by functional modules.
- The data bus width is indicated.
- The number of access states is indicated.

2. Register bits

- Bit configurations of the registers are described in the same order as the register addresses.
- Reserved bits are indicated by — in the bit name column.
- When registers consist of 16 bits, bits are described from the MSB side.

3. Register states in each operating mode

- Register states are described in the same order as the register addresses.
- The register states described here are for the basic operating modes. If there is a specific reset for an on-chip peripheral module, refer to the section on that on-chip peripheral module.

## 15.1 Register Addresses (Address Order)

The data bus width indicates the numbers of bits by which the register is accessed.

The number of access states indicates the number of states based on the specified reference clock.

| Register Name | Abbre-viation | Bit No | Address | Module Name | Data Bus Width | Access State |
|---|---|---|---|---|---|---|
| Flash memory control register 1 | FLMCR1 | 8 | H'F020 | ROM | 8 | 2 |
| Flash memory control register 2 | FLMCR2 | 8 | H'F021 | ROM | 8 | 2 |
| Flash memory power control register | FLPWCR | 8 | H'F022 | ROM | 8 | 2 |
| Erase block register | EBR | 8 | H'F023 | ROM | 8 | 2 |
| Flash memory enable register | FENR | 8 | H'F02B | ROM | 8 | 2 |
| Wakeup edge select register | WEGR | 8 | H'FF90 | Interrupts | 8 | 2 |
| Serial port control register | SPCR | 8 | H'FF91 | SCI3 | 8 | 2 |
| Subclock output select register | CWOSR | 8 | H'FF92 | Timer A | 8 | 2 |
| Event counter control/status register | ECCSR | 8 | H'FF95 | AEC*1 | 8 | 2 |
| Event counter H | ECH | 8 | H'FF96 | AEC*1 | 8 | 2 |
| Event counter L | ECL | 8 | H'FF97 | AEC*1 | 8 | 2 |
| Serial mode register 31 | SMR31 | 8 | H'FF98 | SCI31 | 8 | 3 |
| Bit rate register 31 | BRR31 | 8 | H'FF99 | SCI31 | 8 | 3 |
| Serial control register 31 | SCR31 | 8 | H'FF9A | SCI31 | 8 | 3 |
| Transmit data register 31 | TDR31 | 8 | H’FF9B | SCI31 | 8 | 3 |
| Serial status register 31 | SSR31 | 8 | H’FF9C | SCI31 | 8 | 3 |
| Receive data register 31 | RDR31 | 8 | H’FF9D | SCI31 | 8 | 3 |
| Serial mode register 32 | SMR32 | 8 | H'FFA8 | SCI32 | 8 | 3 |
| Bit rate register 32 | BRR32 | 8 | H'FFA9 | SCI32 | 8 | 3 |
| Serial control register 32 | SCR32 | 8 | H'FFAA | SCI32 | 8 | 3 |
| Transmit data register 32 | TDR32 | 8 | H’FFAB | SCI32 | 8 | 3 |
| Serial status register 32 | SSR32 | 8 | H’FFAC | SCI32 | 8 | 3 |
| Receive data register 32 | RDR32 | 8 | H’FFAD | SCI32 | 8 | 3 |
| Timer mode register A | TMA | 8 | H’FFB0 | Timer A | 8 | 2 |
| Timer counter A | TCA | 8 | H’FFB1 | Timer A | 8 | 2 |

| Register Name | Abbre-viation | Bit No | Address | Module Name | Data Bus Width | Access State |
|---|---|---|---|---|---|---|
| Timer control/status register W | TCSRW | 8 | H'FFB2 | WDT*2 | 8 | 2 |
| Timer counter W | TCW | 8 | H'FFB3 | WDT*2 | 8 | 2 |
| Timer mode register C | TMC | 8 | H'FFB4 | Timer C | 8 | 2 |
| Timer counter C / Timer load register C | TCC/ TLC | 8 | H'FFB5 | Timer C | 8 | 2 |
| Timer control register F | TCRF | 8 | H'FFB6 | Timer F | 8 | 2 |
| Timer control status register F | TCSRF | 8 | H'FFB7 | Timer F | 8 | 2 |
| 8-bit timer counter FH | TCFH | 8 | H'FFB8 | Timer F | 8 | 2 |
| 8-bit timer counter FL | TCFL | 8 | H'FFB9 | Timer F | 8 | 2 |
| Output compare register FH | OCRFH | 8 | H'FFBA | Timer F | 8 | 2 |
| Output compare register FL | OCRFL | 8 | H'FFBB | Timer F | 8 | 2 |
| Timer mode register G | TMG | 8 | H'FFBC | Timer G | 8 | 2 |
| Input capture register GF | ICRGF | 8 | H'FFBD | Timer G | 8 | 2 |
| Input capture register GR | ICRGR | 8 | H'FFBE | Timer G | 8 | 2 |
| LCD port control register | LPCR | 8 | H'FFC0 | LCD*3 | 8 | 2 |
| LCD control register | LCR | 8 | H'FFC1 | LCD*3 | 8 | 2 |
| LCD control register 2 | LCR2 | 8 | H'FFC2 | LCD*3 | 8 | 2 |
| A/D result register H | ADRRH | 8 | H'FFC4 | A/D converter | 8 | 2 |
| A/D result register L | ADRRL | 8 | H'FFC5 | A/D converter | 8 | 2 |
| A/D mode register | AMR | 8 | H'FFC6 | A/D converter | 8 | 2 |
| A/D start register | ADSR | 8 | H'FFC7 | A/D converter | 8 | 2 |
| Port mode register 1 | PMR1 | 8 | H'FFC8 | I/O port | 8 | 2 |
| Port mode register 2 | PMR2 | 8 | H'FFC9 | I/O port | 8 | 2 |
| Port mode register 3 | PMR3 | 8 | H'FFCA | I/O port | 8 | 2 |
| Port mode register 5 | PMR5 | 8 | H'FFCC | I/O port | 8 | 2 |
| PWM control register | PWCR | 8 | H'FFD0 | 14-bit PWM | 8 | 2 |
| PWM data register U | PWDRU | 8 | H'FFD1 | 14-bit PWM | 8 | 2 |
| PWM data register L | PWDRL | 8 | H'FFD2 | 14-bit PWM | 8 | 2 |
| Port data register 1 | PDR1 | 8 | H'FFD4 | I/O port | 8 | 2 |
| Port data register 3 | PDR3 | 8 | H'FFD6 | I/O port | 8 | 2 |
| Port data register 4 | PDR4 | 8 | H'FFD7 | I/O port | 8 | 2 |
| Port data register 5 | PDR5 | 8 | H'FFD8 | I/O port | 8 | 2 |

| Register Name | Abbre-viation | Bit No | Address | Module Name | Data Bus Width | Access State |
|---|---|---|---|---|---|---|
| Port data register 6 | PDR6 | 8 | H'FFD9 | I/O port | 8 | 2 |
| Port data register 7 | PDR7 | 8 | H'FFDA | I/O port | 8 | 2 |
| Port data register 8 | PDR8 | 8 | H'FFDB | I/O port | 8 | 2 |
| Port data register A | PDRA | 8 | H'FFDD | I/O port | 8 | 2 |
| Port data register B | PDRB | 8 | H'FFDE | I/O port | 8 | 2 |
| Port pull-up control register 1 | PUCR1 | 8 | H'FFE0 | I/O port | 8 | 2 |
| Port pull-up control register 3 | PUCR3 | 8 | H'FFE1 | I/O port | 8 | 2 |
| Port pull-up control register 5 | PUCR5 | 8 | H'FFE2 | I/O port | 8 | 2 |
| Port pull-up control register 6 | PUCR6 | 8 | H'FFE3 | I/O port | 8 | 2 |
| Port control register 1 | PCR1 | 8 | H'FFE4 | I/O port | 8 | 2 |
| Port control register 3 | PCR3 | 8 | H'FFE6 | I/O port | 8 | 2 |
| Port control register 4 | PCR4 | 8 | H'FFE7 | I/O port | 8 | 2 |
| Port control register 5 | PCR5 | 8 | H'FFE8 | I/O port | 8 | 2 |
| Port control register 6 | PCR6 | 8 | H'FFE9 | I/O port | 8 | 2 |
| Port control register 7 | PCR7 | 8 | H'FFEA | I/O port | 8 | 2 |
| Port control register 8 | PCR8 | 8 | H'FFEB | I/O port | 8 | 2 |
| Port control register A | PCRA | 8 | H'FFED | I/O port | 8 | 2 |
| System control register 1 | SYSCR1 | 8 | H'FFF0 | SYSTEM | 8 | 2 |
| System control register 2 | SYSCR2 | 8 | H'FFF1 | SYSTEM | 8 | 2 |
| IRQ edge select register | IEGR | 8 | H'FFF2 | Interrupts | 8 | 2 |
| Interrupt enable register 1 | IENR1 | 8 | H'FFF3 | Interrupts | 8 | 2 |
| Interrupt enable register 2 | IENR2 | 8 | H'FFF4 | Interrupts | 8 | 2 |
| Interrupt request register 1 | IRR1 | 8 | H'FFF6 | Interrupts | 8 | 2 |
| Interrupt request register 2 | IRR2 | 8 | H'FFF7 | Interrupts | 8 | 2 |
| Wakeup interrupt request register | IWPR | 8 | H'FFF9 | Interrupts | 8 | 2 |
| Clock stop register 1 | CKSTPR1 | 8 | H'FFFA | SYSTEM | 8 | 2 |
| Clock stop register 2 | CKSTPR2 | 8 | H'FFFB | SYSTEM | 8 | 2 |

Notes: 1. AEC: Asynchronous event counter
2. WDT: Watchdog timer
3. LCD: LCD controller/driver

## 15.2 Register Bits

Register bit names of the on-chip peripheral modules are described below.

| Register Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| FLMCR1 | — | SWE | ESU | PSU | EV | PV | E | P | ROM |
| FLMCR2 | FLER | — | — | — | — | — | — | — | |
| FLPWCR | PDWND | — | — | — | — | — | — | — | |
| EBR | EB7 | EB6 | EB5 | EB4 | EB3 | EB2 | EB1 | EB0 | |
| FENR | FLSHE | — | — | — | — | — | — | — | |
| WEGR | WKEGS7 | WKEGS6 | WKEGS5 | WKEGS4 | WKEGS3 | WKEGS2 | WKEGS1 | WKEGS0 | Interrupts |
| SPCR | — | — | SPC32 | SPC31 | SCINV3 | SCINV2 | SCINV1 | SCINV0 | SCI3 |
| CWOSR | — | — | — | — | — | — | — | CWOS | Timer A |
| ECCSR | OVH | OVL | — | CH2 | CUEH | CUEL | CRCH | CRCL | AEC*1 |
| ECH | ECH7 | ECH6 | ECH5 | ECH4 | ECH3 | ECH2 | ECH1 | ECH0 | |
| ECL | ECL7 | ECL6 | ECL5 | ECL4 | ECL3 | ECL2 | ECL1 | ECL0 | |
| SMR31 | COM31 | CHR31 | PE31 | PM31 | STOP31 | MP31 | CKS311 | CKS310 | SCI31 |
| BRR31 | BRR317 | BRR316 | BRR315 | BRR314 | BRR313 | BRR312 | BRR311 | BRR310 | |
| SCR31 | TIE31 | RIE31 | TE31 | RE31 | MPIE31 | TEIE31 | CKE311 | CKE310 | |
| TDR31 | TDR317 | TDR316 | TDR315 | TDR314 | TDR313 | TDR312 | TDR311 | TDR310 | |
| SSR31 | TDRE31 | RDRF31 | OER31 | FER31 | PER31 | TEND31 | MPBR31 | MPBT31 | |
| RDR31 | RDR317 | RDR316 | RDR315 | RDR314 | RDR313 | RDR312 | RDR311 | RDR310 | |
| SMR32 | COM32 | CHR32 | PE32 | PM32 | STOP32 | MP32 | CKS321 | CKS320 | SCI32 |
| BRR32 | BRR327 | BRR326 | BRR325 | BRR324 | BRR323 | BRR322 | BRR321 | BRR320 | |
| SCR32 | TIE32 | RIE32 | TE32 | RE32 | MPIE32 | TEIE32 | CKE321 | CKE320 | |
| TDR32 | TDR327 | TDR326 | TDR325 | TDR324 | TDR323 | TDR322 | TDR321 | TDR320 | |
| SSR32 | TDRE32 | RDRF32 | OER32 | FER32 | PER32 | TEND32 | MPBR32 | MPBT32 | |
| RDR32 | RDR327 | RDR326 | RDR325 | RDR324 | RDR323 | RDR322 | RDR321 | RDR320 | |
| TMA | TMA7 | TMA6 | TMA5 | — | TMA3 | TMA2 | TMA1 | TMA0 | Timer A |
| TCA | TCA7 | TCA6 | TCA5 | TCA4 | TCA3 | TCA2 | TCA1 | TCA0 | |
| TCSRW | B6WI | TCWE | B4WI | TCSRWE | B2WI | WDON | B0WI | WRST | WDT*2 |
| TCW | TCW7 | TCW6 | TCW5 | TCW4 | TCW3 | TCW2 | TCW1 | TCW0 | |

| Register Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| TMC | TMC7 | TMC6 | TMC5 | — | — | TMC2 | TMC1 | TMC0 | Timer C |
| TCC/ TLC | TCC7/ TLC7 | TCC6/ TLC6 | TCC5/ TLC5 | TCC4/ TLC4 | TCC3/ TLC3 | TCC2/ TLC2 | TCC1/ TLC1 | TCC0/ TLC0 | |
| TCRF | TOLH | CKSH2 | CKSH1 | CKSH0 | TOLL | CKSL2 | CKSL1 | CKSL0 | Timer F |
| TCSRF | OVFH | CMFH | OVIEH | CCLRH | OVFL | CMFL | OVIEL | CCLRL | |
| TCFH | TCFH7 | TCFH6 | TCFH5 | TCFH4 | TCFH3 | TCFH2 | TCFH1 | TCFH0 | |
| TCFL | TCFL7 | TCFL6 | TCFL5 | TCFL4 | TCFL3 | TCFL2 | TCFL1 | TCFL0 | |
| OCRFH | OCRFH7 | OCRFH6 | OCRFH5 | OCRFH4 | OCRFH3 | OCRFH2 | OCRFH1 | OCRFH0 | |
| OCRFL | OCRFL7 | OCRFL6 | OCRFL5 | OCRFL4 | OCRFL3 | OCRFL2 | OCRFL1 | OCRFL0 | |
| TMG | OVFH | OVFL | OVIE | IIEGS | CCLR1 | CCLR0 | CKS1 | CKS0 | Timer G |
| ICRGF | ICRGF7 | ICRGF6 | ICRGF5 | ICRGF4 | ICRGF3 | ICRGF2 | ICRGF1 | ICRGF0 | |
| ICRGR | ICRGR7 | ICRGR6 | ICRGR5 | ICRGR4 | ICRGR3 | ICRGR2 | ICRGR1 | ICRGR0 | |
| LPCR | DTS1 | DTS0 | CMX | — | SGS3 | SGS2 | SGS1 | SGS0 | LCD*3 |
| LCR | — | PSW | ACT | DISP | CKS3 | CKS2 | CKS1 | CKS0 | |
| LCR2 | LCDAB | — | — | — | CDS3 | CDS2 | CDS1 | CDS0 | |
| ADRRH | ADR9 | ADR8 | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | A/D converter |
| ADRRL | ADR1 | ADR0 | — | — | — | — | — | — | |
| AMR | CKS | TRGE | — | — | CH3 | CH2 | CH1 | CH0 | |
| ADSR | ADSF | — | — | — | — | — | — | — | |
| PMR1 | IRQ3 | IRQ2 | IRQ1 | IRQ4 | TMIG | TMOFH | TMOFL | TMOW | I/O port |
| PMR2 | EXCL | — | — | — | — | — | — | — | |
| PMR3 | AEVL | AEVH | WDCKS | NSC | IRQ0 | — | UD | PWM | |
| PMR5 | WKP7 | WKP6 | WKP5 | WKP4 | WKP3 | WKP2 | WKP1 | WKP0 | |
| PWCR | — | — | — | — | — | — | PWCR1 | PWCR0 | 14-bit PWM |
| PWDRU | — | — | PWDRU5 | PWDRU4 | PWDRU3 | PWDRU2 | PWDRU1 | PWDRU0 | |
| PWDRL | PWDRL7 | PWDRL6 | PWDRL5 | PWDRL4 | PWDRL3 | PWDRL2 | PWDRL1 | PWDRL0 | |
| PDR1 | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 | I/O port |
| PDR3 | P37 | P36 | P35 | P34 | P33 | P32 | P31 | P30 | |
| PDR4 | — | — | — | — | P43 | P42 | P41 | P40 | |
| PDR5 | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 | |
| PDR6 | P67 | P66 | P65 | P64 | P63 | P62 | P61 | P60 | |
| PDR7 | P77 | P76 | P75 | P74 | P73 | P72 | P71 | P70 | |
| PDR8 | P87 | P86 | P85 | P84 | P83 | P82 | P81 | P80 | |
| PDRA | — | — | — | — | PA3 | PA2 | PA1 | PA0 | |

| Register Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| PDRB | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 | I/O port |
| PUCR1 | PUCR17 | PUCR16 | PUCR15 | PUCR14 | PUCR13 | PUCR12 | PUCR11 | PUCR10 | |
| PUCR3 | PUCR37 | PUCR36 | PUCR35 | PUCR34 | PUCR33 | PUCR32 | PUCR31 | PUCR30 | |
| PUCR5 | PUCR57 | PUCR56 | PUCR55 | PUCR54 | PUCR53 | PUCR52 | PUCR51 | PUCR50 | |
| PUCR6 | PUCR67 | PUCR66 | PUCR65 | PUCR64 | PUCR63 | PUCR62 | PUCR61 | PUCR60 | |
| PCR1 | PCR17 | PCR16 | PCR15 | PCR14 | PCR13 | PCR12 | PCR11 | PCR10 | |
| PCR3 | PCR37 | PCR36 | PCR35 | PCR34 | PCR33 | PCR32 | PCR31 | PCR30 | |
| PCR4 | — | — | — | — | — | PCR42 | PCR41 | PCR40 | |
| PCR5 | PCR57 | PCR56 | PCR55 | PCR54 | PCR53 | PCR52 | PCR51 | PCR50 | |
| PCR6 | PCR67 | PCR66 | PCR65 | PCR64 | PCR63 | PCR62 | PCR61 | PCR60 | |
| PCR7 | PCR77 | PCR76 | PCR75 | PCR74 | PCR73 | PCR72 | PCR71 | PCR70 | |
| PCR8 | PCR87 | PCR86 | PCR85 | PCR84 | PCR83 | PCR82 | PCR81 | PCR80 | |
| PCRA | — | — | — | — | PCRA3 | PCRA2 | PCRA1 | PCRA0 | |
| SYSCR1 | SSBY | STS2 | STS1 | STS0 | LSON | — | MA1 | MA0 | SYSTEM |
| SYSCR2 | — | — | — | NESEL | DTON | MSON | SA1 | SA0 | |
| IEGR | — | — | — | IEG4 | IEG3 | IEG2 | IEG1 | IEG0 | Interrupts |
| IENR1 | IENTA | — | IENWP | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 | |
| IENR2 | IENDT | IENAD | — | IENTG | IENTFH | IENTFL | IENTC | IENEC | |
| IRR1 | IRRTA | — | — | IRRI4 | IRRI3 | IRRI2 | IRRI1 | IRRI0 | |
| IRR2 | IRRDT | IRRAD | — | IRRTG | IRRTFH | IRRTFL | IRRTC | IRREC | |
| IWPR | IWPF7 | IWPF6 | IWPF5 | IWPF4 | IWPF3 | IWPF2 | IWPF1 | IWPF0 | |
| CKSTPR1 | — | S31CKSTP | S32CKSTP | ADCKSTP | TGCKSTP | TFCKSTP | TCCKSTP | TACKSTP | SYSTEM |
| CKSTPR2 | — | — | — | — | AECKSTP | WDCKSTP | PWCKSTP | LDCKSTP | |

Notes: 1. AEC: Asynchronous event counter
2. WDT: Watchdog timer
3. LCD: LCD controller/driver

## 15.3 Register States in Each Operating Mode

| Register Abbreviation | Reset | Active | Sleep | Watch | Subactive | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|---|---|
| FLMCR1 | Initialized | — | — | Initialized | Initialized | Initialized | Initialized | ROM |
| FLMCR2 | Initialized | — | — | — | — | — | — | |
| FLPWCR | Initialized | — | — | — | — | — | — | |
| EBR | Initialized | — | — | Initialized | Initialized | Initialized | Initialized | |
| FENR | Initialized | — | — | — | — | — | — | |
| WEGR | Initialized | — | — | — | — | — | — | Interrupts |
| SPCR | Initialized | — | — | — | — | — | — | SCI3 |
| CWOSR | Initialized | — | — | — | — | — | — | Timer A |
| ECCSR | Initialized | — | — | — | — | — | — | AEC*1 |
| ECH | Initialized | — | — | — | — | — | — | |
| ECL | Initialized | — | — | — | — | — | — | |
| SMR31 | Initialized | — | — | Initialized | — | — | Initialized | SCI31 |
| BRR31 | Initialized | — | — | Initialized | — | — | Initialized | |
| SCR31 | Initialized | — | — | Initialized | — | — | Initialized | |
| TDR31 | Initialized | — | — | Initialized | — | — | Initialized | |
| SSR31 | Initialized | — | — | Initialized | — | — | Initialized | |
| RDR31 | Initialized | — | — | Initialized | — | — | Initialized | |
| SMR32 | Initialized | — | — | Initialized | — | — | Initialized | SCI32 |
| BRR32 | Initialized | — | — | Initialized | — | — | Initialized | |
| SCR32 | Initialized | — | — | Initialized | — | — | Initialized | |
| TDR32 | Initialized | — | — | Initialized | — | — | Initialized | |
| SSR32 | Initialized | — | — | Initialized | — | — | Initialized | |
| RDR32 | Initialized | — | — | Initialized | — | — | Initialized | |
| TMA | Initialized | — | — | — | — | — | — | Timer A |
| TCA | Initialized | — | — | — | — | — | — | |
| TCSRW | Initialized | — | — | — | — | — | — | WDT*2 |
| TCW | Initialized | — | — | — | — | — | — | |

| Register Abbreviation | Reset | Active | Sleep | Watch | Subactive | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|---|---|
| TMC | Initialized | — | — | — | — | — | — | Timer C |
| TCC | Initialized | — | — | — | — | — | — | |
| TLC | Initialized | — | — | — | — | — | — | |
| TCRF | Initialized | — | — | — | — | — | — | Timer F |
| TCSRF | Initialized | — | — | — | — | — | — | |
| TCFH | Initialized | — | — | — | — | — | — | |
| TCFL | Initialized | — | — | — | — | — | — | |
| OCRFH | Initialized | — | — | — | — | — | — | |
| OCRFL | Initialized | — | — | — | — | — | — | |
| TMG | Initialized | — | — | — | — | — | — | Timer G |
| ICRGF | Initialized | — | — | — | — | — | — | |
| ICRGR | Initialized | — | — | — | — | — | — | |
| LPCR | Initialized | — | — | — | — | — | — | LCD*3 |
| LCR | Initialized | — | — | — | — | — | — | |
| LCR2 | Initialized | — | — | — | — | — | — | |
| ADRRH | — | — | — | — | — | — | — | A/D converter |
| ADRRL | — | — | — | — | — | — | — | |
| AMR | Initialized | — | — | — | — | — | — | |
| ADSR | Initialized | — | — | Initialized | Initialized | Initialized | Initialized | |
| PMR1 | Initialized | — | — | — | — | — | — | I/O port |
| PMR2 | Initialized | — | — | — | — | — | — | |
| PMR3 | Initialized | — | — | — | — | — | — | |
| PMR5 | Initialized | — | — | — | — | — | — | |
| PWCR | Initialized | — | — | — | — | — | — | 14-bit PWM |
| PWDRU | Initialized | — | — | — | — | — | — | |
| PWDRL | Initialized | — | — | — | — | — | — | |
| PDR1 | Initialized | — | — | — | — | — | — | I/O port |
| PDR3 | Initialized | — | — | — | — | — | — | |
| PDR4 | Initialized | — | — | — | — | — | — | |
| PDR5 | Initialized | — | — | — | — | — | — | |
| PDR6 | Initialized | — | — | — | — | — | — | |
| PDR7 | Initialized | — | — | — | — | — | — | |
| PDR8 | Initialized | — | — | — | — | — | — | |
| PDRA | Initialized | — | — | — | — | — | — | |

| Register Abbreviation | Reset | Active | Sleep | Watch | Subactive | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|---|---|
| PDRB | Initialized | — | — | — | — | — | — | I/O port |
| PUCR1 | Initialized | — | — | — | — | — | — | |
| PUCR3 | Initialized | — | — | — | — | — | — | |
| PUCR5 | Initialized | — | — | — | — | — | — | |
| PUCR6 | Initialized | — | — | — | — | — | — | |
| PCR1 | Initialized | — | — | — | — | — | — | |
| PCR3 | Initialized | — | — | — | — | — | — | |
| PCR4 | Initialized | — | — | — | — | — | — | |
| PCR5 | Initialized | — | — | — | — | — | — | |
| PCR6 | Initialized | — | — | — | — | — | — | |
| PCR7 | Initialized | — | — | — | — | — | — | |
| PCR8 | Initialized | — | — | — | — | — | — | |
| PCRA | Initialized | — | — | — | — | — | — | |
| SYSCR1 | Initialized | — | — | — | — | — | — | SYSTEM |
| SYSCR2 | Initialized | — | — | — | — | — | — | |
| IEGR | Initialized | — | — | — | — | — | — | Interrupts |
| IENR1 | Initialized | — | — | — | — | — | — | |
| IENR2 | Initialized | — | — | — | — | — | — | |
| IRR1 | Initialized | — | — | — | — | — | — | |
| IRR2 | Initialized | — | — | — | — | — | — | |
| IWPR | Initialized | — | — | — | — | — | — | |
| CKSTPR1 | Initialized | — | — | — | — | — | — | SYSTEM |
| CKSTPR2 | Initialized | — | — | — | — | — | — | |

Notes: — is not initialized

1. AEC: Asynchronous event counter
2. WDT: Watchdog timer
3. LCD: LCD controller/driver

# Section 16 Electrical Characteristics

## 16.1 Absolute Maximum Ratings

Table 16.1 lists the absolute maximum ratings.

**Table 16.1 Absolute Maximum Ratings**

| Item | | Symbol | Value | Unit | Note |
|---|---|---|---|---|---|
| Power supply voltage | | $V_{CC}$ | –0.3 to +7.0 | V | *1 |
| | | $CV_{CC}$ | –0.3 to +4.3 | V | |
| Analog power supply voltage | | $AV_{CC}$ | –0.3 to +7.0 | V | |
| Input voltage | Other than port B | $V_{in}$ | –0.3 to $V_{CC}$ +0.3 | V | |
| | Port B | $AV_{in}$ | –0.3 to $AV_{CC}$ +0.3 | V | |
| Operating temperature | | $T_{opr}$ | –20 to +75*2 (regular specifications) | °C | |
| | | | –40 to +85*2 (wide-range temperature specifications) | | |
| Storage temperature | | $T_{stg}$ | –55 to +125 | °C | |

Notes: 1. Permanent damage may result if maximum ratings are exceeded. Normal operation should be under the conditions specified in Electrical Characteristics. Exceeding these values can result in incorrect operation and reduced reliability.

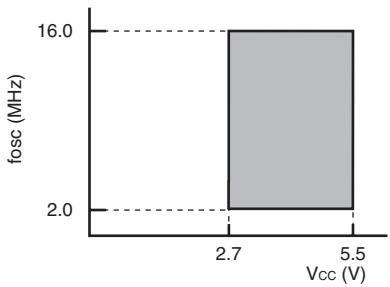
2. The operating temperature ranges from –20°C to +75°C when programming or erasing the flash memory.

## 16.2 Electrical Characteristics

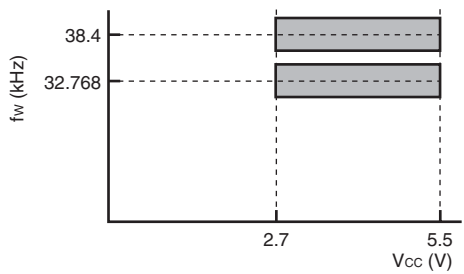
### 16.2.1 Power Supply Voltage and Operating Ranges

The power supply voltage and operating ranges (shaded portions) are shown below.

#### (1) Power Supply Voltage and Oscillation Frequency Range



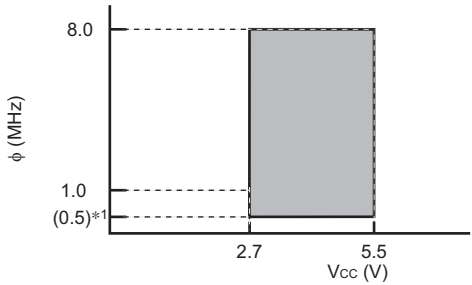

- Active (high-speed) mode
- Sleep (high-speed) mode



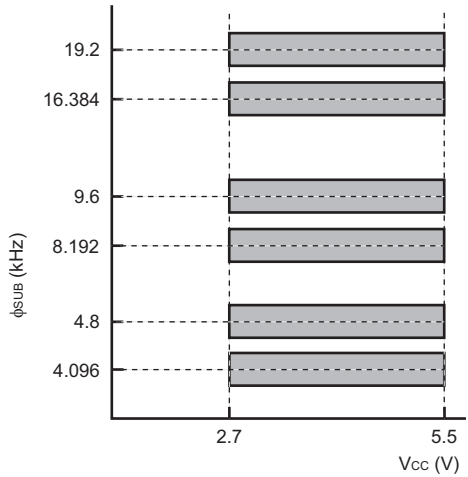

- All operating modes

Note: fosc is the oscillator frequency. When an external clock is used 1 MHz is the minimum fosc value.

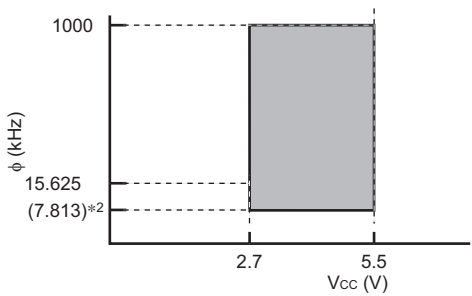
## (2) Power Supply Voltage and Operating Frequency Range




- Active (high-speed) mode
- Sleep (high-speed) mode (except CPU)



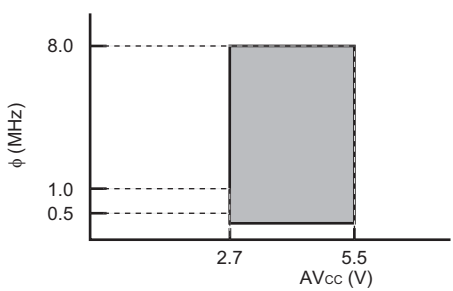

- Subactive mode
- Subsleep mode (except CPU)
- Watch mode (except CPU)



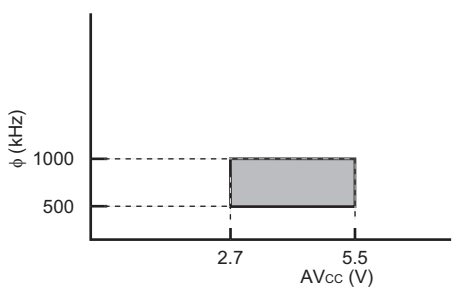

- Active (medium-speed) mode
- Sleep (medium-speed) mode (except A/D converter)

Notes 1. The figure in parentheses ( ) indicates the minimum operating frequency when an external clock is used. When the resonator is used the minimum operating frequency (ϕ) is 1 MHz.
2. The figure in parentheses ( ) indicates the minimum operating frequency when an external clock is used. When the resonator is used the minimum operating frequency (ϕ) is 15.625 kHz.

## (3) Analog Power Supply Voltage and A/D Converter Operating Range




- Active (high-speed) mode
- Sleep (high-speed) mode




- Active (medium-speed) mode
- Sleep (medium-speed) mode

### 16.2.2 DC Characteristics

Table 16.2 lists the DC characteristics.

**Table 16.2 DC Characteristics**

$V_{CC}$ = 2.7 V to 5.5 V, $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, unless otherwise specified

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Input high voltage | $V_{IH}$ | $\overline{RES}$, $\overline{WKP_0}$ to $\overline{WKP_7}$, $\overline{IRQ_0}$ to $\overline{IRQ_4}$, AEVL, AEVH, TMIC, TMIF, TMIG, ADTRG, $SCK_{31}$, $SCK_{32}$ | $V_{CC}$ × 0.8 | — | $V_{CC}$ + 0.3 | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | $V_{CC}$ × 0.9 | — | $V_{CC}$ + 0.3 | | Other than above | |
| | | $RXD_{31}$, $RXD_{32}$, UD | $V_{CC}$ × 0.7 | — | $V_{CC}$ + 0.3 | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | $V_{CC}$ × 0.8 | — | $V_{CC}$ + 0.3 | | Other than above | |
| | | $OSC_1$ | $V_{CC}$ × 0.8 | — | $V_{CC}$ + 0.3 | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | $V_{CC}$ × 0.9 | — | $V_{CC}$ + 0.3 | | Other than above | |
| | | $P1_0$ to $P1_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_3$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $PA_0$ to $PA_3$ | $V_{CC}$ × 0.7 | — | $V_{CC}$ + 0.3 | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | $V_{CC}$ × 0.8 | — | $V_{CC}$ + 0.3 | | Other than above | |
| | | $PB_0$ to $PB_7$ | $V_{CC}$ × 0.7 | — | $AV_{CC}$ + 0.3 | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | $V_{CC}$ × 0.8 | — | $AV_{CC}$ + 0.3 | | Other than above | |
| | | EXCL | $V_{CC}$ × 0.9 | — | $V_{CC}$ + 0.3 | V | | |

Note: Connect the TEST pin to $V_{SS}$.

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Input low voltage | $V_{IL}$ | $\overline{RES}$, $\overline{WKP}_0$ to $\overline{WKP}_7$, $\overline{IRQ}_0$ to $\overline{IRQ}_4$, AEVL, AEVH, TMIC, TMIF, TMIG, ADTRG, $SCK_{31}$, $SCK_{32}$ | – 0.3 | — | $V_{CC} \times 0.2$ | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | – 0.3 | — | $V_{CC} \times 0.1$ | | Other than above | |
| | | $RXD_{31}$, $RXD_{32}$, UD | – 0.3 | — | $V_{CC} \times 0.3$ | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | – 0.3 | — | $V_{CC} \times 0.2$ | | Other than above | |
| | | $OSC_1$ | – 0.3 | — | $V_{CC} \times 0.2$ | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | – 0.3 | — | $V_{CC} \times 0.1$ | | Other than above | |
| | | EXCL | – 0.3 | — | 0.1 $V_{CC}$ | V | | |
| | | $P1_0$ to $P1_7$ $P3_0$ to $P3_7$, $P4_0$ to $P4_3$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $PA_0$ to $PA_3$, $PB_0$ to $PB_7$ | – 0.3 | — | $V_{CC} \times 0.3$ | V | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | – 0.3 | — | $V_{CC} \times 0.2$ | | Other than above | |
| Output high voltage | $V_{OH}$ | $P1_0$ to $P1_7$ $P3_0$ to $P3_7$, $P4_0$ to $P4_2$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $PA_0$ to $PA_3$ | $V_{CC}$ – 1.0 | — | — | V | $V_{CC}$ = 4.0 V to 5.5 V $-I_{OH}$ = 1.0 mA | |
| | | | $V_{CC}$ – 0.5 | — | — | | $V_{CC}$ = 4.0 V to 5.5 V $-I_{OH}$ = 0.5 mA | |
| | | | $V_{CC}$ – 0.3 | — | — | | $-I_{OH}$ = 0.1 mA | |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Output low voltage | $V_{OL}$ | $P1_0$ to $P1_7$, $P4_0$ to $P4_2$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $PA_0$ to $PA_3$ | — | — | 0.6 | V | $V_{CC}$ = 4.0 V to 5.5 V $I_{OL}$ = 1.6 mA | |
| | | | — | — | 0.5 | | $I_{OL}$ = 0.4 mA | |
| | | $P3_0$ to $P3_7$ | — | — | 1.0 | | $V_{CC}$ = 4.0 V to 5.5 V $I_{OL}$ = 10 mA | |
| | | | — | — | 0.6 | | $V_{CC}$ = 4.0 V to 5.5 V $I_{OL}$ = 1.6 mA | |
| | | | — | — | 0.5 | | $I_{OL}$ = 0.4 mA | |
| Input/ output leakage current | $\lvert I_{IL} \rvert$ | $\overline{RES}$, $P4_3$, $P1_0$ to $P1_7$, $OSC_1$, $X_1$, $P3_0$ to $P3_7$, $P4_0$ to $P4_2$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $PA_0$ to $PA_3$ | — | — | 1.0 | μA | $V_{IN}$ = 0.5 V to $V_{CC}$ – 0.5 V | |
| | | $PB_0$ to $PB_7$ | — | — | 1.0 | | $V_{IN}$ = 0.5 V to $AV_{CC}$ – 0.5 V | |
| Pull-up MOS current | $-I_p$ | $P1_0$ to $P1_7$, $P3_0$ to $P3_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$ | 20 | — | 200 | μA | $V_{CC}$ = 5.0 V, $V_{IN}$ = 0.0 V | |
| | | | — | 40 | — | | $V_{CC}$ = 2.7 V, $V_{IN}$ = 0.0 V | Reference value |
| Input capacitance | $C_{in}$ | All input pins except power supply pin | — | — | 15.0 | pF | f = 1 MHz, $V_{IN}$ = 0.0 V, $T_a$ = 25°C | |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Active mode current consumption | $I_{OPE1}$ | $V_{CC}$ | — | 0.8 | — | mA | Active (high-speed) mode $V_{CC}$ = 2.7 V, $f_{OSC}$ = 2 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.2 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.0 | — | | Active (high-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 2 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.5 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 2.0 | — | | Active (high-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 4 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 2.4 | — | | | *2 *3 *4 |
| | | | — | 4.0 | 7.0 | | Active (high-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 10 MHz | *1 *3 *4 |
| | | | — | 4.9 | 7.0 | | | *2 *3 *4 |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Active mode current consumption | $I_{OPE2}$ | $V_{CC}$ | — | 0.4 | — | mA | Active (medium-speed) mode $V_{CC}$ = 2.7 V, $f_{OSC}$ = 2 MHz, $\phi_{OSC}$/128 | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 0.7 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 0.5 | — | | Active (medium-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 2 MHz, $\phi_{OSC}$/128 | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.0 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 0.8 | — | | Active (medium-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 4 MHz, $\phi_{OSC}$/128 | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.2 | — | | | *2 *3 *4 |
| | | | — | 1.2 | 3.0 | | Active (medium-speed) mode $V_{CC}$ = 5 V, $f_{OSC}$ = 10 MHz, $\phi_{OSC}$/128 | *1 *3 *4 |
| | | | — | 1.7 | 3.0 | | | *2 *3 *4 |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Sleep mode current consumption | $I_{SLEEP}$ | $V_{CC}$ | — | 0.5 | — | mA | $V_{CC}$ = 2.7 V, $f_{OSC}$ = 2 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 0.8 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 0.7 | — | | $V_{CC}$ = 5 V, $f_{OSC}$ = 2 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.2 | — | | | *2 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.1 | — | | $V_{CC}$ = 5 V, $f_{OSC}$ = 4 MHz | *1 *3 *4 Approx. max. value = 1.1 × Typ. |
| | | | — | 1.6 | — | | | *2 *3 *4 |
| | | | — | 1.9 | 5.0 | | $V_{CC}$ = 5 V, $f_{OSC}$ = 10 MHz | *1 *3 *4 |
| | | | — | 2.6 | 5.0 | | | *2 *3 *4 |
| Subactive mode current consumption | $I_{SUB}$ | $V_{CC}$ | — | 12 | — | μA | $V_{CC}$ = 2.7 V, LCD on, 32-kHz crystal resonator used ($\phi_{SUB} = \phi_W/8$) | *1 *3 *4 Reference value |
| | | | — | 15 | — | | | *2 *3 *4 Reference value |
| | | | — | 18 | 50 | | $V_{CC}$ = 2.7 V, LCD on, 32-kHz crystal resonator used ($\phi_{SUB} = \phi_W/2$) | *1 *3 *4 |
| | | | — | 30 | 50 | | | *2 *3 *4 |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Subsleep mode current consumption | $I_{SUBSP}$ | $V_{CC}$ | — | 3.8 | 16 | μA | $V_{CC}$ = 2.7 V, LCD on, 32-kHz crystal resonator used ($\phi_{SUB} = \phi_W/2$) | *3 *4 |
| Watch mode current consumption | $I_{WATCH}$ | $V_{CC}$ | — | 1.8 | — | μA | $V_{CC}$ = 2.7 V, $T_a$ = 25°C, 32-kHz crystal resonator used, LCD not used | *1 *3 *4 Reference value |
| | | | — | 1.8 | — | | | *2 *3 *4 Reference value |
| | | | — | 3.0 | 6.0 | | $V_{CC}$ = 2.7 V, 32-kHz crystal resonator used, LCD not used | *3 *4 |
| Standby mode current consumption | $I_{STBY}$ | $V_{CC}$ | — | 0.3 | — | μA | $V_{CC}$ = 2.7 V, $T_a$ = 25°C, 32-kHz crystal resonator not used | *1 *3 *4 Reference value |
| | | | — | 0.3 | — | | $V_{CC}$ = 2.7 V, $T_a$ = 25°C, 32-kHz crystal resonator not used | *2 *3 *4 Reference value |
| | | | — | 0.4 | — | | $V_{CC}$ = 5.0 V, $T_a$ = 25°C, 32-kHz crystal resonator not used | *1 *3 *4 Reference value |
| | | | — | 0.5 | — | | | *2 *3 *4 Reference value |
| | | | — | 1.0 | 5.0 | | 32-kHz crystal resonator not used | *3 *4 |
| RAM data retaining voltage | $V_{RAM}$ | $V_{CC}$ | 2.0 | — | — | V | | *5 |
| Allowable output low current (per pin) | $I_{OL}$ | Output pins except port 3 | — | — | 2.0 | mA | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | Port 3 | — | — | 10.0 | | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | All output pins | — | — | 0.5 | | | |

| Item | Symbol | Applicable Pins | Values Min | Typ | Max | Unit | Test Condition | Notes |
|---|---|---|---|---|---|---|---|---|
| Allowable output low current (total) | $\Sigma I_{OL}$ | Output pins except port 3 | — | — | 40.0 | mA | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | Port 3 | — | — | 80.0 | | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | All output pins | — | — | 20.0 | | | |
| Allowable output high current (per pin) | $-I_{OH}$ | All output pins | — | — | 2.0 | mA | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | — | — | 0.2 | | Other than above | |
| Allowable output high current (total) | $\Sigma -I_{OH}$ | All output pins | — | — | 15.0 | mA | $V_{CC}$ = 4.0 V to 5.5 V | |
| | | | — | — | 10.0 | | Other than above | |
| $V_{CC}$ start voltage | $V_{CCSTART}$ | $V_{CC}$ | 0 | — | 0.1 | V | | *2 |
| $V_{CC}$ rising gradient | $SV_{CC}$ | $V_{CC}$ | 0.05 | — | — | V/ms | | *2 |

Notes: Connect the TEST pin to $V_{SS}$.

1. Applies to the mask-ROM version.
2. Applies to the flash memory version.
3. Pin states when current consumption is measured

| Mode | $\overline{RES}$ Pin | Internal State | Other Pins | LCD Power Supply | Oscillator Pins |
|---|---|---|---|---|---|
| Active (high-speed) mode ($I_{OPE1}$)<br>Active (medium-speed) mode ($I_{OPE2}$) | $V_{CC}$ | Only CPU operates | $V_{CC}$ | Stops | System clock: crystal resonator<br>Subclock: Pin $X_1$ = GND |
| Sleep mode | $V_{CC}$ | Only all on-chip timers operate | $V_{CC}$ | Stops | |
| Subactive mode | $V_{CC}$ | Only CPU operates | $V_{CC}$ | Stops | System clock: crystal resonator<br>Subclock: crystal resonator |
| Subsleep mode | $V_{CC}$ | Only all on-chip timers operate<br>CPU stops | $V_{CC}$ | Stops | |
| Watch mode | $V_{CC}$ | Only clock time base operates<br>CPU stops | $V_{CC}$ | Stops | |
| Standby mode | $V_{CC}$ | CPU and timers both stop | $V_{CC}$ | Stops | System clock: crystal resonator<br>Subclock: Pin $X_1$ = GND |

4. Except current which flows to the pull-up MOS or output buffer
5. Voltage maintained in standby mode

### 16.2.3 AC Characteristics

Table 16.3 lists the control signal timing and table 16.4 lists the serial interface timing.

**Table 16.3 Control Signal Timing**

$V_{CC}$ = 2.7 V to 5.5 V, $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, unless otherwise specified

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| System clock oscillation frequency | $f_{OSC}$ | $OSC_1$, $OSC_2$ | 2.0 | — | 16.0 | MHz | | |
| OSC clock ($\phi_{OSC}$) cycle time | $t_{OSC}$ | $OSC_1$, $OSC_2$ | 62.5 | — | 500 (1000) | ns | | Figure 16.1 *2 |
| System clock ($\phi$) cycle time | $t_{cyc}$ | | 2 | — | 128 | $t_{OSC}$ | | |
| | | | — | — | 128 | μs | | |
| Subclock oscillation frequency | $f_W$ | $X_1$, $X_2$, EXCL | — | 32.768 or 38.4 | — | kHz | | |
| Watch clock ($\phi_W$) cycle time | $t_W$ | $X_1$, $X_2$, EXCL | — | 30.5 or 26.0 | — | μs | | Figure 16.1 |
| Subclock ($\phi_{SUB}$) cycle time | $t_{subcyc}$ | | 2 | — | 4 | $t_W$ | | *1 |
| Instruction cycle time | | | 2 | — | — | $t_{cyc}$ $t_{subcyc}$ | | |
| Oscillation stabilization time | $t_{rc}$ | $OSC_1$, $OSC_2$ | — | 20 | 45 | μs | Ceramic resonator ($V_{CC}$ = 3.0 to 5.5 V) | Figure 16.9 |
| | | | — | 80 | — | | Ceramic resonator other than above | |
| | | | — | 0.8 | 2 | ms | Crystal resonator | |
| | | | — | — | 50 | | Other than above | |
| | $t_{rc}$ | $X_1$, $X_2$ | — | — | 2.0 | s | | |
| External clock high width | $t_{CPH}$ | $OSC_1$ | 25 | — | — | ns | | Figure 16.1 |
| | | EXCL | — | 15.26 or 13.02 | — | μs | | Figure 16.1 |
| External clock low width | $t_{CPL}$ | $OSC_1$ | 25 | — | — | ns | | Figure 16.1 |
| | | EXCL | — | 15.26 or 13.02 | — | μs | | Figure 16.1 |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| External clock rise time | $t_{CPr}$ | $OSC_1$ | — | — | 6 | ns | | Figure 16.1 |
| | | EXCL | — | — | 55.0 | | | Figure 16.1 |
| External clock fall time | $t_{CPf}$ | $OSC_1$ | — | — | 6 | ns | | Figure 16.1 |
| | | EXCL | — | — | 55.0 | | | Figure 16.1 |
| $\overline{RES}$ pin low width | $t_{REL}$ | $\overline{RES}$ | 10 | — | — | $t_{cyc}$ | | Figure 16.2 |
| Input pin high width | $t_{IH}$ | $\overline{IRQ_0}$ to $\overline{IRQ_4}$, $\overline{WKP_0}$ to $\overline{WKP_7}$, $\overline{ADTRG}$, TMIC, TMIF, TMIG | 2 | — | — | $t_{cyc}$ $t_{subcyc}$ | | Figure 16.3 |
| | | AEVL, AEVH | 32 | — | — | ns | | |
| Input pin low width | $t_{IL}$ | $\overline{IRQ_0}$ to $\overline{IRQ_4}$, $\overline{WKP_0}$ to $\overline{WKP_7}$, $\overline{ADTRG}$, TMIC, TMIF, TMIG | 2 | — | — | $t_{cyc}$ $t_{subcyc}$ | | Figure 16.3 |
| | | AEVL, AEVH | 32 | — | — | ns | | |
| UD pin minimum transition width | $t_{UDH}$ $t_{UDL}$ | UD | 4 | — | — | $t_{cyc}$ $t_{subcyc}$ | | Figure 16.4 |

Notes: 1. Determined by the SA1 and SA0 bits in the system control register 2 (SYSCR2).

2. The figure in parentheses ( ) indicates the maximum fosc value when an external clock is used.

### Table 16.4 Serial Interface (SCI3) Timing

$V_{CC}$ = 2.7 V to 5.5 V, $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, unless otherwise specified

| Item | | Symbol | Values Min | Typ | Max | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| Input clock cycle | Asynchronous | $t_{scyc}$ | 4 | — | — | $t_{cyc}$ or $t_{subcyc}$ | | Figure 16.5 |
| | Clocked synchronous | | 6 | — | — | | | |
| Input clock pulse width | | $t_{SCKW}$ | 0.4 | — | 0.6 | $t_{scyc}$ | | Figure 16.5 |
| Transmit data delay time (clocked synchronous) | | $t_{TXD}$ | — | — | 1 | $t_{cyc}$ or $t_{subcyc}$ | | Figure 16.6 |
| Receive data setup time (clocked synchronous) | | $t_{RXS}$ | 200 | — | — | ns | | Figure 16.6 |
| Receive data hold time (clocked synchronous) | | $t_{RXH}$ | 200 | — | — | ns | | Figure 16.6 |

## 16.2.4 A/D Converter Characteristics

Table 16.5 shows the A/D converter characteristics.

### Table 16.5 A/D Converter Characteristics

$V_{CC}$ = 2.7 V to 5.5 V, $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, unless otherwise specified

| Item | Symbol | Applicable Pins | Values Min | Typ | Max | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| Analog power supply voltage | $AV_{CC}$ | $AV_{CC}$ | 2.7 | — | 5.5 | V | | *1 |
| Analog input voltage | $AV_{IN}$ | $AN_0$ to $AN_3$ | – 0.3 | — | $AV_{CC}$ + 0.3 | V | | |
| Analog power supply current | $AI_{OPE}$ | $AV_{CC}$ | — | — | 1.5 | mA | $AV_{CC}$ = 5.0 V | |
| | $AI_{STOP1}$ | $AV_{CC}$ | — | 600 | — | μA | | *2 Reference value |
| | $AI_{STOP2}$ | $AV_{CC}$ | — | — | 5.0 | μA | | *3 |
| Analog input capacitance | $C_{AIN}$ | $AN_0$ to $AN_7$ | — | — | 15.0 | pF | | |
| Allowable signal source impedance | $R_{AIN}$ | | — | — | 10.0 | kΩ | | |
| Resolution (data length) | | | — | — | 10 | bit | | |

| Item | Symbol | Applicable Pins | Values Min | Values Typ | Values Max | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| Nonlinearity error | | | — | — | ±3.5 | LSB | $AV_{CC}$ = 4.0 V to 5.5 V | |
| | | | — | — | ±7.5 | | $AV_{CC}$ = 2.7 V to 5.5 V | |
| Quantization error | | | — | — | ±0.5 | LSB | | |
| Absolute accuracy | | | — | ±2.0 | ±4.0 | LSB | $AV_{CC}$ = 4.0 V to 5.5 V | |
| | | | — | ±2.0 | ±8.0 | | $AV_{CC}$ = 2.7 V to 5.5 V | |
| Conversion time | | | 7.8 | — | 124 | μs | | |

Notes: 1. Set $AV_{CC} = V_{CC}$ when the A/D converter is not used.
2. $AI_{STOP1}$ is the current in active and sleep modes while the A/D converter is idle.
3. $AI_{STOP2}$ is the current at reset and in standby, watch, subactive, and subsleep modes while the A/D converter is idle.

### 16.2.5 LCD Characteristics

Table 16.6 shows the LCD characteristics.

**Table 16.6 LCD Characteristics**

$V_{CC}$ = 2.7 V to 5.5 V, $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, unless otherwise specified

| Item | Symbol | Applicable Pins | Values | | | Unit | Test Condition | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | | | |
| Segment driver step-down voltage | $V_{DS}$ | $SEG_1$ to $SEG_{32}$ | — | — | 0.6 | V | $I_D$ = 2 μA<br>V1 = 2.7 V to 5.5 V | *1 |
| Common driver step-down voltage | $V_{DC}$ | $COM_1$ to $COM_4$ | — | — | 0.3 | V | $I_D$ = 2 μA<br>V1 = 2.7 V to 5.5 V | *1 |
| LCD power supply split-resistance | $R_{LCD}$ | | 1.5 | 3.0 | 7.0 | MΩ | Between V1 and $V_{SS}$ | |
| Liquid crystal display voltage | $V_{LCD}$ | $V_1$ | 2.7 | — | 5.5 | V | | *2 |

Notes: 1. The voltage step-down from power supply pins V1, V2, V3, and $V_{SS}$ to each segment pin or common pin.

2. When the liquid crystal display voltage is supplied from an external power supply, ensure that the following relationship is maintained: V1 ≥ V2 ≥ V3 ≥ $V_{SS}$.

### 16.2.6 Flash Memory Characteristics

**Table 16.7 Flash Memory Characteristics**

Condition: $AV_{CC}$ = 2.7 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0.0 V, $V_{CC}$ = 2.7 V to 5.5 V (range of operating voltage when reading), $V_{CC}$ = 3.0 V to 5.5 V (range of operating voltage when programming/erasing), $T_a$ = –20°C to +75°C (range of operating temperature when programming/erasing: product with regular specifications, product with wide-range temperature specifications)

| Item | | Symbol | Values Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Programming time*1*2*4 | | $t_P$ | — | 7 | 200 | ms/128 bytes | |
| Erase time*1*3*5 | | $t_E$ | — | 100 | 1200 | ms/block | |
| Reprogramming count | | $N_{WEC}$ | 1000*8 | 10000*9 | — | times | |
| Data retain period | | $t_{DRP}$ | 10*10 | — | — | year | |
| Programming | Wait time after SWE-bit setting*1 | x | 1 | — | — | μs | |
| | Wait time after PSU-bit setting*1 | y | 50 | — | — | μs | |
| | Wait time after P-bit setting*1*4 | z1 | 28 | 30 | 32 | μs | 1 ≤ n ≤ 6 |
| | | z2 | 198 | 200 | 202 | μs | 7 ≤ n ≤ 1000 |
| | | z3 | 8 | 10 | 12 | μs | Additional programming |
| | Wait time after P-bit clear*1 | α | 5 | — | — | μs | |
| | Wait time after PSU-bit clear*1 | β | 5 | — | — | μs | |
| | Wait time after PV-bit setting*1 | γ | 4 | — | — | μs | |
| | Wait time after dummy write*1 | ε | 2 | — | — | μs | |
| | Wait time after PV-bit clear*1 | η | 2 | — | — | μs | |
| | Wait time after SWE-bit clear*1 | θ | 100 | — | — | μs | |
| | Maximum programming count*1*4*5 | N | — | — | 1000 | times | |

| Item | | Symbol | Values Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Erase | Wait time after SWE-bit setting*1 | x | 1 | — | — | μs | |
| | Wait time after ESU-bit setting*1 | y | 100 | — | — | μs | |
| | Wait time after E-bit setting*1*6 | z | 10 | — | 100 | ms | |
| | Wait time after E-bit clear*1 | α | 10 | — | — | μs | |
| | Wait time after ESU-bit clear*1 | β | 10 | — | — | μs | |
| | Wait time after EV-bit setting*1 | γ | 20 | — | — | μs | |
| | Wait time after dummy write*1 | ε | 2 | — | — | μs | |
| | Wait time after EV-bit clear*1 | η | 4 | — | — | μs | |
| | Wait time after SWE-bit clear*1 | θ | 100 | — | — | μs | |
| | Maximum erase count*1*6*7 | N | — | — | 120 | times | |

Notes:
1. Set the times according to the program/erase algorithms.
2. Programming time per 128 bytes (Shows the total period for which the P bit in FLMCR1 is set. It does not include the programming verification time.)
3. Block erase time (Shows the total period for which the E bit in FLMCR1 is set. It does not include the erase verification time.)
4. Maximum programming time ($t_P$ (max))
   $t_P$ (max) = Wait time after P-bit setting (z) × maximum number of writes (N)
5. The maximum number of writes (N) should be set according to the actual set value of z1, z2, and z3 to allow programming within the maximum programming time ($t_P$ (max)).
   The wait time after P-bit setting (z1 and z2) should be alternated according to the number of writes (n) as follows:
   $1 \leq n \leq 6$ z1 = 30 μs
   $7 \leq n \leq 1000$ z2 = 200 μs
6. Maximum erase time ($t_E$ (max))
   $t_E$ (max) = Wait time after E-bit setting (z) × maximum erase count (N)
7. The maximum number of erases (N) should be set according to the actual set value of z to allow erasing within the maximum erase time ($t_E$ (max)).
8. This minimum value guarantees all characteristics after reprogramming (the guaranteed range is from 1 to the minimum value).
9. Reference value when the temperature is 25°C (normally reprogramming will be performed by this count).
10. This is a data retain characteristic when reprogramming is performed within the specification range including this minimum value.

## 16.3 Operation Timing

Figures 16.1 to 16.6 show timing diagrams.

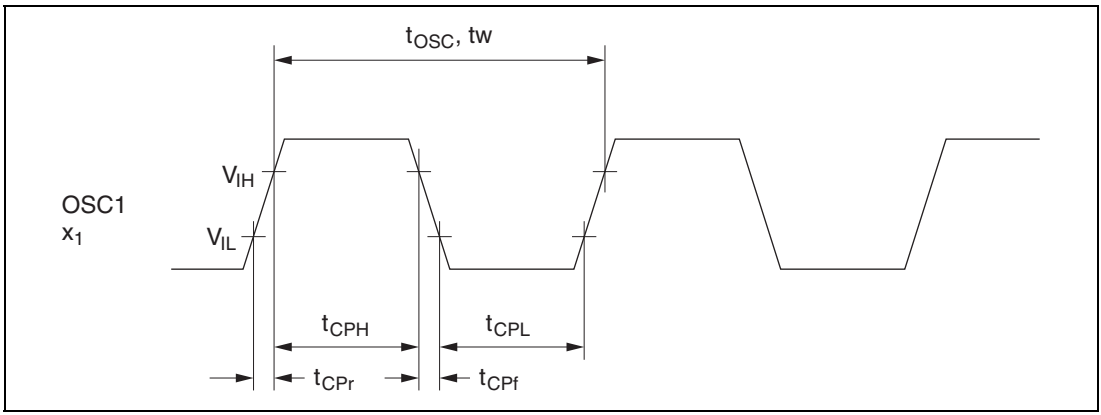



**Figure 16.1 Clock Input Timing**

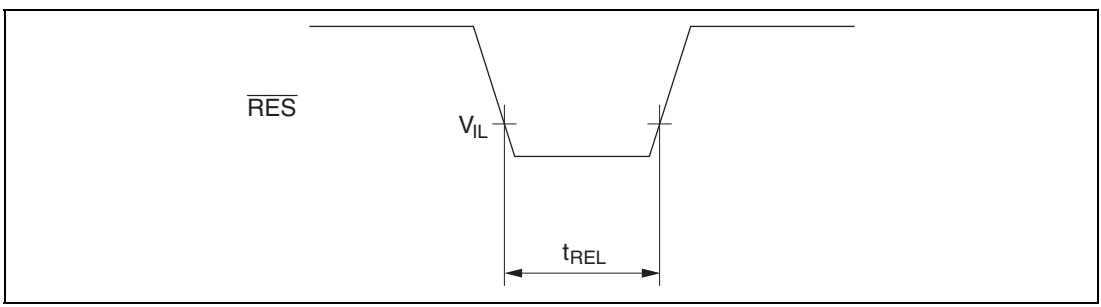



**Figure 16.2 $\overline{\text{RES}}$ Low Width**

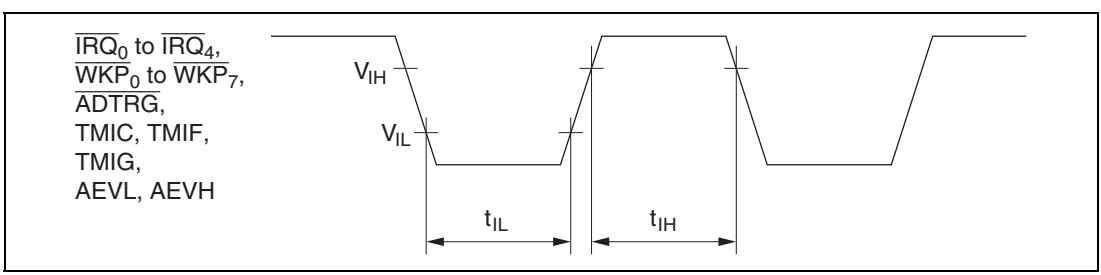



**Figure 16.3 Input Timing**

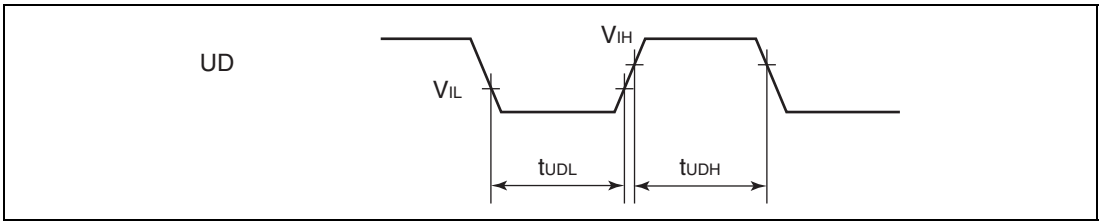



**Figure 16.4 UD Pin Minimum Modulation Width Timing**

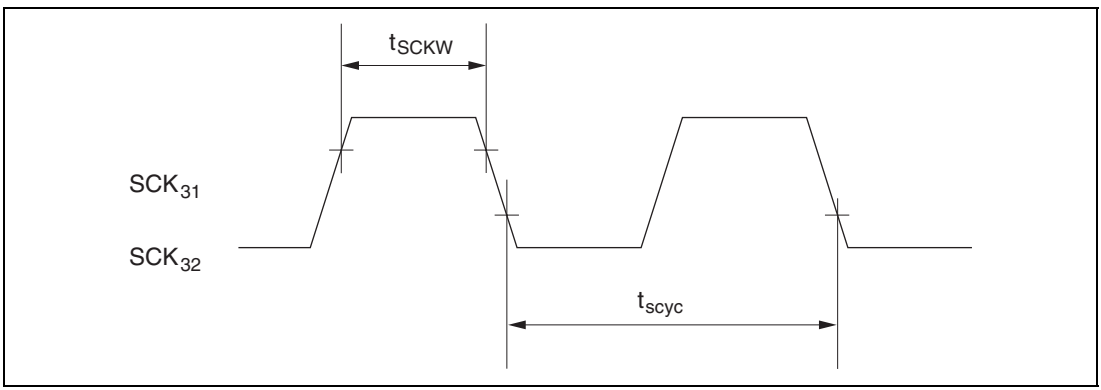
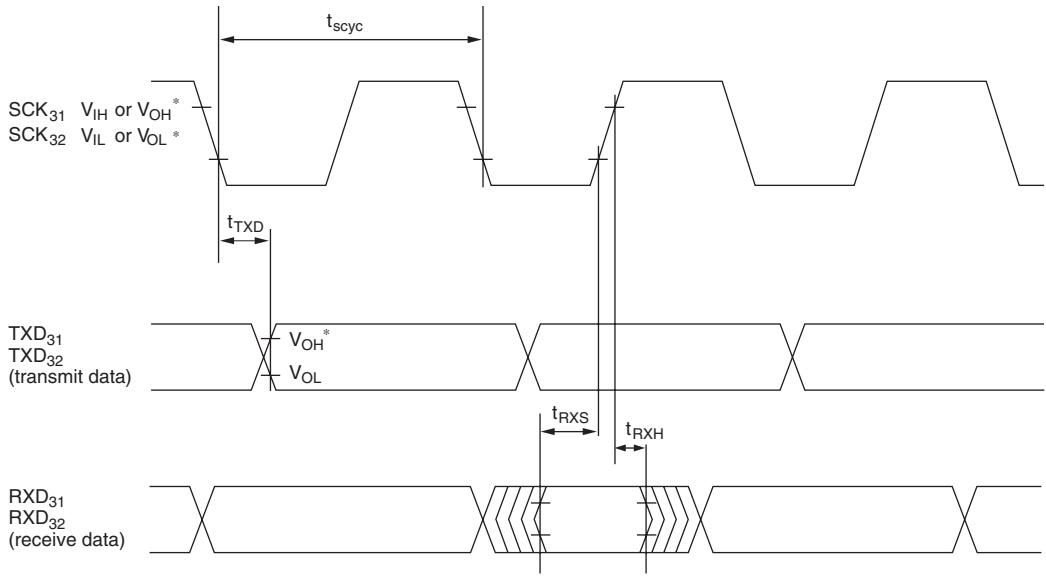



**Figure 16.5 SCK3 Input Clock Timing**




Note: * Output timing reference levels

| | |
|---|---|
| Output high | $V_{OH}$ = 1/2Vcc + 0.2 V |
| Output low | $V_{OL}$ = 0.8 V |

Load conditions are shown in figure 16.7.

**Figure 16.6 SCI3 Synchronous Mode Input/Output Timing**

## 16.4 Output Load Circuit

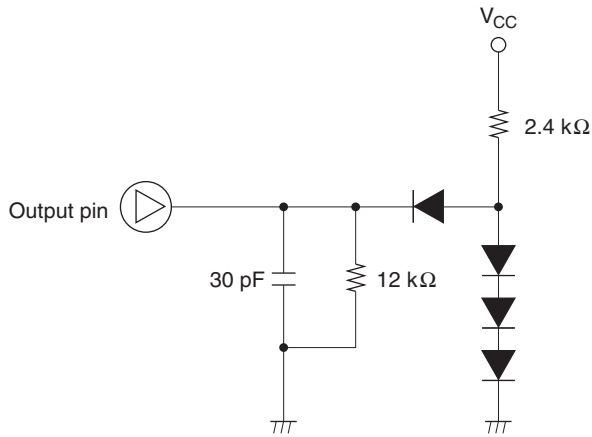
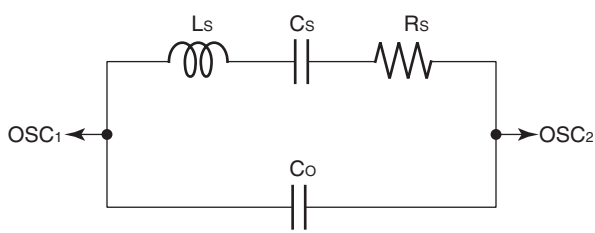



**Figure 16.7 Output Load Condition**

## 16.5 Resonator




Ceramic Oscillator Parameters

| Frequency | 4 MHz | Manufacturer | Products Name |
|---|---|---|---|
| $R_S$ | Manufacturer's Publicly Released Values Max. 8.8 | MURATA | CSTLS 4M00G 53/56 |
| $C_O$ | Max. 36 pF | | |

Crystal Oscillator Parameters

| Frequency | 4.193 MHz | Manufacturer | Products Name |
|---|---|---|---|
| $R_S$ | Manufacturer's Publicly Released Values Max. 100 | Nihon Denpa Kogyo | NR-18 |
| $C_O$ | Max. 16 pF | | |

**Figure 16.8 Resonator Equivalent Circuit**

Crystal resonator

| Resonating Frequency | Manufacturer | Model | $C_1$, $C_2$ |
|---|---|---|---|
| 4 MHz | Nihon Denpa Kogyo | NR-18 | 12pF ± 20% |
| 10 MHz | | | |

Ceramic resonator

| Resonating Frequency | Manufacturer | Model | $C_1$, $C_2$ |
|---|---|---|---|
| 2 MHz | MURATA | CSTCC2M00G53-B0 | 15pF ± 20% |
| | | CSTCC2M00G56-B0 | 47pF ± 20% |
| 4 MHz | | CSTLS4M00G53-B0 | 15pF ± 20% |
| | | CSTLS4M00G56-B0 | 47pF ± 20% |
| 10 MHz | | CSTLS10M0G53-B0 | 15pF ± 20% |
| | | CSTLS10M0G56-B0 | 47pF ± 20% |

**Figure 16.9 Resonator Equivalent Circuit**

## 16.6 Usage Note

Each of the products covered in this manual satisfy the electrical characteristics indicated. However, the actual electrical characteristics, operating margin and noise margin may differ from the indicated values due to differences in the manufacturing process, built-in ROM, layout pattern and other factors.

If a system evaluation test is conducted with the flash memory version, when switching to a mask ROM version, perform the same evaluation test with the mask ROM version.

# Appendix

## A. Instruction Set

### A.1 Instruction List

#### Condition Code

| Symbol | Description |
|---|---|
| Rd | General destination register |
| Rs | General source register |
| Rn | General register |
| ERd | General destination register (address register or 32-bit register) |
| ERs | General source register (address register or 32-bit register) |
| ERn | General register (32-bit register) |
| (EAd) | Destination operand |
| (EAs) | Source operand |
| PC | Program counter |
| SP | Stack pointer |
| CCR | Condition-code register |
| N | N (negative) flag in CCR |
| Z | Z (zero) flag in CCR |
| V | V (overflow) flag in CCR |
| C | C (carry) flag in CCR |
| disp | Displacement |
| → | Transfer from the operand on the left to the operand on the right, or transition from the state on the left to the state on the right |
| + | Addition of the operands on both sides |
| – | Subtraction of the operand on the right from the operand on the left |
| × | Multiplication of the operands on both sides |
| ÷ | Division of the operand on the left by the operand on the right |
| ∧ | Logical AND of the operands on both sides |
| ∨ | Logical OR of the operands on both sides |
| ⊕ | Logical exclusive OR of the operands on both sides |

| Symbol | Description |
|---|---|
| ¬ | NOT (logical complement) |
| ( ), < > | Contents of operand |

Note: General registers include 8-bit registers (R0H to R7H and R0L to R7L) and 16-bit registers (R0 to R7 and E0 to E7).

## Condition Code Notation (cont)

| Symbol | Description |
|---|---|
| ↕ | Changed according to execution result |
| * | Undetermined (no guaranteed value) |
| 0 | Cleared to 0 |
| 1 | Set to 1 |
| — | Not affected by execution of the instruction |
| Δ | Varies depending on conditions, described in notes |

# Table A.1 Instruction Set

## 1. Data Transfer Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| MOV | MOV.B #xx:8, Rd | B | 2 | | | | | | | | | #xx:8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | MOV.B Rs, Rd | B | | 2 | | | | | | | | Rs8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | MOV.B @ERs, Rd | B | | | 2 | | | | | | | @ERs → Rd8 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.B @(d:16, ERs), Rd | B | | | | 4 | | | | | | @(d:16, ERs) → Rd8 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B @(d:24, ERs), Rd | B | | | | 8 | | | | | | @(d:24, ERs) → Rd8 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.B @ERs+, Rd | B | | | | | 2 | | | | | @ERs → Rd8<br>ERs32+1 → ERs32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B @aa:8, Rd | B | | | | | | 2 | | | | @aa:8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.B @aa:16, Rd | B | | | | | | 4 | | | | @aa:16 → Rd8 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B @aa:24, Rd | B | | | | | | 6 | | | | @aa:24 → Rd8 | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.B Rs, @ERd | B | | | 2 | | | | | | | Rs8 → @ERd | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.B Rs, @(d:16, ERd) | B | | | | 4 | | | | | | Rs8 → @(d:16, ERd) | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B Rs, @(d:24, ERd) | B | | | | 8 | | | | | | Rs8 → @(d:24, ERd) | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.B Rs, @–ERd | B | | | | | 2 | | | | | ERd32–1 → ERd32<br>Rs8 → @ERd | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B Rs, @aa:8 | B | | | | | | 2 | | | | Rs8 → @aa:8 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.B Rs, @aa:16 | B | | | | | | 4 | | | | Rs8 → @aa:16 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.B Rs, @aa:24 | B | | | | | | 6 | | | | Rs8 → @aa:24 | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.W #xx:16, Rd | W | 4 | | | | | | | | | #xx:16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.W Rs, Rd | W | | 2 | | | | | | | | Rs16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | MOV.W @ERs, Rd | W | | | 2 | | | | | | | @ERs → Rd16 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.W @(d:16, ERs), Rd | W | | | | 4 | | | | | | @(d:16, ERs) → Rd16 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W @(d:24, ERs), Rd | W | | | | 8 | | | | | | @(d:24, ERs) → Rd16 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.W @ERs+, Rd | W | | | | | 2 | | | | | @ERs → Rd16<br>ERs32+2 → @ERd32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W @aa:16, Rd | W | | | | | | 4 | | | | @aa:16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W @aa:24, Rd | W | | | | | | 6 | | | | @aa:24 → Rd16 | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.W Rs, @ERd | W | | | 2 | | | | | | | Rs16 → @ERd | — | — | ↕ | ↕ | 0 | — | 4 | |
| | MOV.W Rs, @(d:16, ERd) | W | | | | 4 | | | | | | Rs16 → @(d:16, ERd) | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W Rs, @(d:24, ERd) | W | | | | 8 | | | | | | Rs16 → @(d:24, ERd) | — | — | ↕ | ↕ | 0 | — | 10 | |

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| MOV | MOV.W Rs, @–ERd | W | | | | | 2 | | | | | ERd32–2 → ERd32<br>Rs16 → @ERd | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W Rs, @aa:16 | W | | | | | | 4 | | | | Rs16 → @aa:16 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.W Rs, @aa:24 | W | | | | | | 6 | | | | Rs16 → @aa:24 | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.L #xx:32, ERd | L | 6 | | | | | | | | | #xx:32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | MOV.L ERs, ERd | L | | 2 | | | | | | | | ERs32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | MOV.L @ERs, ERd | L | | | 4 | | | | | | | @ERs → ERd32 | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.L @(d:16, ERs), ERd | L | | | | 6 | | | | | | @(d:16, ERs) → ERd32 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L @(d:24, ERs), ERd | L | | | | 10 | | | | | | @(d:24, ERs) → ERd32 | — | — | ↕ | ↕ | 0 | — | 14 | |
| | MOV.L @ERs+, ERd | L | | | | | 4 | | | | | @ERs → ERd32<br>ERs32+4 → ERs32 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L @aa:16, ERd | L | | | | | | 6 | | | | @aa:16 → ERd32 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L @aa:24, ERd | L | | | | | | 8 | | | | @aa:24 → ERd32 | — | — | ↕ | ↕ | 0 | — | 12 | |
| | MOV.L ERs, @ERd | L | | | 4 | | | | | | | ERs32 → @ERd | — | — | ↕ | ↕ | 0 | — | 8 | |
| | MOV.L ERs, @(d:16, ERd) | L | | | | 6 | | | | | | ERs32 → @(d:16, ERd) | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L ERs, @(d:24, ERd) | L | | | | 10 | | | | | | ERs32 → @(d:24, ERd) | — | — | ↕ | ↕ | 0 | — | 14 | |
| | MOV.L ERs, @–ERd | L | | | | | 4 | | | | | ERd32–4 → ERd32<br>ERs32 → @ERd | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L ERs, @aa:16 | L | | | | | | 6 | | | | ERs32 → @aa:16 | — | — | ↕ | ↕ | 0 | — | 10 | |
| | MOV.L ERs, @aa:24 | L | | | | | | 8 | | | | ERs32 → @aa:24 | — | — | ↕ | ↕ | 0 | — | 12 | |
| POP | POP.W Rn | W | | | | | | | | | 2 | @SP → Rn16<br>SP+2 → SP | — | — | ↕ | ↕ | 0 | — | 6 | |
| | POP.L ERn | L | | | | | | | | | 4 | @SP → ERn32<br>SP+4 → SP | — | — | ↕ | ↕ | 0 | — | 10 | |
| PUSH | PUSH.W Rn | W | | | | | | | | | 2 | SP–2 → SP<br>Rn16 → @SP | — | — | ↕ | ↕ | 0 | — | 6 | |
| | PUSH.L ERn | L | | | | | | | | | 4 | SP–4 → SP<br>ERn32 → @SP | — | — | ↕ | ↕ | 0 | — | 10 | |
| MOVFPE | MOVFPE @aa:16, Rd | B | | | | | | 4 | | | | Cannot be used in this LSI | Cannot be used in this LSI | | | | | | | |
| MOVTPE | MOVTPE Rs, @aa:16 | B | | | | | | 4 | | | | Cannot be used in this LSI | Cannot be used in this LSI | | | | | | | |

## 2. Arithmetic Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| ADD | ADD.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8+#xx:8 → Rd8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | ADD.B Rs, Rd | B | | 2 | | | | | | | | Rd8+Rs8 → Rd8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | ADD.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16+#xx:16 → Rd16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 4 | |
| | ADD.W Rs, Rd | W | | 2 | | | | | | | | Rd16+Rs16 → Rd16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 2 | |
| | ADD.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32+#xx:32 → ERd32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 6 | |
| | ADD.L ERs, ERd | L | | 2 | | | | | | | | ERd32+ERs32 → ERd32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 2 | |
| ADDX | ADDX.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8+#xx:8 +C → Rd8 | — | ↕ | ↕ | (3) | ↕ | ↕ | 2 | |
| | ADDX.B Rs, Rd | B | | 2 | | | | | | | | Rd8+Rs8 +C → Rd8 | — | ↕ | ↕ | (3) | ↕ | ↕ | 2 | |
| ADDS | ADDS.L #1, ERd | L | | 2 | | | | | | | | ERd32+1 → ERd32 | — | — | — | — | — | — | 2 | |
| | ADDS.L #2, ERd | L | | 2 | | | | | | | | ERd32+2 → ERd32 | — | — | — | — | — | — | 2 | |
| | ADDS.L #4, ERd | L | | 2 | | | | | | | | ERd32+4 → ERd32 | — | — | — | — | — | — | 2 | |
| INC | INC.B Rd | B | | 2 | | | | | | | | Rd8+1 → Rd8 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | INC.W #1, Rd | W | | 2 | | | | | | | | Rd16+1 → Rd16 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | INC.W #2, Rd | W | | 2 | | | | | | | | Rd16+2 → Rd16 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | INC.L #1, ERd | L | | 2 | | | | | | | | ERd32+1 → ERd32 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | INC.L #2, ERd | L | | 2 | | | | | | | | ERd32+2 → ERd32 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| DAA | DAA Rd | B | | 2 | | | | | | | | Rd8 decimal adjust → Rd8 | — | * | ↕ | ↕ | * | — | 2 | |
| SUB | SUB.B Rs, Rd | B | | 2 | | | | | | | | Rd8–Rs8 → Rd8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | SUB.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16–#xx:16 → Rd16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 4 | |
| | SUB.W Rs, Rd | W | | 2 | | | | | | | | Rd16–Rs16 → Rd16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 2 | |
| | SUB.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32–#xx:32 → ERd32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 6 | |
| | SUB.L ERs, ERd | L | | 2 | | | | | | | | ERd32–ERs32 → ERd32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 2 | |
| SUBX | SUBX.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8–#xx:8–C → Rd8 | — | ↕ | ↕ | (3) | ↕ | ↕ | 2 | |
| | SUBX.B Rs, Rd | B | | 2 | | | | | | | | Rd8–Rs8–C → Rd8 | — | ↕ | ↕ | (3) | ↕ | ↕ | 2 | |
| SUBS | SUBS.L #1, ERd | L | | 2 | | | | | | | | ERd32–1 → ERd32 | — | — | — | — | — | — | 2 | |
| | SUBS.L #2, ERd | L | | 2 | | | | | | | | ERd32–2 → ERd32 | — | — | — | — | — | — | 2 | |
| | SUBS.L #4, ERd | L | | 2 | | | | | | | | ERd32–4 → ERd32 | — | — | — | — | — | — | 2 | |
| DEC | DEC.B Rd | B | | 2 | | | | | | | | Rd8–1 → Rd8 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | DEC.W #1, Rd | W | | 2 | | | | | | | | Rd16–1 → Rd16 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | DEC.W #2, Rd | W | | 2 | | | | | | | | Rd16–2 → Rd16 | — | — | ↕ | ↕ | ↕ | — | 2 | |

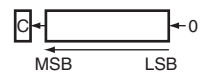
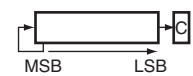
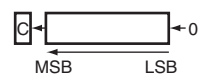
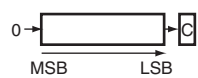
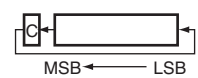
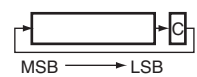
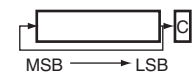
| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| DEC | DEC.L #1, ERd | L | | 2 | | | | | | | | ERd32–1 → ERd32 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| | DEC.L #2, ERd | L | | 2 | | | | | | | | ERd32–2 → ERd32 | — | — | ↕ | ↕ | ↕ | — | 2 | |
| DAS | DAS.Rd | B | | 2 | | | | | | | | Rd8 decimal adjust → Rd8 | — | * | ↕ | ↕ | * | — | 2 | |
| MULXU | MULXU. B Rs, Rd | B | | 2 | | | | | | | | Rd8 × Rs8 → Rd16 (unsigned multiplication) | — | — | — | — | — | — | 14 | |
| | MULXU. W Rs, ERd | W | | 2 | | | | | | | | Rd16 × Rs16 → ERd32 (unsigned multiplication) | — | — | — | — | — | — | 22 | |
| MULXS | MULXS. B Rs, Rd | B | | 4 | | | | | | | | Rd8 × Rs8 → Rd16 (signed multiplication) | — | — | ↕ | ↕ | — | — | 16 | |
| | MULXS. W Rs, ERd | W | | 4 | | | | | | | | Rd16 × Rs16 → ERd32 (signed multiplication) | — | — | ↕ | ↕ | — | — | 24 | |
| DIVXU | DIVXU. B Rs, Rd | B | | 2 | | | | | | | | Rd16 ÷ Rs8 → Rd16 (RdH: remainder, RdL: quotient) (unsigned division) | — | — | (6) | (7) | — | — | 14 | |
| | DIVXU. W Rs, ERd | W | | 2 | | | | | | | | ERd32 ÷ Rs16 → ERd32 (Ed: remainder, Rd: quotient) (unsigned division) | — | — | (6) | (7) | — | — | 22 | |
| DIVXS | DIVXS. B Rs, Rd | B | | 4 | | | | | | | | Rd16 ÷ Rs8 → Rd16 (RdH: remainder, RdL: quotient) (signed division) | — | — | (8) | (7) | — | — | 16 | |
| | DIVXS. W Rs, ERd | W | | 4 | | | | | | | | ERd32 ÷ Rs16 → ERd32 (Ed: remainder, Rd: quotient) (signed division) | — | — | (8) | (7) | — | — | 24 | |
| CMP | CMP.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8–#xx:8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | CMP.B Rs, Rd | B | | 2 | | | | | | | | Rd8–Rs8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | CMP.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16–#xx:16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 4 | |
| | CMP.W Rs, Rd | W | | 2 | | | | | | | | Rd16–Rs16 | — | (1) | ↕ | ↕ | ↕ | ↕ | 2 | |
| | CMP.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32–#xx:32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 4 | |
| | CMP.L ERs, ERd | L | | 2 | | | | | | | | ERd32–ERs32 | — | (2) | ↕ | ↕ | ↕ | ↕ | 2 | |

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | \| | | I | H | N | Z | V | C | Normal | Advanced |
| NEG | NEG.B Rd | B | | 2 | | | | | | | | 0–Rd8 → Rd8 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | NEG.W Rd | W | | 2 | | | | | | | | 0–Rd16 → Rd16 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | NEG.L ERd | L | | 2 | | | | | | | | 0–ERd32 → ERd32 | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| EXTU | EXTU.W Rd | W | | 2 | | | | | | | | 0 → (<bits 15 to 8> of Rd16) | — | — | 0 | ↕ | 0 | — | 2 | |
| | EXTU.L ERd | L | | 2 | | | | | | | | 0 → (<bits 31 to 16> of ERd32) | — | — | 0 | ↕ | 0 | — | 2 | |
| EXTS | EXTS.W Rd | W | | 2 | | | | | | | | (<bit 7> of Rd16) → (<bits 15 to 8> of Rd16) | — | — | ↕ | ↕ | 0 | — | 2 | |
| | EXTS.L ERd | L | | 2 | | | | | | | | (<bit 15> of ERd32) → (<bits 31 to 16> of ERd32) | — | — | ↕ | ↕ | 0 | — | 2 | |

## 3. Logic Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| AND | AND.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8∧#xx:8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | AND.B Rs, Rd | B | | 2 | | | | | | | | Rd8∧Rs8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | AND.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16∧#xx:16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | AND.W Rs, Rd | W | | 2 | | | | | | | | Rd16∧Rs16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | AND.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32∧#xx:32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | AND.L ERs, ERd | L | | 4 | | | | | | | | ERd32∧ERs32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 4 | |
| OR | OR.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8∨#xx:8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | OR.B Rs, Rd | B | | 2 | | | | | | | | Rd8∨Rs8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | OR.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16∨#xx:16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | OR.W Rs, Rd | W | | 2 | | | | | | | | Rd16∨Rs16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | OR.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32∨#xx:32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | OR.L ERs, ERd | L | | 4 | | | | | | | | ERd32∨ERs32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 4 | |
| XOR | XOR.B #xx:8, Rd | B | 2 | | | | | | | | | Rd8⊕#xx:8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | XOR.B Rs, Rd | B | | 2 | | | | | | | | Rd8⊕Rs8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | XOR.W #xx:16, Rd | W | 4 | | | | | | | | | Rd16⊕#xx:16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 4 | |
| | XOR.W Rs, Rd | W | | 2 | | | | | | | | Rd16⊕Rs16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | XOR.L #xx:32, ERd | L | 6 | | | | | | | | | ERd32⊕#xx:32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 6 | |
| | XOR.L ERs, ERd | L | | 4 | | | | | | | | ERd32⊕ERs32 → ERd32 | — | — | ↕ | ↕ | 0 | — | 4 | |
| NOT | NOT.B Rd | B | | 2 | | | | | | | | ¬ Rd8 → Rd8 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | NOT.W Rd | W | | 2 | | | | | | | | ¬ Rd16 → Rd16 | — | — | ↕ | ↕ | 0 | — | 2 | |
| | NOT.L ERd | L | | 2 | | | | | | | | ¬ Rd32 → Rd32 | — | — | ↕ | ↕ | 0 | — | 2 | |

## 4. Shift Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| SHAL | SHAL.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | ↕ | ↕ | 2 | |
| | SHAL.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | ↕ | ↕ | 2 | |
| | SHAL.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | ↕ | ↕ | 2 | |
| SHAR | SHAR.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHAR.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHAR.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| SHLL | SHLL.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHLL.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHLL.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| SHLR | SHLR.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHLR.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | SHLR.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| ROTXL | ROTXL.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTXL.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTXL.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| ROTXR | ROTXR.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTXR.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTXR.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| ROTL | ROTL.B Rd | B | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTL.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTL.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| ROTR | ROTR.B Rd | B | | 2 | | | | | | | |  | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTR.W Rd | W | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |
| | ROTR.L ERd | L | | 2 | | | | | | | | | — | — | ↕ | ↕ | 0 | ↕ | 2 | |

## 5. Bit-Manipulation Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | \| | | I | H | N | Z | V | C | Normal | Advanced |
| BSET | BSET #xx:3, Rd | B | | 2 | | | | | | | | (#xx:3 of Rd8) ← 1 | — | — | — | — | — | — | 2 | |
| | BSET #xx:3, @ERd | B | | | 4 | | | | | | | (#xx:3 of @ERd) ← 1 | — | — | — | — | — | — | 8 | |
| | BSET #xx:3, @aa:8 | B | | | | | | 4 | | | | (#xx:3 of @aa:8) ← 1 | — | — | — | — | — | — | 8 | |
| | BSET Rn, Rd | B | | 2 | | | | | | | | (Rn8 of Rd8) ← 1 | — | — | — | — | — | — | 2 | |
| | BSET Rn, @ERd | B | | | 4 | | | | | | | (Rn8 of @ERd) ← 1 | — | — | — | — | — | — | 8 | |
| | BSET Rn, @aa:8 | B | | | | | | 4 | | | | (Rn8 of @aa:8) ← 1 | — | — | — | — | — | — | 8 | |
| BCLR | BCLR #xx:3, Rd | B | | 2 | | | | | | | | (#xx:3 of Rd8) ← 0 | — | — | — | — | — | — | 2 | |
| | BCLR #xx:3, @ERd | B | | | 4 | | | | | | | (#xx:3 of @ERd) ← 0 | — | — | — | — | — | — | 8 | |
| | BCLR #xx:3, @aa:8 | B | | | | | | 4 | | | | (#xx:3 of @aa:8) ← 0 | — | — | — | — | — | — | 8 | |
| | BCLR Rn, Rd | B | | 2 | | | | | | | | (Rn8 of Rd8) ← 0 | — | — | — | — | — | — | 2 | |
| | BCLR Rn, @ERd | B | | | 4 | | | | | | | (Rn8 of @ERd) ← 0 | — | — | — | — | — | — | 8 | |
| | BCLR Rn, @aa:8 | B | | | | | | 4 | | | | (Rn8 of @aa:8) ← 0 | — | — | — | — | — | — | 8 | |
| BNOT | BNOT #xx:3, Rd | B | | 2 | | | | | | | | (#xx:3 of Rd8) ← ¬ (#xx:3 of Rd8) | — | — | — | — | — | — | 2 | |
| | BNOT #xx:3, @ERd | B | | | 4 | | | | | | | (#xx:3 of @ERd) ← ¬ (#xx:3 of @ERd) | — | — | — | — | — | — | 8 | |
| | BNOT #xx:3, @aa:8 | B | | | | | | 4 | | | | (#xx:3 of @aa:8) ← ¬ (#xx:3 of @aa:8) | — | — | — | — | — | — | 8 | |
| | BNOT Rn, Rd | B | | 2 | | | | | | | | (Rn8 of Rd8) ← ¬ (Rn8 of Rd8) | — | — | — | — | — | — | 2 | |
| | BNOT Rn, @ERd | B | | | 4 | | | | | | | (Rn8 of @ERd) ← ¬ (Rn8 of @ERd) | — | — | — | — | — | — | 8 | |
| | BNOT Rn, @aa:8 | B | | | | | | 4 | | | | (Rn8 of @aa:8) ← ¬ (Rn8 of @aa:8) | — | — | — | — | — | — | 8 | |
| BTST | BTST #xx:3, Rd | B | | 2 | | | | | | | | ¬ (#xx:3 of Rd8) → Z | — | — | — | ↕ | — | — | 2 | |
| | BTST #xx:3, @ERd | B | | | 4 | | | | | | | ¬ (#xx:3 of @ERd) → Z | — | — | — | ↕ | — | — | 6 | |
| | BTST #xx:3, @aa:8 | B | | | | | | 4 | | | | ¬ (#xx:3 of @aa:8) → Z | — | — | — | ↕ | — | — | 6 | |
| | BTST Rn, Rd | B | | 2 | | | | | | | | ¬ (Rn8 of @Rd8) → Z | — | — | — | ↕ | — | — | 2 | |
| | BTST Rn, @ERd | B | | | 4 | | | | | | | ¬ (Rn8 of @ERd) → Z | — | — | — | ↕ | — | — | 6 | |
| | BTST Rn, @aa:8 | B | | | | | | 4 | | | | ¬ (Rn8 of @aa:8) → Z | — | — | — | ↕ | — | — | 6 | |
| BLD | BLD #xx:3, Rd | B | | 2 | | | | | | | | (#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | \| | | I | H | N | Z | V | C | Normal | Advanced |
| BLD | BLD #xx:3, @ERd | B | | | 4 | | | | | | | (#xx:3 of @ERd) → C | — | — | — | — | — | ↕ | 6 | |
| | BLD #xx:3, @aa:8 | B | | | | | | 4 | | | | (#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BILD | BILD #xx:3, Rd | B | | 2 | | | | | | | | ¬ (#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BILD #xx:3, @ERd | B | | | 4 | | | | | | | ¬ (#xx:3 of @ERd) → C | — | — | — | — | — | ↕ | 6 | |
| | BILD #xx:3, @aa:8 | B | | | | | | 4 | | | | ¬ (#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BST | BST #xx:3, Rd | B | | 2 | | | | | | | | C → (#xx:3 of Rd8) | — | — | — | — | — | — | 2 | |
| | BST #xx:3, @ERd | B | | | 4 | | | | | | | C → (#xx:3 of @ERd24) | — | — | — | — | — | — | 8 | |
| | BST #xx:3, @aa:8 | B | | | | | | 4 | | | | C → (#xx:3 of @aa:8) | — | — | — | — | — | — | 8 | |
| BIST | BIST #xx:3, Rd | B | | 2 | | | | | | | | ¬ C → (#xx:3 of Rd8) | — | — | — | — | — | — | 2 | |
| | BIST #xx:3, @ERd | B | | | 4 | | | | | | | ¬ C → (#xx:3 of @ERd24) | — | — | — | — | — | — | 8 | |
| | BIST #xx:3, @aa:8 | B | | | | | | 4 | | | | ¬ C → (#xx:3 of @aa:8) | — | — | — | — | — | — | 8 | |
| BAND | BAND #xx:3, Rd | B | | 2 | | | | | | | | C∧(#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BAND #xx:3, @ERd | B | | | 4 | | | | | | | C∧(#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BAND #xx:3, @aa:8 | B | | | | | | 4 | | | | C∧(#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BIAND | BIAND #xx:3, Rd | B | | 2 | | | | | | | | C∧ ¬ (#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BIAND #xx:3, @ERd | B | | | 4 | | | | | | | C∧ ¬ (#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BIAND #xx:3, @aa:8 | B | | | | | | 4 | | | | C∧ ¬ (#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BOR | BOR #xx:3, Rd | B | | 2 | | | | | | | | C∨(#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BOR #xx:3, @ERd | B | | | 4 | | | | | | | C∨(#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BOR #xx:3, @aa:8 | B | | | | | | 4 | | | | C∨(#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BIOR | BIOR #xx:3, Rd | B | | 2 | | | | | | | | C∨ ¬ (#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BIOR #xx:3, @ERd | B | | | 4 | | | | | | | C∨ ¬ (#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BIOR #xx:3, @aa:8 | B | | | | | | 4 | | | | C∨ ¬ (#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BXOR | BXOR #xx:3, Rd | B | | 2 | | | | | | | | C⊕(#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BXOR #xx:3, @ERd | B | | | 4 | | | | | | | C⊕(#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BXOR #xx:3, @aa:8 | B | | | | | | 4 | | | | C⊕(#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| BIXOR | BIXOR #xx:3, Rd | B | | 2 | | | | | | | | C⊕ ¬ (#xx:3 of Rd8) → C | — | — | — | — | — | ↕ | 2 | |
| | BIXOR #xx:3, @ERd | B | | | 4 | | | | | | | C⊕ ¬ (#xx:3 of @ERd24) → C | — | — | — | — | — | ↕ | 6 | |
| | BIXOR #xx:3, @aa:8 | B | | | | | | 4 | | | | C⊕ ¬ (#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |

## 6. Branching Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | \| | Operation | Branch Condition | Condition Code I | H | N | Z | V | C | No. of States*1 Normal | Advanced |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bcc | BRA d:8 (BT d:8) | — | | | | | | | 2 | | | If condition is true then PC ← PC+d else next; | Always | — | — | — | — | — | — | 4 | |
| | BRA d:16 (BT d:16) | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BRN d:8 (BF d:8) | — | | | | | | | 2 | | | | Never | — | — | — | — | — | — | 4 | |
| | BRN d:16 (BF d:16) | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BHI d:8 | — | | | | | | | 2 | | | | C∨ Z = 0 | — | — | — | — | — | — | 4 | |
| | BHI d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BLS d:8 | — | | | | | | | 2 | | | | C∨ Z = 1 | — | — | — | — | — | — | 4 | |
| | BLS d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BCC d:8 (BHS d:8) | — | | | | | | | 2 | | | | C = 0 | — | — | — | — | — | — | 4 | |
| | BCC d:16 (BHS d:16) | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BCS d:8 (BLO d:8) | — | | | | | | | 2 | | | | C = 1 | — | — | — | — | — | — | 4 | |
| | BCS d:16 (BLO d:16) | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BNE d:8 | — | | | | | | | 2 | | | | Z = 0 | — | — | — | — | — | — | 4 | |
| | BNE d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BEQ d:8 | — | | | | | | | 2 | | | | Z = 1 | — | — | — | — | — | — | 4 | |
| | BEQ d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BVC d:8 | — | | | | | | | 2 | | | | V = 0 | — | — | — | — | — | — | 4 | |
| | BVC d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BVS d:8 | — | | | | | | | 2 | | | | V = 1 | — | — | — | — | — | — | 4 | |
| | BVS d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BPL d:8 | — | | | | | | | 2 | | | | N = 0 | — | — | — | — | — | — | 4 | |
| | BPL d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BMI d:8 | — | | | | | | | 2 | | | | N = 1 | — | — | — | — | — | — | 4 | |
| | BMI d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BGE d:8 | — | | | | | | | 2 | | | | N⊕V = 0 | — | — | — | — | — | — | 4 | |
| | BGE d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BLT d:8 | — | | | | | | | 2 | | | | N⊕V = 1 | — | — | — | — | — | — | 4 | |
| | BLT d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BGT d:8 | — | | | | | | | 2 | | | | Z∨ (N⊕V) = 0 | — | — | — | — | — | — | 4 | |
| | BGT d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |
| | BLE d:8 | — | | | | | | | 2 | | | | Z∨ (N⊕V) = 1 | — | — | — | — | — | — | 4 | |
| | BLE d:16 | — | | | | | | | 4 | | | | | — | — | — | — | — | — | 6 | |

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| JMP | JMP @ERn | — | | | 2 | | | | | | | PC ← ERn | — | — | — | — | — | — | 4 | |
| | JMP @aa:24 | — | | | | | | 4 | | | | PC ← aa:24 | — | — | — | — | — | — | 6 | |
| | JMP @@aa:8 | — | | | | | | | | 2 | | PC ← @aa:8 | — | — | — | — | — | — | 8 | 10 |
| BSR | BSR d:8 | — | | | | | | | 2 | | | PC → @–SP<br>PC ← PC+d:8 | — | — | — | — | — | — | 6 | 8 |
| | BSR d:16 | — | | | | | | | 4 | | | PC → @–SP<br>PC ← PC+d:16 | — | — | — | — | — | — | 8 | 10 |
| JSR | JSR @ERn | — | | | 2 | | | | | | | PC → @–SP<br>PC ← ERn | — | — | — | — | — | — | 6 | 8 |
| | JSR @aa:24 | — | | | | | | 4 | | | | PC → @–SP<br>PC ← aa:24 | — | — | — | — | — | — | 8 | 10 |
| | JSR @@aa:8 | — | | | | | | | | 2 | | PC → @–SP<br>PC ← @aa:8 | — | — | — | — | — | — | 8 | 12 |
| RTS | RTS | — | | | | | | | | | 2 | PC ← @SP+ | — | — | — | — | — | — | 8 | 10 |

# 7. System Control Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | I | H | N | Z | V | C | Normal | Advanced |
| RTE | RTE | — | | | | | | | | | | CCR ← @SP+<br>PC ← @SP+ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 10 | |
| SLEEP | SLEEP | — | | | | | | | | | | Transition to power-down state | | | | | | | 2 | |
| LDC | LDC #xx:8, CCR | B | 2 | | | | | | | | | #xx:8 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | LDC Rs, CCR | B | | 2 | | | | | | | | Rs8 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | LDC @ERs, CCR | W | | | 4 | | | | | | | @ERs → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 6 | |
| | LDC @(d:16, ERs), CCR | W | | | | 6 | | | | | | @(d:16, ERs) → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 8 | |
| | LDC @(d:24, ERs), CCR | W | | | | 10 | | | | | | @(d:24, ERs) → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 12 | |
| | LDC @ERs+, CCR | W | | | | | 4 | | | | | @ERs → CCR<br>ERs32+2 → ERs32 | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 8 | |
| | LDC @aa:16, CCR | W | | | | | | 6 | | | | @aa:16 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 8 | |
| | LDC @aa:24, CCR | W | | | | | | 8 | | | | @aa:24 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 10 | |
| STC | STC CCR, Rd | B | | 2 | | | | | | | | CCR → Rd8 | | | | | | | 2 | |
| | STC CCR, @ERd | W | | | 4 | | | | | | | CCR → @ERd | | | | | | | 6 | |
| | STC CCR, @(d:16, ERd) | W | | | | 6 | | | | | | CCR → @(d:16, ERd) | | | | | | | 8 | |
| | STC CCR, @(d:24, ERd) | W | | | | 10 | | | | | | CCR → @(d:24, ERd) | | | | | | | 12 | |
| | STC CCR, @–ERd | W | | | | | 4 | | | | | ERd32–2 → ERd32<br>CCR → @ERd | | | | | | | 8 | |
| | STC CCR, @aa:16 | W | | | | | | 6 | | | | CCR → @aa:16 | | | | | | | 8 | |
| | STC CCR, @aa:24 | W | | | | | | 8 | | | | CCR → @aa:24 | | | | | | | 10 | |
| ANDC | ANDC #xx:8, CCR | B | 2 | | | | | | | | | CCR∧#xx:8 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| ORC | ORC #xx:8, CCR | B | 2 | | | | | | | | | CCR∨#xx:8 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| XORC | XORC #xx:8, CCR | B | 2 | | | | | | | | | CCR⊕#xx:8 → CCR | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| NOP | NOP | — | | | | | | | | | 2 | PC ← PC+2 | | | | | | | 2 | |

## 8. Block Transfer Instructions

| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | \| | | I | H | N | Z | V | C | Normal | Advanced |
| EEPMOV | EEPMOV. B | — | | | | | | | | | 4 | if R4L ≠ 0 then<br>repeat @R5 → @R6<br>R5+1 → R5<br>R6+1 → R6<br>R4L–1 → R4L<br>until R4L=0<br>else next | — | — | — | — | — | — | 8+<br>4n*2 | |
| | EEPMOV. W | — | | | | | | | | | 4 | if R4 ≠ 0 then<br>repeat @R5 → @R6<br>R5+1 → R5<br>R6+1 → R6<br>R4–1 → R4<br>until R4=0<br>else next | — | — | — | — | — | — | 8+<br>4n*2 | |

Notes: 1. The number of states in cases where the instruction code and its operands are located in on-chip memory is shown here. For other cases, see appendix A.3, Number of Execution States.

2. n is the value set in register R4L or R4.

(1) Set to 1 when a carry or borrow occurs at bit 11; otherwise cleared to 0.

(2) Set to 1 when a carry or borrow occurs at bit 27; otherwise cleared to 0.

(3) Retains its previous value when the result is zero; otherwise cleared to 0.

(4) Set to 1 when the adjustment produces a carry; otherwise retains its previous value.

(5) The number of states required for execution of an instruction that transfers data in synchronization with the E clock is variable.

(6) Set to 1 when the divisor is negative; otherwise cleared to 0.

(7) Set to 1 when the divisor is zero; otherwise cleared to 0.

(8) Set to 1 when the quotient is negative; otherwise cleared to 0.

## A.2 Operation Code Map

### Table A.2 Operation Code Map (1)

Instruction code:

| 1st byte | | 2nd byte | |
|---|---|---|---|
| AH | AL | BH | BL |

Instruction when most significant bit of BH is 0.

Instruction when most significant bit of BH is 1.

| AH \ AL | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NOP | Table A.2 (2) | STC | LDC | ORC | XORC | ANDC | LDC | ADD | | Table A.2 (2) | Table A.2 (2) | MOV | | ADDX | Table A.2 (2) |
| 1 | Table A.2 (2) | Table A.2 (2) | Table A.2 (2) | Table A.2 (2) | OR.B | XOR.B | AND.B | Table A.2 (2) | SUB | | Table A.2 (2) | Table A.2 (2) | CMP | | SUBX | Table A.2 (2) |
| 2 | MOV.B | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | |
| 4 | BRA | BRN | BHI | BLS | BCC | BCS | BNE | BEQ | BVC | BVS | BPL | BMI | BGE | BLT | BGT | BLE |
| 5 | MULXU | DIVXU | MULXU | DIVXU | RTS | BSR | RTE | | Table A.2 (2) | JMP | | | BSR | JSR | | |
| 6 | BSET | BNOT | BCLR | BTST | OR | XOR | AND | BST / BIST | MOV | MOV | | | | | | |
| 7 | | | | | BOR / BIOR | BXOR / BIXOR | BAND / BIAND | BLD / BILD | | Table A.2 (2) | Table A.2 (2) | EEPMOV | Table A.2 (3) | | | |
| 8 | ADD | | | | | | | | | | | | | | | |
| 9 | ADDX | | | | | | | | | | | | | | | |
| A | CMP | | | | | | | | | | | | | | | |
| B | SUBX | | | | | | | | | | | | | | | |
| C | OR | | | | | | | | | | | | | | | |
| D | XOR | | | | | | | | | | | | | | | |
| E | AND | | | | | | | | | | | | | | | |
| F | MOV | | | | | | | | | | | | | | | |

## Table A.2 Operation Code Map (2)

Instruction code:

| 1st byte | | 2nd byte | |
|---|---|---|---|
| AH | AL | BH | BL |

| AH AL \ BH | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | MOV | | | | LDC/STC | | | | SLEEP | | | | Table A-2 (3) | Table A-2 (3) | | Table A-2 (3) |
| 0A | INC | | | | | | | | ADD | | | | | | | |
| 0B | ADDS | | | | | INC | | INC | ADDS | | | | | INC | | INC |
| 0F | DAA | | | | | | | | MOV | | | | | | | |
| 10 | SHLL | | | SHLL | | | | | SHAL | | | SHAL | | | | |
| 11 | SHLR | | | SHLR | | | | | SHAR | | | SHAR | | | | |
| 12 | ROTXL | | | ROTXL | | | | | ROTL | | | ROTL | | | | |
| 13 | ROTXR | | | ROTXR | | | | | ROTR | | | ROTR | | | | |
| 17 | NOT | | | NOT | | EXTU | | EXTU | NEG | | | NEG | | EXTS | | EXTS |
| 1A | DEC | | | | | | | | SUB | | | | | | | |
| 1B | SUBS | | | | | DEC | | DEC | SUB | | | | | DEC | | DEC |
| 1F | DAS | | | | | | | | CMP | | | | | | | |
| 58 | BRA | BRN | BHI | BLS | BCC | BCS | BNE | BEQ | BVC | BVS | BPL | BMI | BGE | BLT | BGT | BLE |
| 79 | MOV | ADD | CMP | SUB | OR | XOR | AND | | | | | | | | | |
| 7A | MOV | ADD | CMP | SUB | OR | XOR | AND | | | | | | | | | |

## Table A.2 Operation Code Map (3)

Instruction code:

| 1st byte | 2nd byte | 3rd byte | 4th byte |
|---|---|---|---|
| AH AL | BH BL | CH CL | DH DL |

Instruction when most significant bit of DH is 0.

Instruction when most significant bit of DH is 1.

| CL / AH ALBH BLCH | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01406 | | | | | | | | | | LDC / STC | | LDC / STC | | LDC / STC | | LDC / STC |
| 01C05 | MULXS | | MULXS | | | | | | | | | | | | | |
| 01D05 | | DIVXS | | DIVXS | | | | | | | | | | | | |
| 01F06 | | | | | OR | XOR | AND | | | | | | | | | |
| 7Cr06*1 | | | | BTST | | | | | | | | | | | | |
| 7Cr07*1 | | | | BTST | BOR / BIOR | BXOR / BIXOR | BAND / BIAND | BLD / BILD | | | | | | | | |
| 7Dr06*1 | BSET | BNOT | BCLR | | | | | BST / BIST | | | | | | | | |
| 7Dr07*1 | BSET | BNOT | BCLR | | | | | | | | | | | | | |
| 7Eaa6*2 | | | | BTST | | | | | | | | | | | | |
| 7Eaa7*2 | | | | BTST | BOR / BIOR | BXOR / BIXOR | BAND / BIAND | BLD / BILD | | | | | | | | |
| 7Faa6*2 | BSET | BNOT | BCLR | | | | | BST / BIST | | | | | | | | |
| 7Faa7*2 | BSET | BNOT | BCLR | | | | | | | | | | | | | |

Notes: 1. r is the register designation field.
2. aa is the absolute address field.

## A.3 Number of Execution States

The status of execution for each instruction of the H8/300H CPU and the method of calculating the number of states required for instruction execution are shown below. Table A.4 shows the number of cycles of each type occurring in each instruction, such as instruction fetch and data read/write. Table A.3 shows the number of states required for each cycle. The total number of states required for execution of an instruction can be calculated by the following expression:

$$\text{Execution states} = I \times S_I + J \times S_J + K \times S_K + L \times S_L + M \times S_M + N \times S_N$$

**Examples:** When instruction is fetched from on-chip ROM, and an on-chip RAM is accessed.

BSET #0, @FF00

From table A.4:

$I = L = 2, \quad J = K = M = N = 0$

From table A.3:

$S_I = 2, \quad S_L = 2$

Number of states required for execution = $2 \times 2 + 2 \times 2 = 8$

When instruction is fetched from on-chip ROM, branch address is read from on-chip ROM, and on-chip RAM is used for stack area.

JSR @@ 30

From table A.4:

$I = 2, \quad J = K = 1, \quad L = M = N = 0$

From table A.3:

$S_I = S_J = S_K = 2$

Number of states required for execution = $2 \times 2 + 1 \times 2 + 1 \times 2 = 8$

## Table A.3 Number of Cycles in Each Instruction

| Execution Status (Instruction Cycle) | | Access Location | |
|---|---|---|---|
| | | On-Chip Memory | On-Chip Peripheral Module |
| Instruction fetch | $S_I$ | 2 | — |
| Branch address read | $S_J$ | | |
| Stack operation | $S_K$ | | |
| Byte data access | $S_L$ | | 2 or 3* |
| Word data access | $S_M$ | | — |
| Internal operation | $S_N$ | 1 | |

Note: * Depends on which on-chip peripheral module is accessed. See section 15.1, Register Addresses (Address Order).

## Table A.4 Number of Cycles in Each Instruction

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| ADD | ADD.B #xx:8, Rd | 1 | | | | | |
| | ADD.B Rs, Rd | 1 | | | | | |
| | ADD.W #xx:16, Rd | 2 | | | | | |
| | ADD.W Rs, Rd | 1 | | | | | |
| | ADD.L #xx:32, ERd | 3 | | | | | |
| | ADD.L ERs, ERd | 1 | | | | | |
| ADDS | ADDS #1/2/4, ERd | 1 | | | | | |
| ADDX | ADDX #xx:8, Rd | 1 | | | | | |
| | ADDX Rs, Rd | 1 | | | | | |
| AND | AND.B #xx:8, Rd | 1 | | | | | |
| | AND.B Rs, Rd | 1 | | | | | |
| | AND.W #xx:16, Rd | 2 | | | | | |
| | AND.W Rs, Rd | 1 | | | | | |
| | AND.L #xx:32, ERd | 3 | | | | | |
| | AND.L ERs, ERd | 2 | | | | | |
| ANDC | ANDC #xx:8, CCR | 1 | | | | | |
| BAND | BAND #xx:3, Rd | 1 | | | | | |
| | BAND #xx:3, @ERd | 2 | | | 1 | | |
| | BAND #xx:3, @aa:8 | 2 | | | 1 | | |
| Bcc | BRA d:8 (BT d:8) | 2 | | | | | |
| | BRN d:8 (BF d:8) | 2 | | | | | |
| | BHI d:8 | 2 | | | | | |
| | BLS d:8 | 2 | | | | | |
| | BCC d:8 (BHS d:8) | 2 | | | | | |
| | BCS d:8 (BLO d:8) | 2 | | | | | |
| | BNE d:8 | 2 | | | | | |
| | BEQ d:8 | 2 | | | | | |
| | BVC d:8 | 2 | | | | | |
| | BVS d:8 | 2 | | | | | |
| | BPL d:8 | 2 | | | | | |
| | BMI d:8 | 2 | | | | | |
| | BGE d:8 | 2 | | | | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| Bcc | BLT d:8 | 2 | | | | | |
| | BGT d:8 | 2 | | | | | |
| | BLE d:8 | 2 | | | | | |
| | BRA d:16(BT d:16) | 2 | | | | | 2 |
| | BRN d:16(BF d:16) | 2 | | | | | 2 |
| | BHI d:16 | 2 | | | | | 2 |
| | BLS d:16 | 2 | | | | | 2 |
| | BCC d:16(BHS d:16) | 2 | | | | | 2 |
| | BCS d:16(BLO d:16) | 2 | | | | | 2 |
| | BNE d:16 | 2 | | | | | 2 |
| | BEQ d:16 | 2 | | | | | 2 |
| | BVC d:16 | 2 | | | | | 2 |
| | BVS d:16 | 2 | | | | | 2 |
| | BPL d:16 | 2 | | | | | 2 |
| | BMI d:16 | 2 | | | | | 2 |
| | BGE d:16 | 2 | | | | | 2 |
| | BLT d:16 | 2 | | | | | 2 |
| | BGT d:16 | 2 | | | | | 2 |
| | BLE d:16 | 2 | | | | | 2 |
| BCLR | BCLR #xx:3, Rd | 1 | | | | | |
| | BCLR #xx:3, @ERd | 2 | | | 2 | | |
| | BCLR #xx:3, @aa:8 | 2 | | | 2 | | |
| | BCLR Rn, Rd | 1 | | | | | |
| | BCLR Rn, @ERd | 2 | | | 2 | | |
| | BCLR Rn, @aa:8 | 2 | | | 2 | | |
| BIAND | BIAND #xx:3, Rd | 1 | | | | | |
| | BIAND #xx:3, @ERd | 2 | | | 1 | | |
| | BIAND #xx:3, @aa:8 | 2 | | | 1 | | |
| BILD | BILD #xx:3, Rd | 1 | | | | | |
| | BILD #xx:3, @ERd | 2 | | | 1 | | |
| | BILD #xx:3, @aa:8 | 2 | | | 1 | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| BIOR | BIOR #xx:3, Rd | 1 | | | | | |
| | BIOR #xx:3, @ERd | 2 | | | 1 | | |
| | BIOR #xx:3, @aa:8 | 2 | | | 1 | | |
| BIST | BIST #xx:3, Rd | 1 | | | | | |
| | BIST #xx:3, @ERd | 2 | | | 2 | | |
| | BIST #xx:3, @aa:8 | 2 | | | 2 | | |
| BIXOR | BIXOR #xx:3, Rd | 1 | | | | | |
| | BIXOR #xx:3, @ERd | 2 | | | 1 | | |
| | BIXOR #xx:3, @aa:8 | 2 | | | 1 | | |
| BLD | BLD #xx:3, Rd | 1 | | | | | |
| | BLD #xx:3, @ERd | 2 | | | 1 | | |
| | BLD #xx:3, @aa:8 | 2 | | | 1 | | |
| BNOT | BNOT #xx:3, Rd | 1 | | | | | |
| | BNOT #xx:3, @ERd | 2 | | | 2 | | |
| | BNOT #xx:3, @aa:8 | 2 | | | 2 | | |
| | BNOT Rn, Rd | 1 | | | | | |
| | BNOT Rn, @ERd | 2 | | | 2 | | |
| | BNOT Rn, @aa:8 | 2 | | | 2 | | |
| BOR | BOR #xx:3, Rd | 1 | | | | | |
| | BOR #xx:3, @ERd | 2 | | | 1 | | |
| | BOR #xx:3, @aa:8 | 2 | | | 1 | | |
| BSET | BSET #xx:3, Rd | 1 | | | | | |
| | BSET #xx:3, @ERd | 2 | | | 2 | | |
| | BSET #xx:3, @aa:8 | 2 | | | 2 | | |
| | BSET Rn, Rd | 1 | | | | | |
| | BSET Rn, @ERd | 2 | | | 2 | | |
| | BSET Rn, @aa:8 | 2 | | | 2 | | |
| BSR | BSR d:8 | 2 | | 1 | | | |
| | BSR d:16 | 2 | | 1 | | | 2 |
| BST | BST #xx:3, Rd | 1 | | | | | |
| | BST #xx:3, @ERd | 2 | | | 2 | | |
| | BST #xx:3, @aa:8 | 2 | | | 2 | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| BTST | BTST #xx:3, Rd | 1 | | | | | |
| | BTST #xx:3, @ERd | 2 | | | 1 | | |
| | BTST #xx:3, @aa:8 | 2 | | | 1 | | |
| | BTST Rn, Rd | 1 | | | | | |
| | BTST Rn, @ERd | 2 | | | 1 | | |
| | BTST Rn, @aa:8 | 2 | | | 1 | | |
| BXOR | BXOR #xx:3, Rd | 1 | | | | | |
| | BXOR #xx:3, @ERd | 2 | | | 1 | | |
| | BXOR #xx:3, @aa:8 | 2 | | | 1 | | |
| CMP | CMP.B #xx:8, Rd | 1 | | | | | |
| | CMP.B Rs, Rd | 1 | | | | | |
| | CMP.W #xx:16, Rd | 2 | | | | | |
| | CMP.W Rs, Rd | 1 | | | | | |
| | CMP.L #xx:32, ERd | 3 | | | | | |
| | CMP.L ERs, ERd | 1 | | | | | |
| DAA | DAA Rd | 1 | | | | | |
| DAS | DAS Rd | 1 | | | | | |
| DEC | DEC.B Rd | 1 | | | | | |
| | DEC.W #1/2, Rd | 1 | | | | | |
| | DEC.L #1/2, ERd | 1 | | | | | |
| DUVXS | DIVXS.B Rs, Rd | 2 | | | | | 12 |
| | DIVXS.W Rs, ERd | 2 | | | | | 20 |
| DIVXU | DIVXU.B Rs, Rd | 1 | | | | | 12 |
| | DIVXU.W Rs, ERd | 1 | | | | | 20 |
| EEPMOV | EEPMOV.B | 2 | | | 2n+2*[1] | | |
| | EEPMOV.W | 2 | | | 2n+2*[1] | | |
| EXTS | EXTS.W Rd | 1 | | | | | |
| | EXTS.L ERd | 1 | | | | | |
| EXTU | EXTU.W Rd | 1 | | | | | |
| | EXTU.L ERd | 1 | | | | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| INC | INC.B Rd | 1 | | | | | |
| | INC.W #1/2, Rd | 1 | | | | | |
| | INC.L #1/2, ERd | 1 | | | | | |
| JMP | JMP @ERn | 2 | | | | | |
| | JMP @aa:24 | 2 | | | | | 2 |
| | JMP @@aa:8 | 2 | 1 | | | | 2 |
| JSR | JSR @ERn | 2 | | 1 | | | |
| | JSR @aa:24 | 2 | | 1 | | | 2 |
| | JSR @@aa:8 | 2 | 1 | 1 | | | |
| LDC | LDC #xx:8, CCR | 1 | | | | | |
| | LDC Rs, CCR | 1 | | | | | |
| | LDC@ERs, CCR | 2 | | | | 1 | |
| | LDC@(d:16, ERs), CCR | 3 | | | | 1 | |
| | LDC@(d:24,ERs), CCR | 5 | | | | 1 | |
| | LDC@ERs+, CCR | 2 | | | | 1 | 2 |
| | LDC@aa:16, CCR | 3 | | | | 1 | |
| | LDC@aa:24, CCR | 4 | | | | 1 | |
| MOV | MOV.B #xx:8, Rd | 1 | | | | | |
| | MOV.B Rs, Rd | 1 | | | | | |
| | MOV.B @ERs, Rd | 1 | | | 1 | | |
| | MOV.B @(d:16, ERs), Rd | 2 | | | 1 | | |
| | MOV.B @(d:24, ERs), Rd | 4 | | | 1 | | |
| | MOV.B @ERs+, Rd | 1 | | | 1 | | 2 |
| | MOV.B @aa:8, Rd | 1 | | | 1 | | |
| | MOV.B @aa:16, Rd | 2 | | | 1 | | |
| | MOV.B @aa:24, Rd | 3 | | | 1 | | |
| | MOV.B Rs, @Erd | 1 | | | 1 | | |
| | MOV.B Rs, @(d:16, ERd) | 2 | | | 1 | | |
| | MOV.B Rs, @(d:24, ERd) | 4 | | | 1 | | |
| | MOV.B Rs, @-ERd | 1 | | | 1 | | 2 |
| | MOV.B Rs, @aa:8 | 1 | | | 1 | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| MOV | MOV.B Rs, @aa:16 | 2 | | | 1 | | |
| | MOV.B Rs, @aa:24 | 3 | | | 1 | | |
| | MOV.W #xx:16, Rd | 2 | | | | | |
| | MOV.W Rs, Rd | 1 | | | | | |
| | MOV.W @ERs, Rd | 1 | | | | 1 | |
| | MOV.W @(d:16,ERs), Rd | 2 | | | | 1 | |
| | MOV.W @(d:24,ERs), Rd | 4 | | | | 1 | |
| | MOV.W @ERs+, Rd | 1 | | | | 1 | 2 |
| | MOV.W @aa:16, Rd | 2 | | | | 1 | |
| | MOV.W @aa:24, Rd | 3 | | | | 1 | |
| | MOV.W Rs, @ERd | 1 | | | | 1 | |
| | MOV.W Rs, @(d:16,ERd) | 2 | | | | 1 | |
| | MOV.W Rs, @(d:24,ERd) | 4 | | | | 1 | |
| MOV | MOV.W Rs, @-ERd | 1 | | | | 1 | 2 |
| | MOV.W Rs, @aa:16 | 2 | | | | 1 | |
| | MOV.W Rs, @aa:24 | 3 | | | | 1 | |
| | MOV.L #xx:32, ERd | 3 | | | | | |
| | MOV.L ERs, ERd | 1 | | | | | |
| | MOV.L @ERs, ERd | 2 | | | | 2 | |
| | MOV.L @(d:16,ERs), ERd | 3 | | | | 2 | |
| | MOV.L @(d:24,ERs), ERd | 5 | | | | 2 | |
| | MOV.L @ERs+, ERd | 2 | | | | 2 | 2 |
| | MOV.L @aa:16, ERd | 3 | | | | 2 | |
| | MOV.L @aa:24, ERd | 4 | | | | 2 | |
| | MOV.L ERs,@ERd | 2 | | | | 2 | |
| | MOV.L ERs, @(d:16,ERd) | 3 | | | | 2 | |
| | MOV.L ERs, @(d:24,ERd) | 5 | | | | 2 | |
| | MOV.L ERs, @-ERd | 2 | | | | 2 | 2 |
| | MOV.L ERs, @aa:16 | 3 | | | | 2 | |
| | MOV.L ERs, @aa:24 | 4 | | | | 2 | |
| MOVFPE | MOVFPE @aa:16, Rd*2 | 2 | | | 1 | | |
| MOVTPE | MOVTPE Rs,@aa:16*2 | 2 | | | 1 | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| MULXS | MULXS.B Rs, Rd | 2 | | | | | 12 |
| | MULXS.W Rs, ERd | 2 | | | | | 20 |
| MULXU | MULXU.B Rs, Rd | 1 | | | | | 12 |
| | MULXU.W Rs, ERd | 1 | | | | | 20 |
| NEG | NEG.B Rd | 1 | | | | | |
| | NEG.W Rd | 1 | | | | | |
| | NEG.L ERd | 1 | | | | | |
| NOP | NOP | 1 | | | | | |
| NOT | NOT.B Rd | 1 | | | | | |
| | NOT.W Rd | 1 | | | | | |
| | NOT.L ERd | 1 | | | | | |
| OR | OR.B #xx:8, Rd | 1 | | | | | |
| | OR.B Rs, Rd | 1 | | | | | |
| | OR.W #xx:16, Rd | 2 | | | | | |
| | OR.W Rs, Rd | 1 | | | | | |
| | OR.L #xx:32, ERd | 3 | | | | | |
| | OR.L ERs, ERd | 2 | | | | | |
| ORC | ORC #xx:8, CCR | 1 | | | | | |
| POP | POP.W Rn | 1 | | | | 1 | 2 |
| | POP.L ERn | 2 | | | | 2 | 2 |
| PUSH | PUSH.W Rn | 1 | | | | 1 | 2 |
| | PUSH.L ERn | 2 | | | | 2 | 2 |
| ROTL | ROTL.B Rd | 1 | | | | | |
| | ROTL.W Rd | 1 | | | | | |
| | ROTL.L ERd | 1 | | | | | |
| ROTR | ROTR.B Rd | 1 | | | | | |
| | ROTR.W Rd | 1 | | | | | |
| | ROTR.L ERd | 1 | | | | | |
| ROTXL | ROTXL.B Rd | 1 | | | | | |
| | ROTXL.W Rd | 1 | | | | | |
| | ROTXL.L ERd | 1 | | | | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| ROTXR | ROTXR.B Rd | 1 | | | | | |
| | ROTXR.W Rd | 1 | | | | | |
| | ROTXR.L ERd | 1 | | | | | |
| RTE | RTE | 2 | | 2 | | | 2 |
| RTS | RTS | 2 | | 1 | | | 2 |
| SHAL | SHAL.B Rd | 1 | | | | | |
| | SHAL.W Rd | 1 | | | | | |
| | SHAL.L ERd | 1 | | | | | |
| SHAR | SHAR.B Rd | 1 | | | | | |
| | SHAR.W Rd | 1 | | | | | |
| | SHAR.L ERd | 1 | | | | | |
| SHLL | SHLL.B Rd | 1 | | | | | |
| | SHLL.W Rd | 1 | | | | | |
| | SHLL.L ERd | 1 | | | | | |
| SHLR | SHLR.B Rd | 1 | | | | | |
| | SHLR.W Rd | 1 | | | | | |
| | SHLR.L ERd | 1 | | | | | |
| SLEEP | SLEEP | 1 | | | | | |
| STC | STC CCR, Rd | 1 | | | | | |
| | STC CCR, @ERd | 2 | | | | 1 | |
| | STC CCR, @(d:16,ERd) | 3 | | | | 1 | |
| | STC CCR, @(d:24,ERd) | 5 | | | | 1 | |
| | STC CCR,@-ERd | 2 | | | | 1 | 2 |
| | STC CCR, @aa:16 | 3 | | | | 1 | |
| | STC CCR, @aa:24 | 4 | | | | 1 | |
| SUB | SUB.B Rs, Rd | 1 | | | | | |
| | SUB.W #xx:16, Rd | 2 | | | | | |
| | SUB.W Rs, Rd | 1 | | | | | |
| | SUB.L #xx:32, ERd | 3 | | | | | |
| | SUB.L ERs, ERd | 1 | | | | | |
| SUBS | SUBS #1/2/4, ERd | 1 | | | | | |

| Instruction | Mnemonic | Instruction Fetch I | Branch Addr. Read J | Stack Operation K | Byte Data Access L | Word Data Access M | Internal Operation N |
|---|---|---|---|---|---|---|---|
| SUBX | SUBX #xx:8, Rd | 1 | | | | | |
| | SUBX. Rs, Rd | 1 | | | | | |
| XOR | XOR.B #xx:8, Rd | 1 | | | | | |
| | XOR.B Rs, Rd | 1 | | | | | |
| | XOR.W #xx:16, Rd | 2 | | | | | |
| | XOR.W Rs, Rd | 1 | | | | | |
| | XOR.L #xx:32, ERd | 3 | | | | | |
| | XOR.L ERs, ERd | 2 | | | | | |
| XORC | XORC #xx:8, CCR | 1 | | | | | |

Notes: 1. n: Specified value in R4L. The source and destination operands are accessed n+1 times respectively.
2. It cannot be used in this LSI.

## A.4 Combinations of Instructions and Addressing Modes

### Table A.5 Combinations of Instructions and Addressing Modes

| Functions | Instructions | Addressing Mode | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #xx | Rn | @ERn | @(d:16.ERn) | @(d:24.ERn) | @ERn+/@ERn | @aa:8 | @aa:16 | @aa:24 | @(d:8.PC) | @(d:16.PC) | @@aa:8 | \| |
| Data transfer instructions | MOV | BWL | BWL | BWL | BWL | BWL | BWL | B | BWL | BWL | — | — | — | — |
| | POP, PUSH | — | — | — | — | — | — | — | — | — | — | — | — | WL |
| | MOVFPE, MOVTPE | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Arithmetic operations | ADD, CMP | BWL | BWL | — | — | — | — | — | — | — | — | — | — | — |
| | SUB | WL | BWL | — | — | — | — | — | — | — | — | — | — | — |
| | ADDX, SUBX | B | B | — | — | — | — | — | — | — | — | — | — | — |
| | ADDS, SUBS | — | L | — | — | — | — | — | — | — | — | — | — | — |
| | INC, DEC | — | BWL | — | — | — | — | — | — | — | — | — | — | — |
| | DAA, DAS | — | B | — | — | — | — | — | — | — | — | — | — | — |
| | MULXU, MULXS, DIVXU, DIVXS | — | BW | — | — | — | — | — | — | — | — | — | — | — |
| | NEG | — | BWL | — | — | — | — | — | — | — | — | — | — | — |
| | EXTU, EXTS | — | WL | — | — | — | — | — | — | — | — | — | — | — |
| Logical operations | AND, OR, XOR | — | BWL | — | — | — | — | — | — | — | — | — | — | — |
| | NOT | — | BWL | — | — | — | — | — | — | — | — | — | — | — |
| Shift operations | | — | BWL | — | — | — | — | — | — | — | — | — | — | — |
| Bit manipulations | | — | B | B | — | — | — | B | — | — | — | — | — | — |
| Branching instructions | BCC, BSR | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | JMP, JSR | — | — | ○ | — | — | — | — | — | — | ○ | ○ | — | — |
| | RTS | — | — | — | — | — | — | — | — | ○ | — | — | ○ | — |
| System control instructions | RTE | — | — | — | — | — | — | — | — | — | — | — | — | ○ |
| | SLEEP | — | — | — | — | — | — | — | — | — | — | — | — | ○ |
| | LDC | B | B | W | W | W | W | — | W | W | — | — | — | ○ |
| | STC | — | B | W | W | W | W | — | W | W | — | — | — | — |
| | ANDC, ORC, XORC | B | — | — | — | — | — | — | — | — | — | — | — | — |
| | NOP | — | — | — | — | — | — | — | — | — | — | — | — | ○ |
| Block data transfer instructions | | — | — | — | — | — | — | — | — | — | — | — | — | BW |

# B. I/O Port Block Diagrams

## B.1 Block Diagrams of Port 1

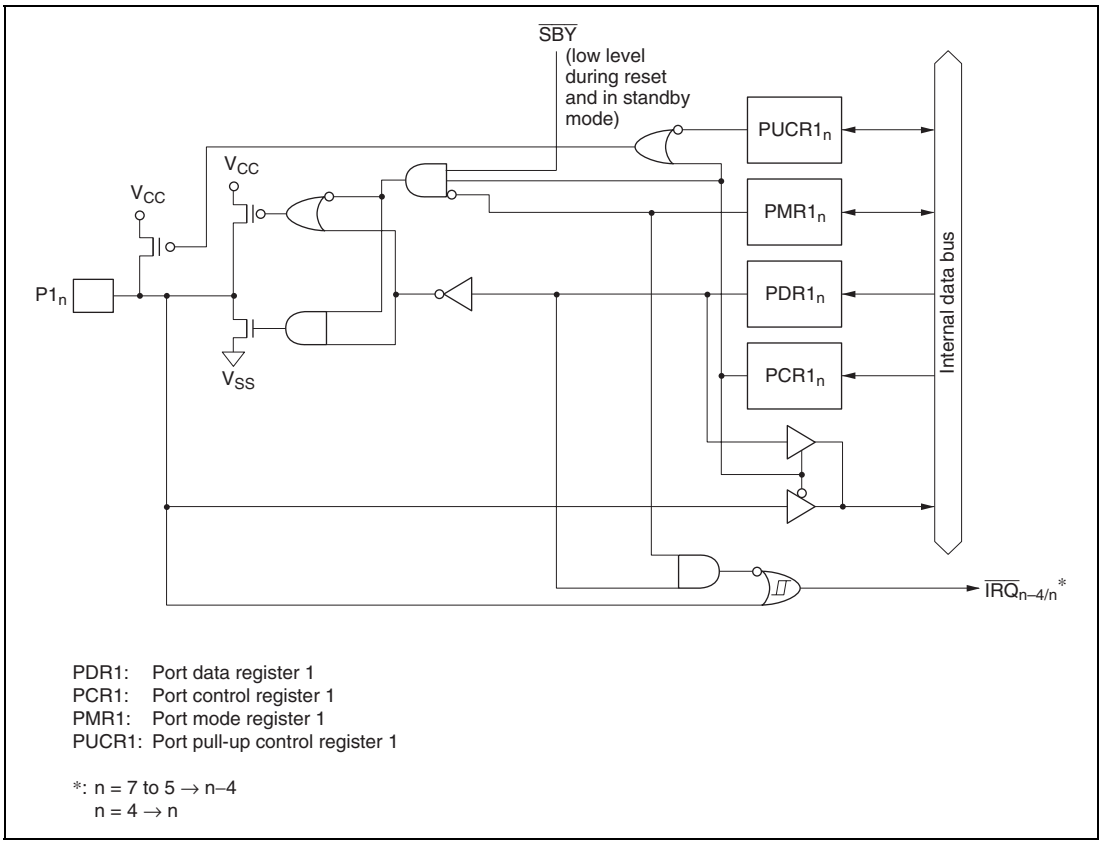



**Figure B.1 (a) Port 1 Block Diagram (Pins $P1_7$ to $P1_4$)**

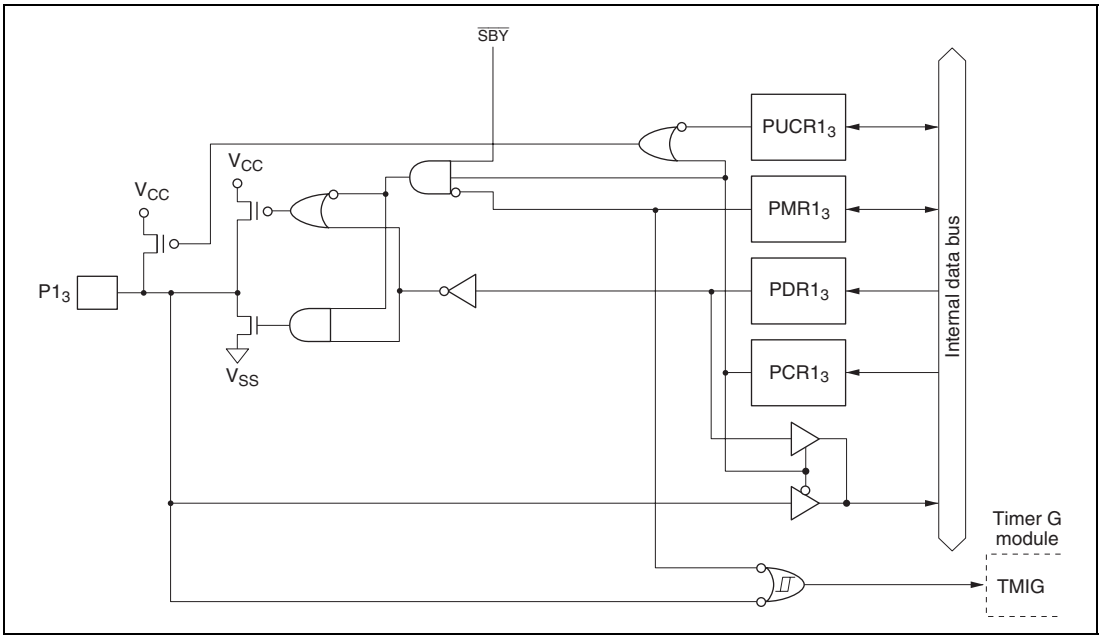



**Figure B.1 (b) Port 1 Block Diagram (Pin $P1_3$)**

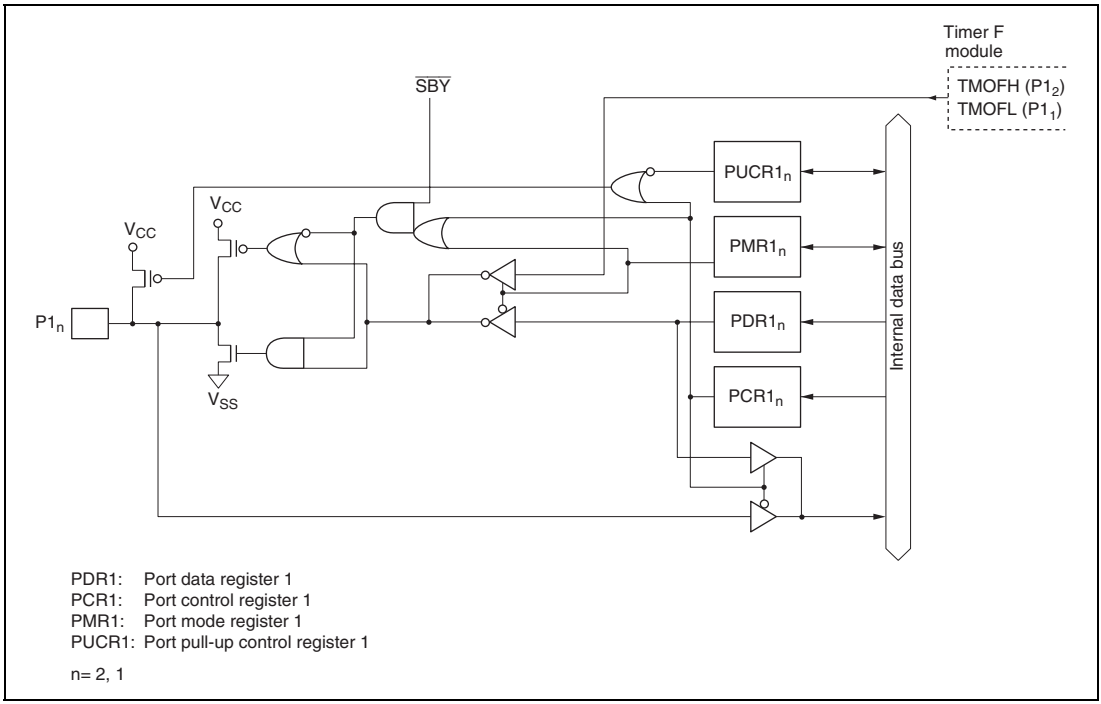



**Figure B.1 (c) Port 1 Block Diagram (Pin $P1_2$, $P1_1$)**

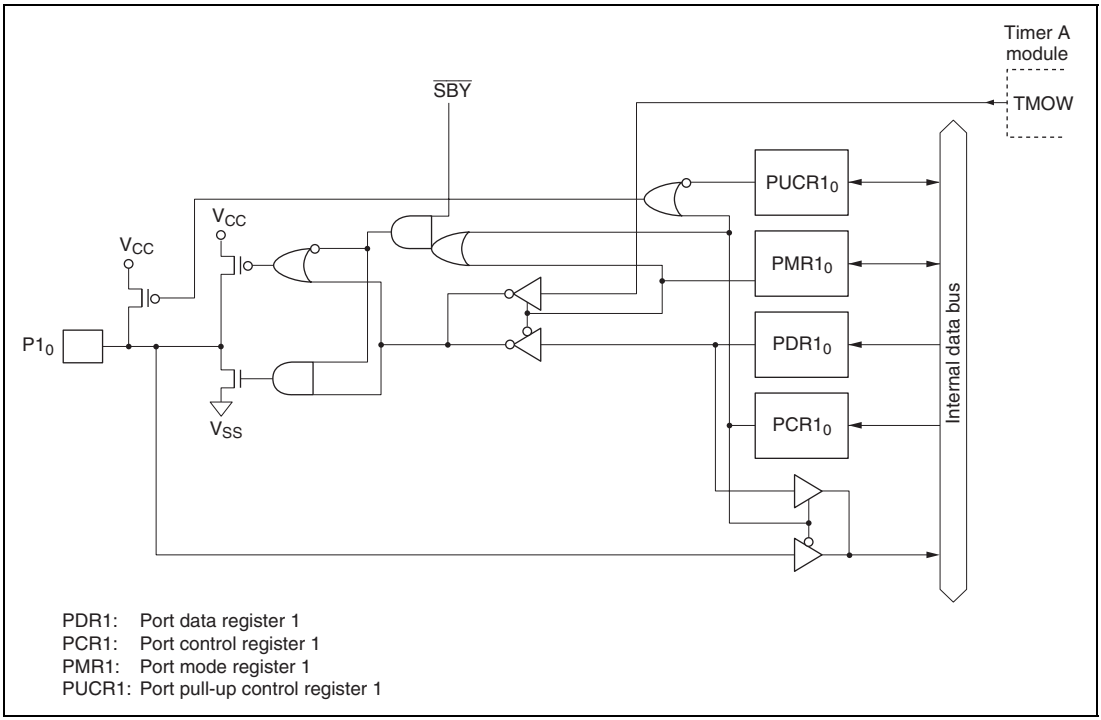



**Figure B.1 (d) Port 1 Block Diagram (Pin $P1_0$)**

## B.2 Block Diagrams of Port 3

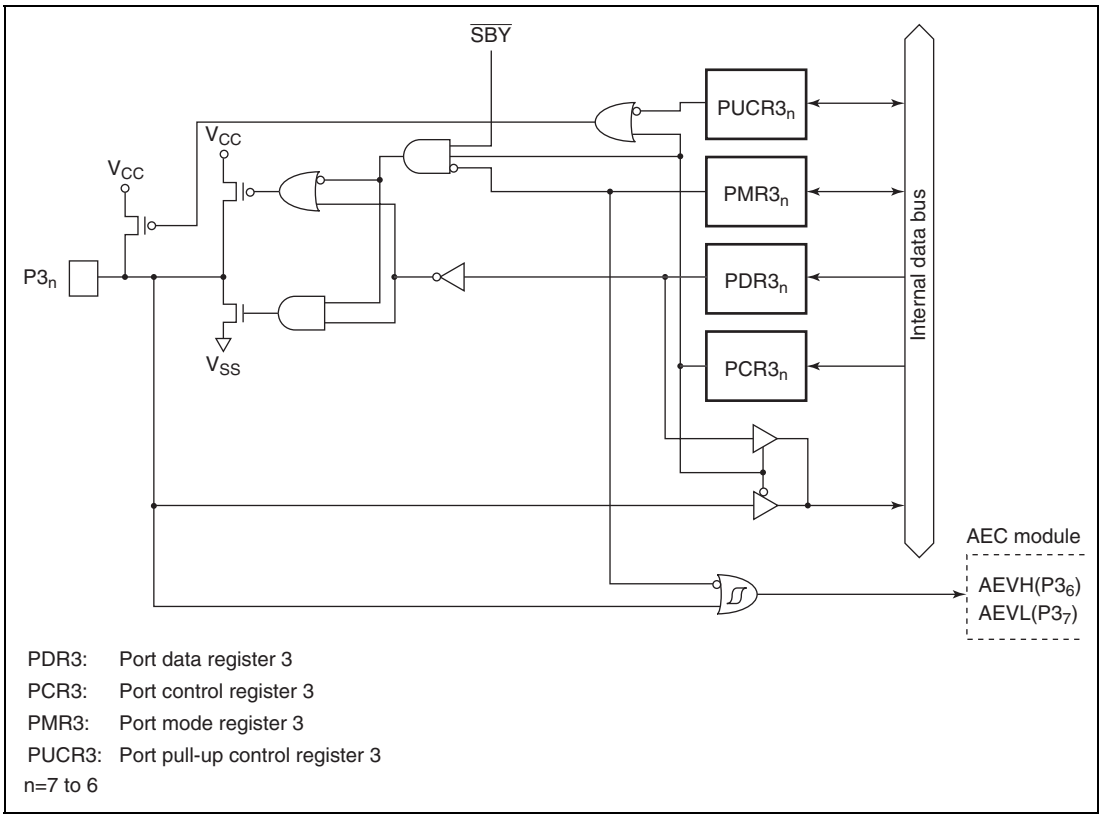
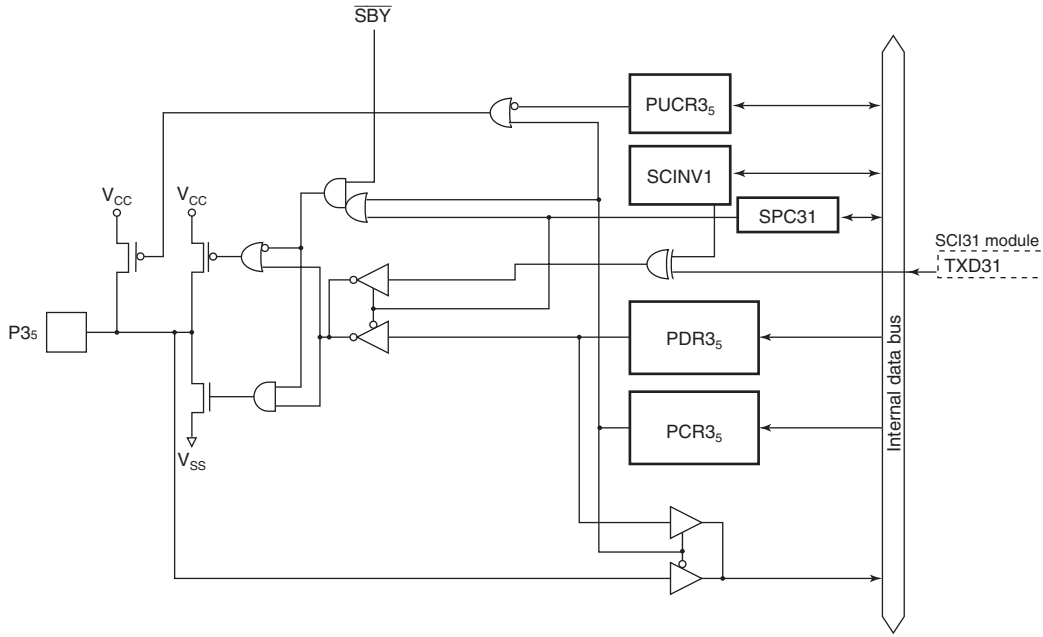



**Figure B.2 (a) Port 3 Block Diagram (Pin $P3_7$ to $P3_6$)**




PDR3: Port data register 3
PCR3: Port control register 3
PUCR3: Port pull-up control register 3
SCINV1: Bit 1 of serial port control register (SPCR)
SPC31: Bit 4 of serial port control register (SPCR)

**Figure B.2 (b) Port 3 Block Diagram (Pin $P3_5$)**

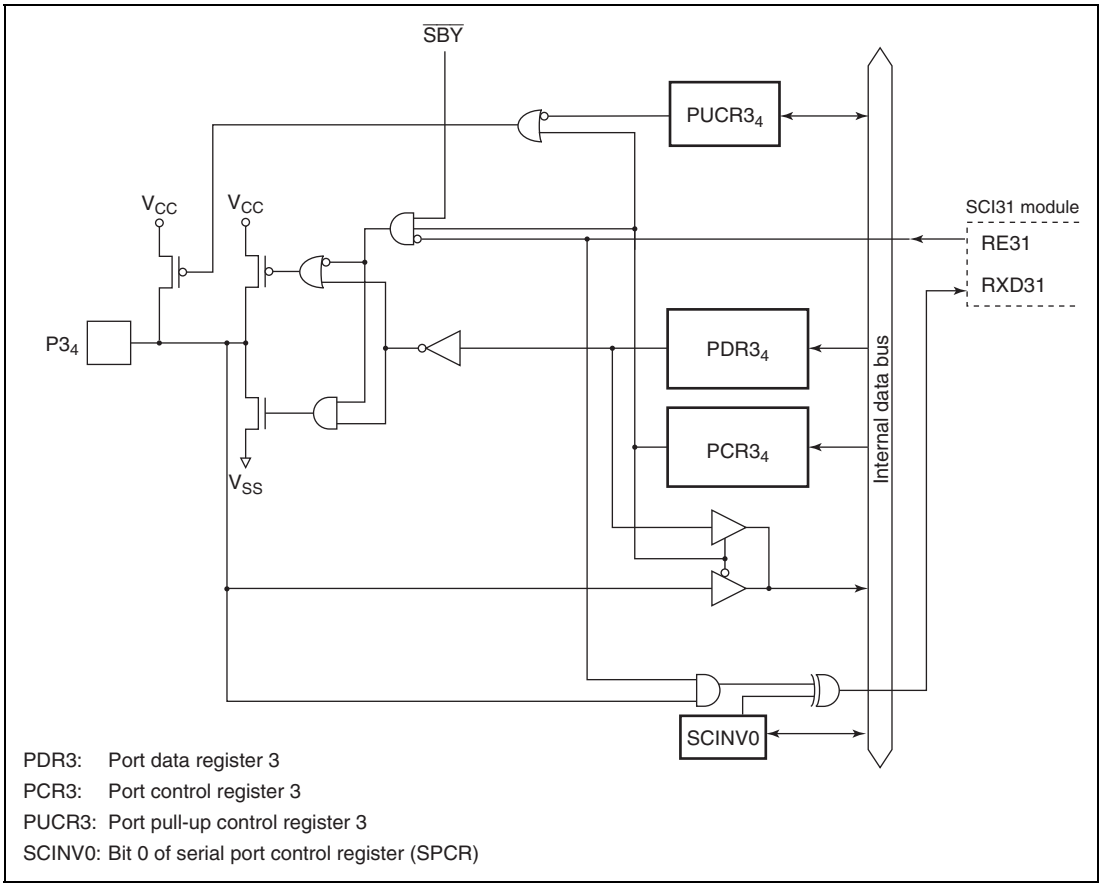



**Figure B.2 (c) Port 3 Block Diagram (Pin P3$_4$)**

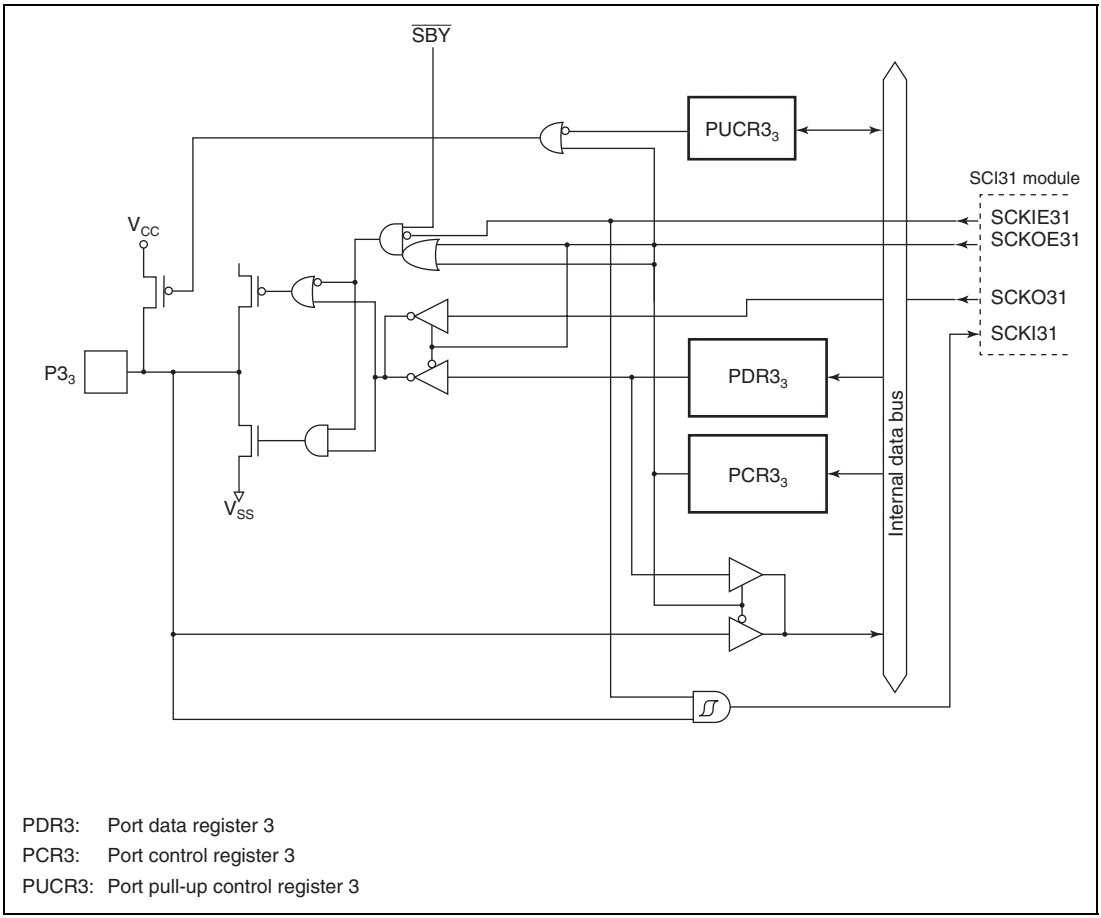



**Figure B.2 (d) Port 3 Block Diagram (Pin P3$_3$)**

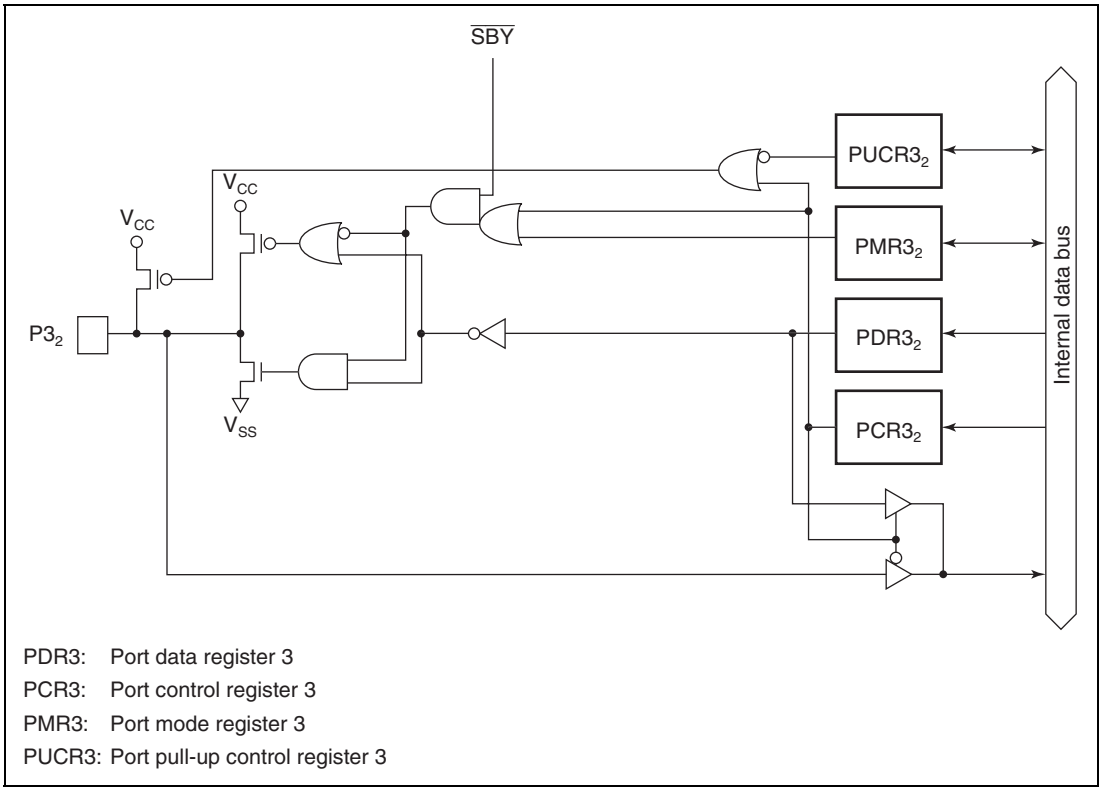



**Figure B.2 (e-1) Port 3 Block Diagram (Pin $P3_2$ in the Mask ROM Version)**

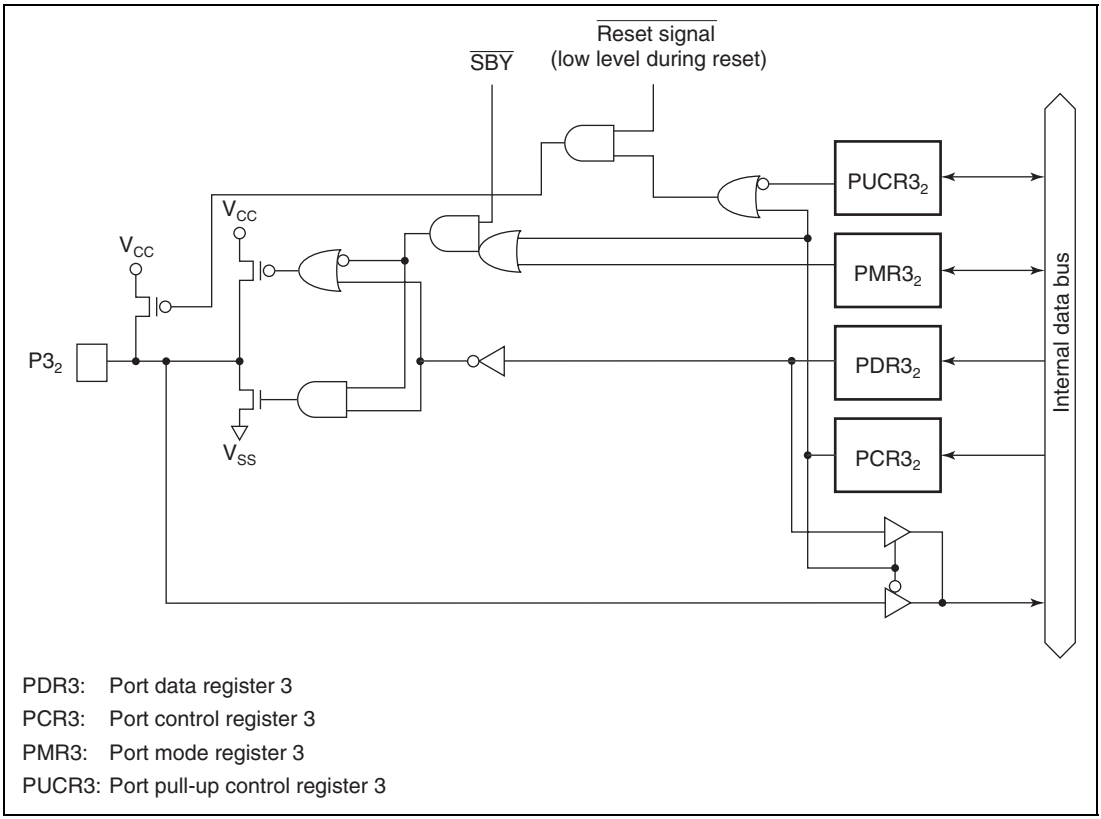



**Figure B.2 (e-2) Port 3 Block Diagram (Pin $P3_2$ in the Flash Memory Version)**

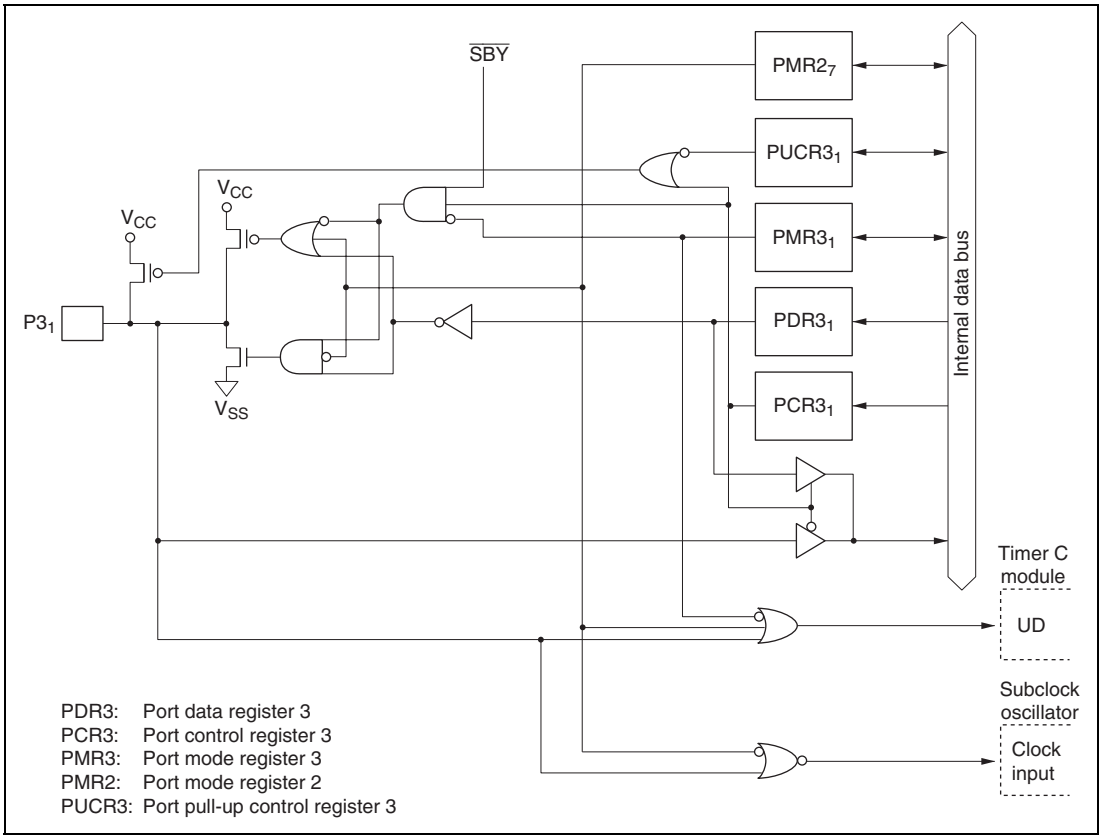



**Figure B.2 (f) Port 3 Block Diagram (Pin $P3_1$)**

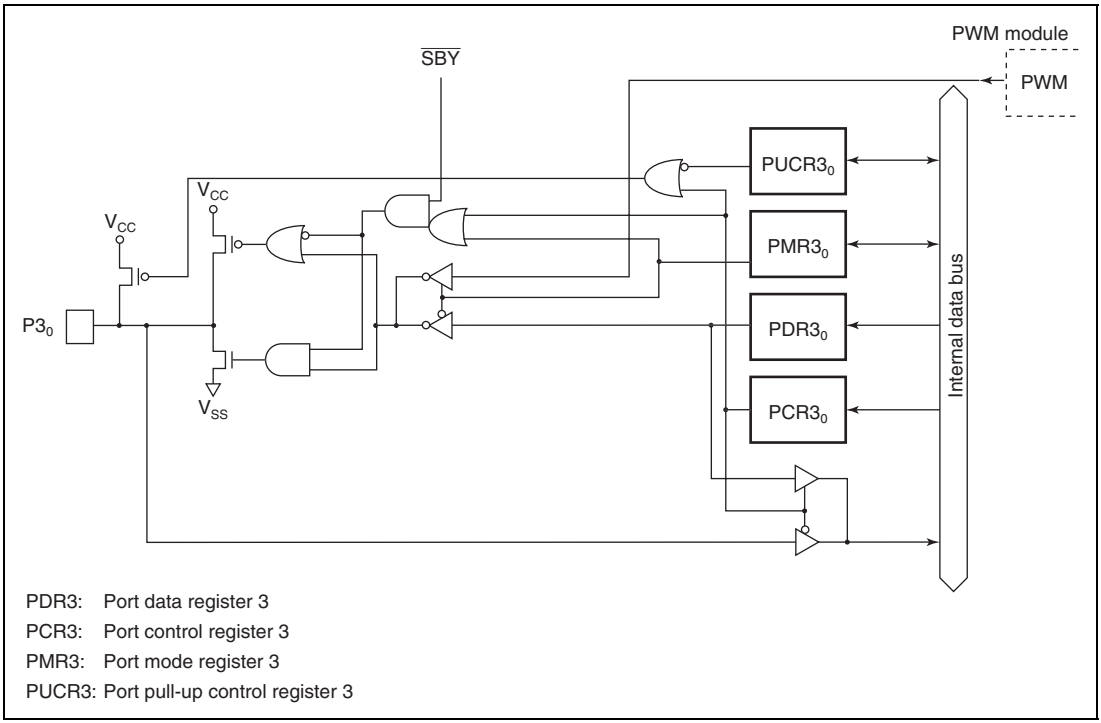



**Figure B.2 (g) Port 3 Block Diagram (Pin $P3_0$)**

## B.3 Block Diagrams of Port 4

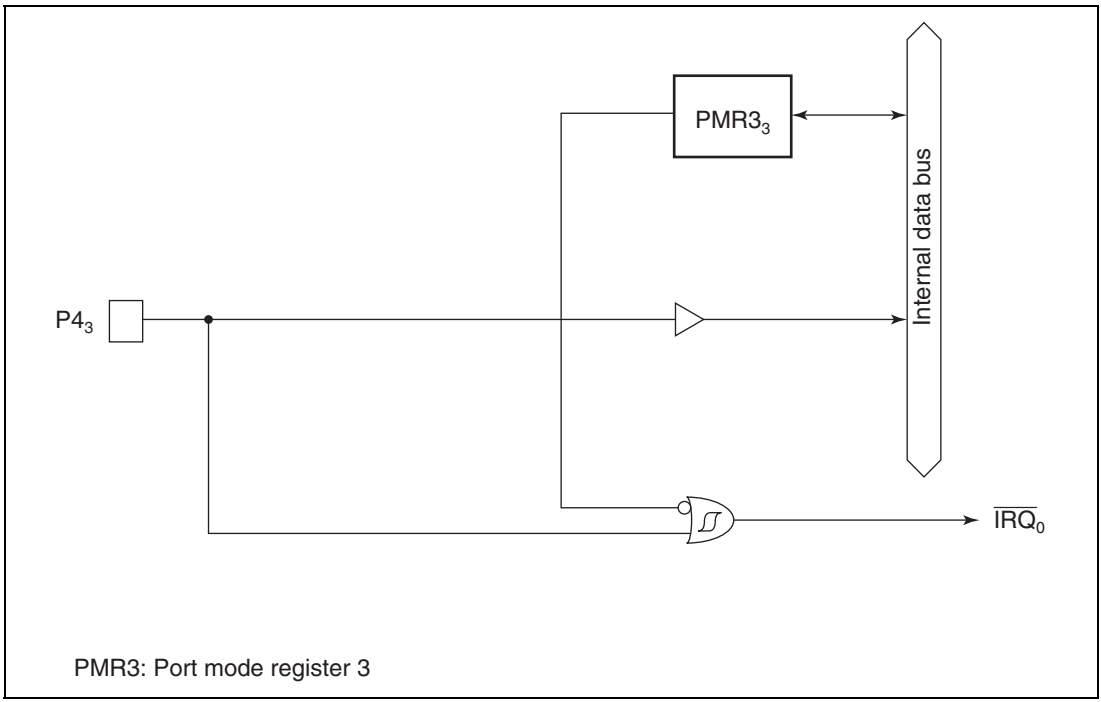
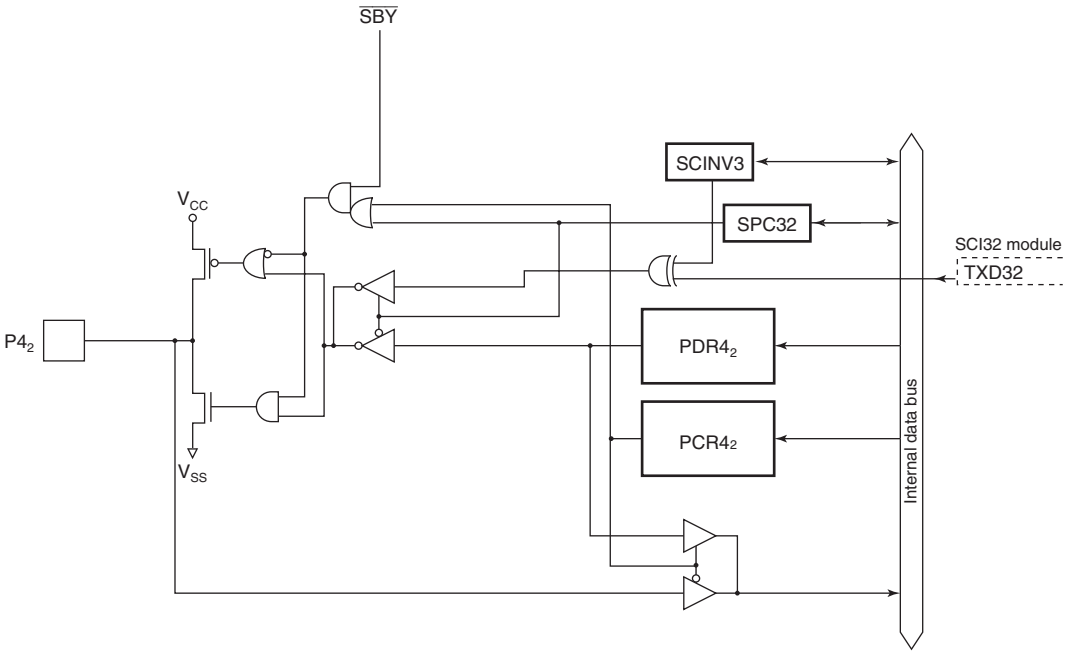



**Figure B.3 (a) Port 4 Block Diagram (Pin P4$_3$)**




PDR4: Port data register 4
PCR4: Port control register 4
SCINV3: Bit 3 of serial port control register (SPCR)
SPC32: Bit 5 of serial port control register (SPCR)

**Figure B.3 (b) Port 4 Block Diagram (Pin $P4_2$)**

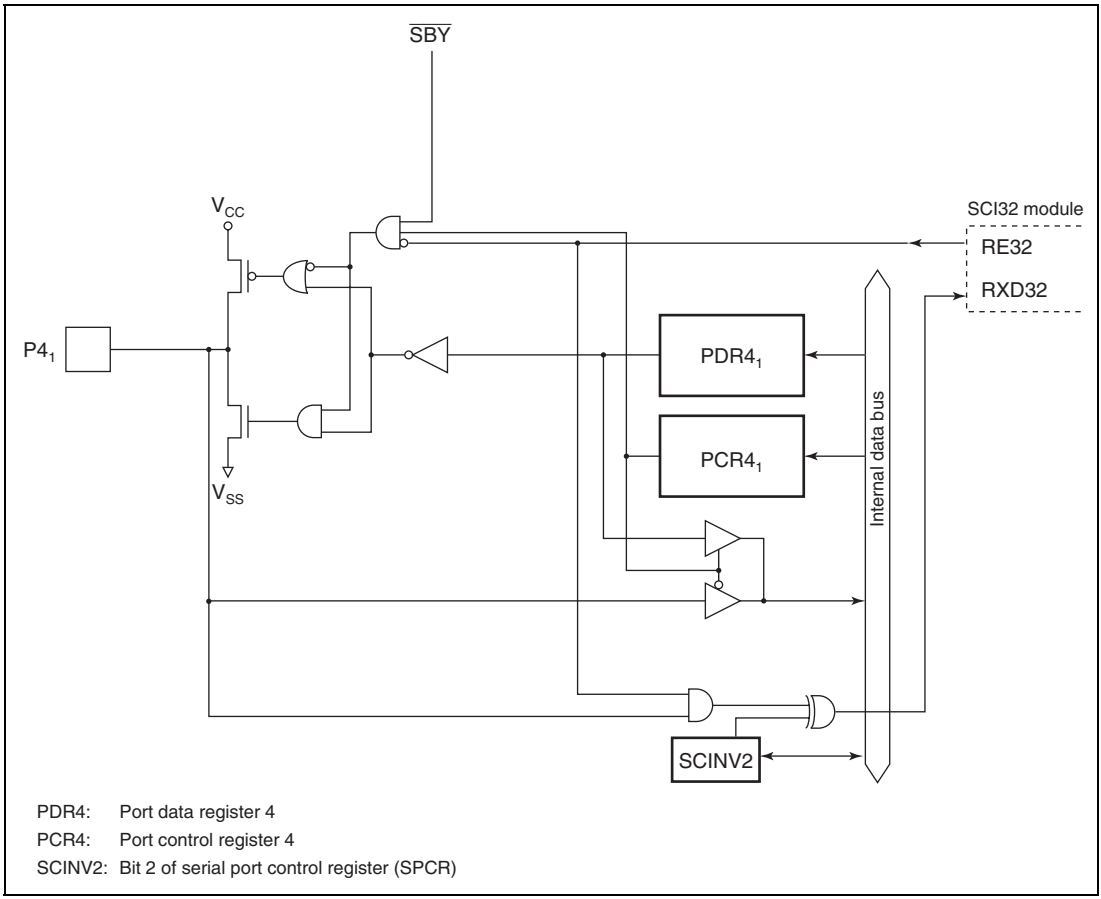



**Figure B.3 (c) Port 4 Block Diagram (Pin $P4_1$)**

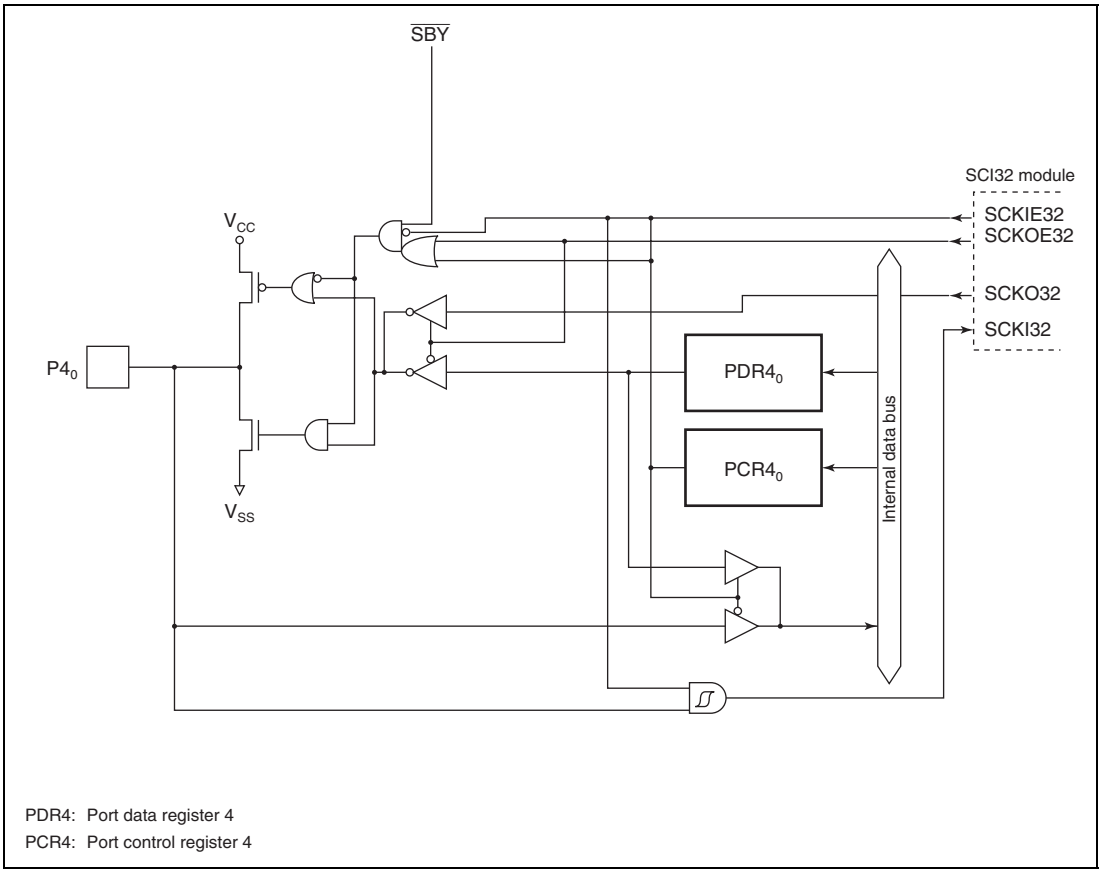



**Figure B.3 (d) Port 4 Block Diagram (Pin $P4_0$)**

## B.4 Block Diagram of Port 5

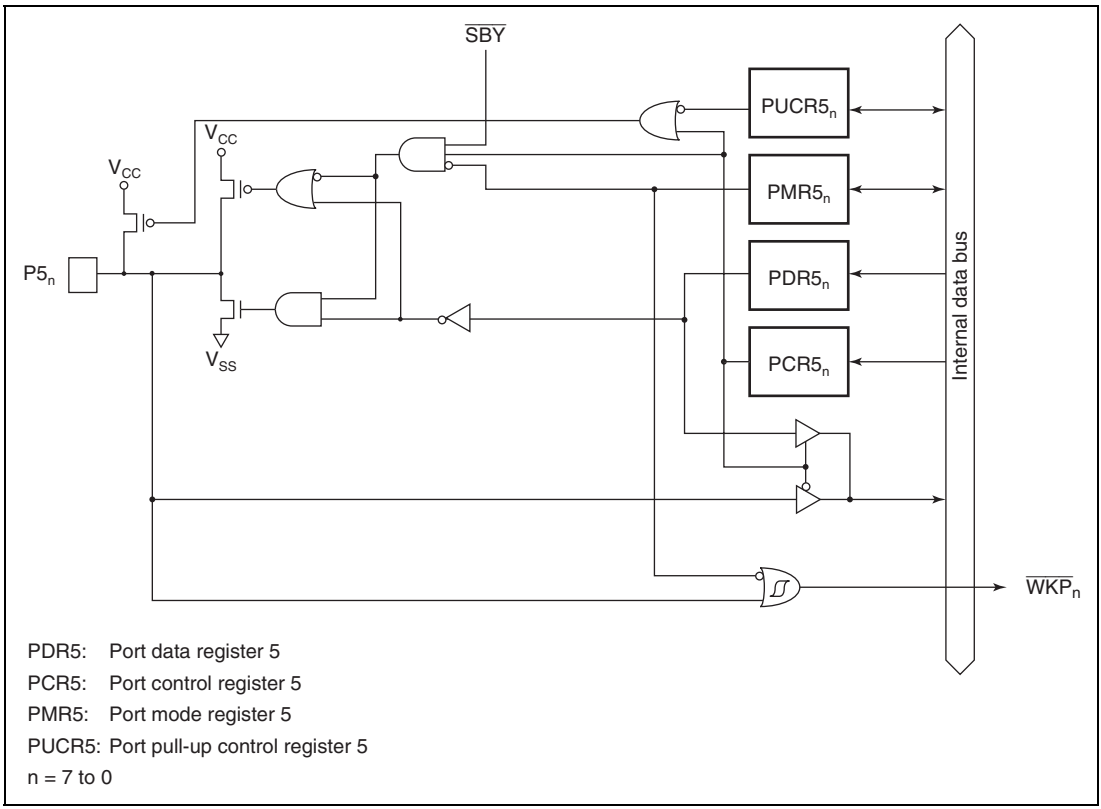
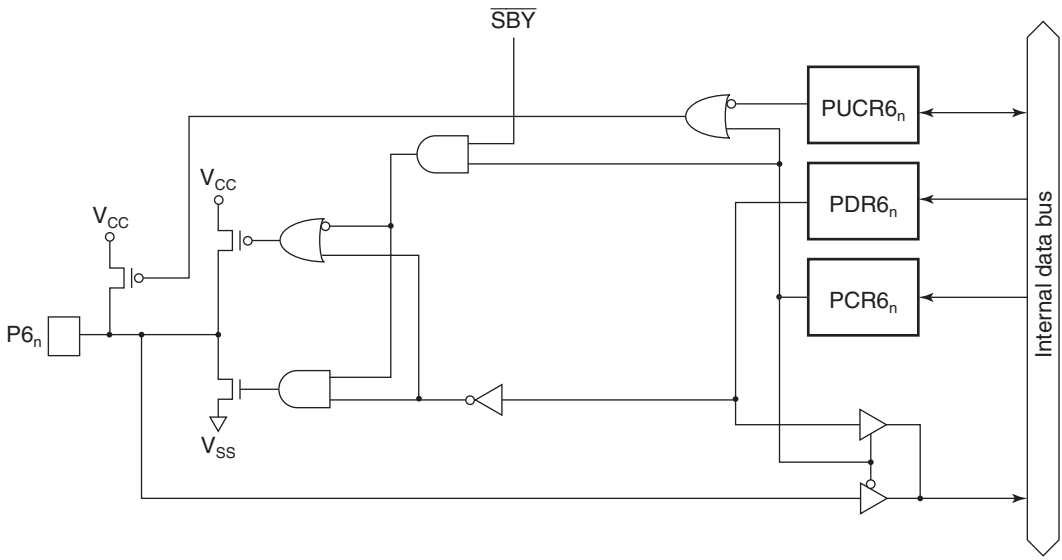



**Figure B.4 Port 5 Block Diagram**

## B.5 Block Diagram of Port 6




PDR6: Port data register 6
PCR6: Port control register 6
PUCR6: Port pull-up control register 6
n = 7 to 0

**Figure B.5 Port 6 Block Diagram**

## B.6 Block Diagram of Port 7

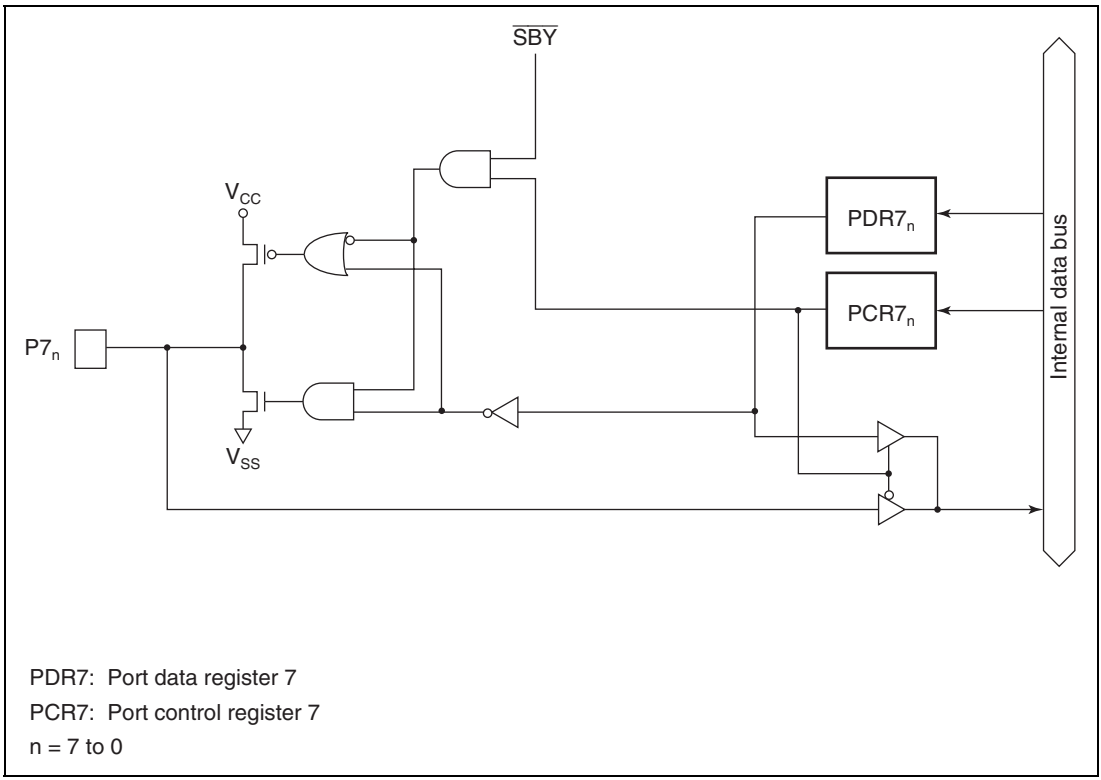
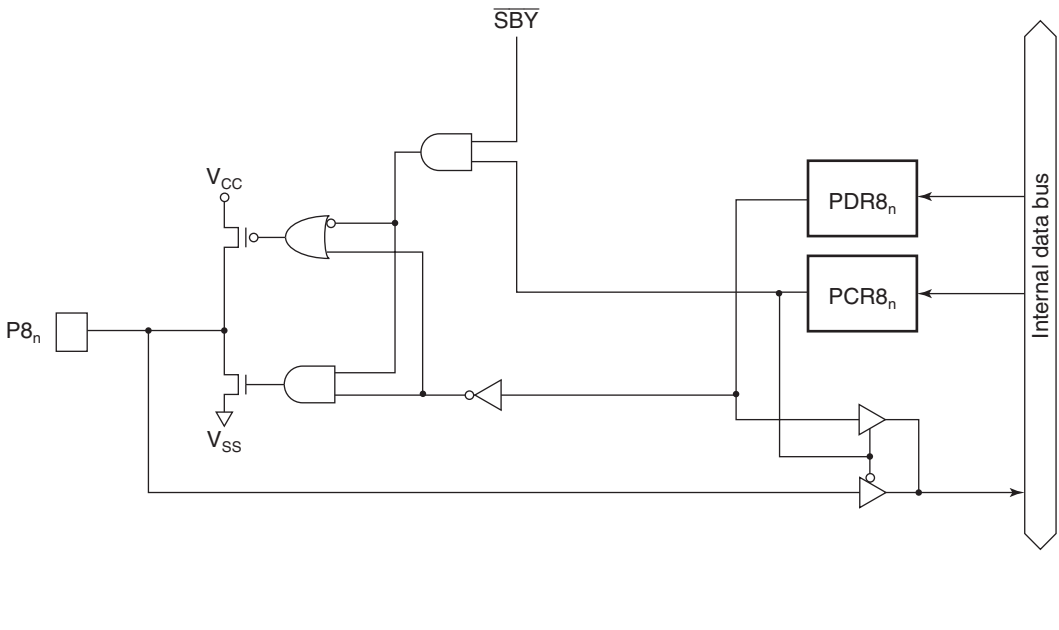



**Figure B.6 Port 7 Block Diagram**

## B.7 Block Diagrams of Port 8




PDR8: Port data register 8
PCR8: Port control register 8
n= 7 to 0

**Figure B.7 Port 8 Block Diagram**

## B.8 Block Diagram of Port A

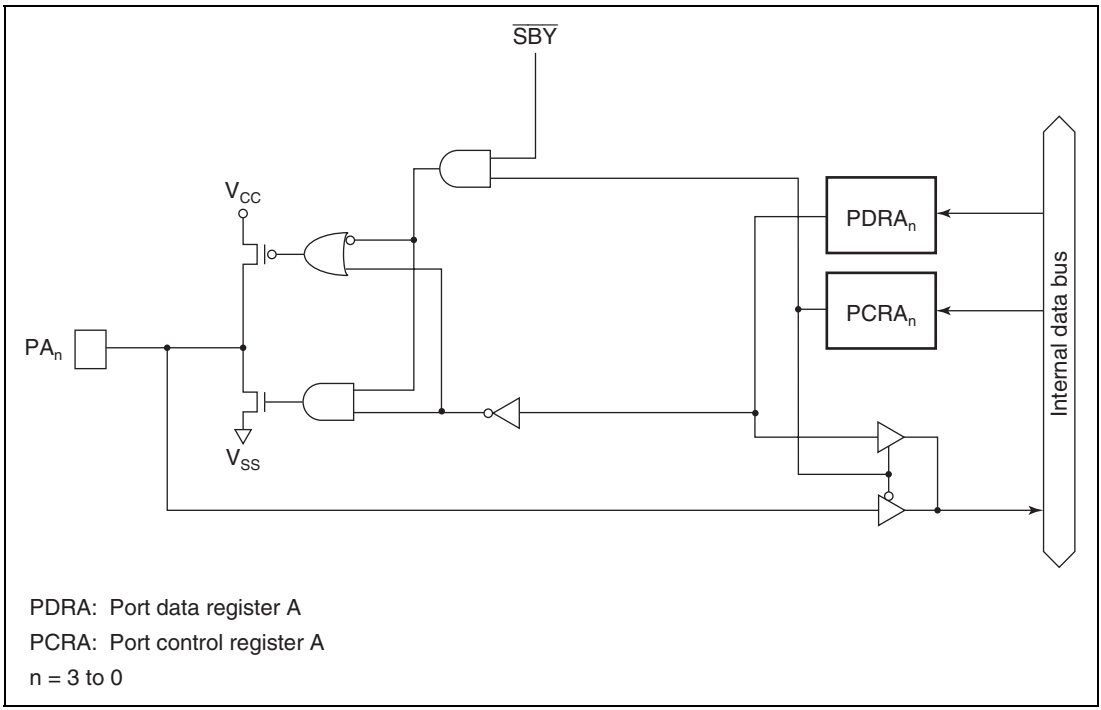



**Figure B.8 Port A Block Diagram**

## B.9 Block Diagram of Port B

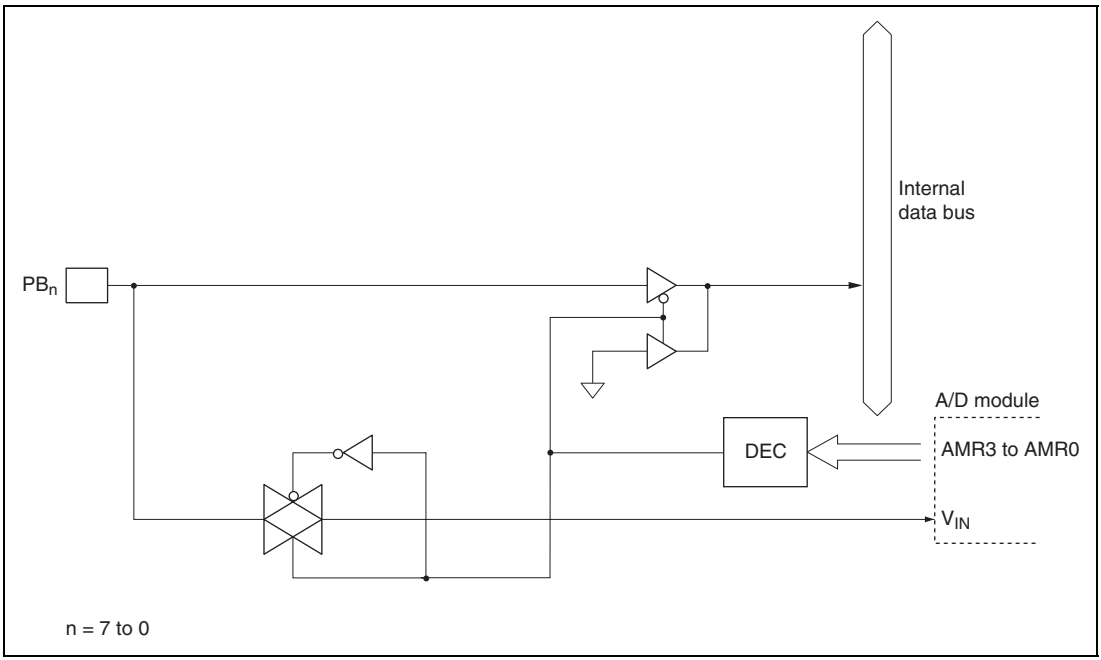



**Figure B.9 Port B Block Diagram**

# C. Port States in the Different Processing States

## Table C.1 Port States Overview

| Port | Reset | Sleep | Subsleep | Standby | Watch | Subactive | Active |
|---|---|---|---|---|---|---|---|
| $P1_7$ to $P1_0$ | High impedance | Retained | Retained | High impedance*1 | Retained | Functions | Functions |
| $P3_7$ to $P3_0$ | High impedance*2 | Retained | Retained | High impedance*1 | Retained | Functions | Functions |
| $P4_3$ to $P4_0$ | High impedance | Retained | Retained | High impedance | Retained | Functions | Functions |
| $P5_7$ to $P5_0$ | High impedance | Retained | Retained | High impedance*1 | Retained | Functions | Functions |
| $P6_7$ to $P6_0$ | High impedance | Retained | Retained | High impedance | Retained | Functions | Functions |
| $P7_7$ to $P7_0$ | High impedance | Retained | Retained | High impedance | Retained | Functions | Functions |
| $P8_7$ to $P8_0$ | High impedance | Retained | Retained | High impedance | Retained | Functions | Functions |
| $PA_3$ to $PA_0$ | High impedance | Retained | Retained | High impedance | Retained | Functions | Functions |
| $PB_7$ to $PB_0$ | High impedance | High impedance | High impedance | High impedance | High impedance | High impedance | High impedance |

Notes: 1. High level output when MOS pull-up is in on state.
2. On-chip pull-up MOS turns on for pin $P3_2$ only (Flash Memory Version).

# D. List of Product Codes

## Table D.1 Product Code Lineup

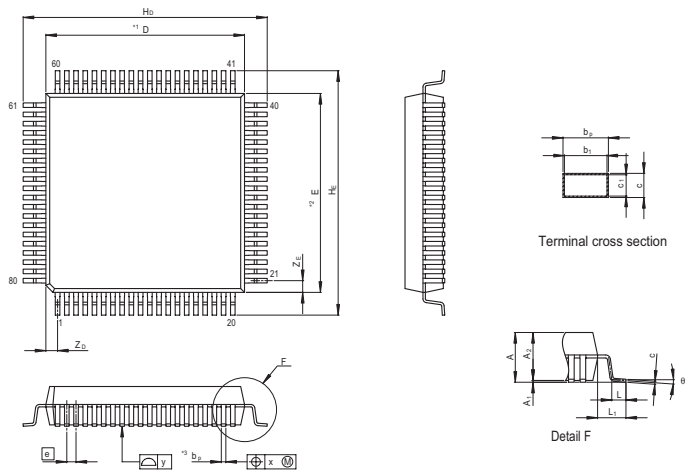
| Product Type | | | | Product Code | Mark Code | Package (Package Code) |
|---|---|---|---|---|---|---|
| H8/38537 Group | H8/38532 | Mask ROM versions | Regular products | HD64338532H | 38532H | 80-pin QFP (FP-80A) |
| | | | | HD64338532W | 38532W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338532HW | 38532H | 80-pin QFP (FP-80A) |
| | | | | HD64338532WW | 38532W | 80-pin TQFP (TFP-80C) |
| | H8/38533 | Mask ROM versions | Regular products | HD64338533H | 38533H | 80-pin QFP (FP-80A) |
| | | | | HD64338533W | 38533W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338533HW | 38533H | 80-pin QFP (FP-80A) |
| | | | | HD64338533WW | 38533W | 80-pin TQFP (TFP-80C) |
| | H8/38534 | Mask ROM versions | Regular products | HD64338534H | 38534H | 80-pin QFP (FP-80A) |
| | | | | HD64338534W | 38534W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338534HW | 38534H | 80-pin QFP (FP-80A) |
| | | | | HD64338534WW | 38534W | 80-pin TQFP (TFP-80C) |
| | | Flash memory versions | Regular products | HD64F38534H | F38534H | 80-pin QFP (FP-80A) |
| | | | | HD64F38534W | F38534W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64F38534HW | F38534H | 80-pin QFP (FP-80A) |
| | | | | HD64F38534WW | F38534W | 80-pin TQFP (TFP-80C) |

| Product Type | | | | Product Code | Mark Code | Package (Package Code) |
|---|---|---|---|---|---|---|
| H8/38537 Group | H8/38535 | Mask ROM versions | Regular products | HD64338535H | 38535H | 80-pin QFP (FP-80A) |
| | | | | HD64338535W | 38535W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338535HW | 38535H | 80-pin QFP (FP-80A) |
| | | | | HD64338535WW | 38535W | 80-pin TQFP (TFP-80C) |
| | H8/38536 | Mask ROM versions | Regular products | HD64338536H | 38536H | 80-pin QFP (FP-80A) |
| | | | | HD64338536W | 38536W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338536HW | 38536H | 80-pin QFP (FP-80A) |
| | | | | HD64338536WW | 38536W | 80-pin TQFP (TFP-80C) |
| | H8/38537 | Mask ROM versions | Regular products | HD64338537H | 38537H | 80-pin QFP (FP-80A) |
| | | | | HD64338537W | 38537W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64338537HW | 38537H | 80-pin QFP (FP-80A) |
| | | | | HD64338537WW | 38537W | 80-pin TQFP (TFP-80C) |
| | | Flash memory versions | Regular products | HD64F38537H | F38537H | 80-pin QFP (FP-80A) |
| | | | | HD64F38537W | F38537W | 80-pin TQFP (TFP-80C) |
| | | | Wide-range specification products | HD64F38537HW | F38537H | 80-pin QFP (FP-80A) |
| | | | | HD64F38537WW | F38537W | 80-pin TQFP (TFP-80C) |

# E. Package Dimensions

Dimensional drawings of the packages FP-80A and TFP-80C are shown in figures E.1 and E.2, below.

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-QFP80-14x14-0.65 | PRQP0080JB-A | FP-80A/FP-80AV | 1.2g |

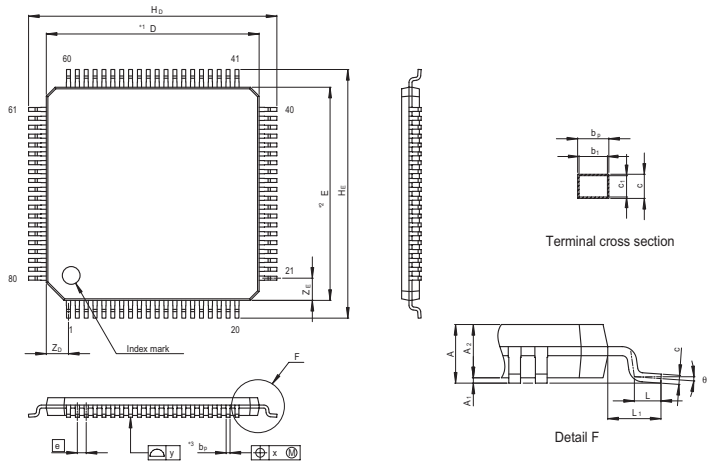



NOTE)
1. DIMENSIONS"*1"AND"*2" DO NOT INCLUDE MOLD FLASH
2. DIMENSION"*3"DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | — | 14 | — |
| E | — | 14 | — |
| $A_2$ | — | 2.70 | — |
| $H_D$ | 16.9 | 17.2 | 17.5 |
| $H_E$ | 16.9 | 17.2 | 17.5 |
| A | — | — | 3.05 |
| $A_1$ | 0.00 | 0.10 | 0.25 |
| $b_p$ | 0.24 | 0.32 | 0.40 |
| $b_1$ | — | 0.30 | — |
| c | 0.12 | 0.17 | 0.22 |
| $c_1$ | — | 0.15 | — |
| θ | 0° | — | 8° |
| e | — | 0.65 | — |
| x | — | — | 0.12 |
| y | — | — | 0.10 |
| $Z_D$ | — | 0.83 | — |
| $Z_E$ | — | 0.83 | — |
| L | 0.5 | 0.8 | 1.1 |
| $L_1$ | — | 1.6 | — |

**Figure E.1 FP-80A Package Dimensions**

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-TQFP80-12x12-0.50 | PTQP0080KC-A | TFP-80C/TFP-80CV | 0.4g |




NOTE)
1. DIMENSIONS"*1"AND"*2"
DO NOT INCLUDE MOLD FLASH
2. DIMENSION"*3"DOES NOT
INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | — | 12 | — |
| E | — | 12 | — |
| $A_2$ | — | 1.00 | — |
| $H_D$ | 13.8 | 14.0 | 14.2 |
| $H_1$ | 13.8 | 14.0 | 14.2 |
| A | — | — | 1.20 |
| $A_1$ | 0.00 | 0.10 | 0.20 |
| $b_p$ | 0.17 | 0.22 | 0.27 |
| $b_1$ | — | 0.20 | — |
| c | 0.12 | 0.17 | 0.22 |
| $c_1$ | — | 0.15 | — |
| θ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.10 |
| y | — | — | 0.10 |
| $Z_D$ | — | 1.25 | — |
| $Z_E$ | — | 1.25 | — |
| L | 0.4 | 0.5 | 0.6 |
| $L_1$ | — | 1.0 | — |

**Figure E.2 TFP-80C Package Dimensions**

# Index


**A**

ADRRH ..... 352
ADRRL ..... 352
ADSR ..... 354
AMR ..... 352

**B**

BRR ..... 303

**C**

CKSTPR1 .....212, 220, 234, 251, 308, 355
CKSTPR2 ..... 269, 279, 345, 374
CWOSR ..... 212

**E**

EBR ..... 130
ECCSR ..... 275
ECH ..... 278
ECL ..... 279

**F**

FENR ..... 132
FLMCR1 ..... 126
FLMCR2 ..... 129
FLPWCR ..... 131

**I**

ICRGF ..... 248
ICRGR ..... 248
IEGR ..... 63
IENR1 ..... 65
IENR2 ..... 66
IRR1 ..... 68
IRR2 ..... 69
IWPR ..... 72

**L**

LCR ..... 370
LCR2 ..... 372
LPCR ..... 368

**O**

OCRF ..... 228
OCRFH ..... 228
OCRFL ..... 228

**P**

PCR1 ..... 162
PCR3 ..... 170
PCR4 ..... 179
PCR5 ..... 182
PCR6 ..... 186
PCR7 ..... 190
PCR8 ..... 193
PCRA ..... 196
PDR1 ..... 162
PDR3 ..... 170
PDR4 ..... 178
PDR5 ..... 182
PDR6 ..... 186
PDR7 ..... 190
PDR8 ..... 193
PDRA ..... 196
PDRB ..... 199
PMR1 ..... 163
PMR2 ..... 173

PMR3 ..... 171, 270
PMR5 ..... 183
PUCR1 ..... 162
PUCR3 ..... 171
PUCR5 ..... 182
PUCR6 ..... 187
PWCR ..... 343
PWDRL ..... 344
PWDRU ..... 344

## R

RDR ..... 289
RSR ..... 289

## S

SCR3 ..... 294
SMR ..... 291
SPCR ..... 201, 309
SSR ..... 298
SYSCR1 ..... 100
SYSCR2 ..... 102

## T

TCA ..... 211
TCC ..... 219
TCF ..... 227
TCFH ..... 227
TCFL ..... 227
TCG ..... 247
TCRF ..... 229
TCSRF ..... 231
TCSRW ..... 266
TCW ..... 269
TDR ..... 290
TLC ..... 219
TMA ..... 209
TMC ..... 217
TMG ..... 249
TSR ..... 290

## W

WEGR ..... 72

**Renesas 16-Bit Single-Chip Microcomputer**
**Hardware Manual**
**H8/38537 Group**

Publication Date: Rev.1.00, May 30, 2008
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Customer Support Department
Global Strategic Communication Div.
Renesas Solutions Corp.

© 2008. Renesas Technology Corp., All rights reserved. Printed in Japan.

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

## RENESAS SALES OFFICES

http://www.renesas.com

Refer to "**http://www.renesas.com/en/network**" for the latest and detailed information.

**Renesas Technology America, Inc.**
450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

**Renesas Technology Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

**Renesas Technology (Shanghai) Co., Ltd.**
Unit 204, 205, AZIACenter, No.1233 Lujiazui Ring Rd, Pudong District, Shanghai, China 200120
Tel: <86> (21) 5877-1818, Fax: <86> (21) 6887-7858/7898

**Renesas Technology Hong Kong Ltd.**
7th Floor, North Tower, World Finance Centre, Harbour City, Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2377-3473

**Renesas Technology Taiwan Co., Ltd.**
10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 3518-3399

**Renesas Technology Singapore Pte. Ltd.**
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

**Renesas Technology Korea Co., Ltd.**
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> (2) 796-3115, Fax: <82> (2) 796-2145

**Renesas Technology Malaysia Sdn. Bhd**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

Colophon 6.2

# H8/38537 Group
# Hardware Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan