REJ09B0143-0400

Everywhere you imagine. RENESAS

The revision list can be viewed directly by clicking the title page.

The revision list summarizes the locations of revisions and additions. Details should always be checked by referring to the relevant text.

16

# H8/3672 Group

## Hardware Manual

Renesas 16-Bit Single-Chip Microcomputer
H8 Family/H8/300H Tiny Series

H8/3672F HD64F3672
H8/3670F HD64F3670

Hardware Manual

Rev.4.00
Revision Date: Nov. 02, 2005

Renesas Technology
www.renesas.com

## Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

## Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.

## General Precautions on Handling of Product

1. Treatment of NC Pins

Note: Do not connect anything to the NC pins.
The NC (not connected) pins are either not connected to any of the internal circuitry or are used as test pins or to reduce noise. If something is connected to the NC pins, the operation of the LSI is not guaranteed.

2. Treatment of Unused Input Pins

Note: Fix all unused input pins to high or low level.
Generally, the input pins of CMOS products are high-impedance input pins. If unused pins are in their open states, intermediate levels are induced by noise in the vicinity, a pass-through current flows internally, and a malfunction may occur.

3. Processing before Initialization

Note: When power is first supplied, the product's state is undefined.
The states of internal circuits are undefined until full power is supplied throughout the chip and a low level is input on the reset pin. During the period where the states are undefined, the register settings and the output state of each pin are also undefined. Design your system so that it does not malfunction because of processing while it is in this undefined state. For those products which have a reset function, reset the LSI immediately after the power supply has been turned on.

4. Prohibition of Access to Undefined or Reserved Addresses

Note: Access to undefined or reserved addresses is prohibited.
The undefined or reserved addresses may be used to expand functions, or test registers may have been be allocated to these addresses. Do not access these registers; the system's operation is not guaranteed if they are accessed.

# Configuration of This Manual

This manual comprises the following items:

1. General Precautions on Handling of Product
2. Configuration of This Manual
3. Preface
4. Contents
5. Overview
6. Description of Functional Modules
   - CPU and System-Control Modules
   - On-Chip Peripheral Modules

     The configuration of the functional description of each module differs according to the module. However, the generic style includes the following items:

   i) Feature
   ii) Input/Output Pin
   iii) Register Description
   iv) Operation
   v) Usage Note

When designing an application system that includes this LSI, take notes into account. Each section includes notes in relation to the descriptions given, and usage notes are given, as required, as the final part of each section.

7. List of Registers
8. Electrical Characteristics
9. Appendix
10. Main Revisions and Additions in this Edition (only for revised versions)

The list of revisions is a summary of points that have been revised or added to earlier versions. This does not include all of the revised contents. For details, see the actual locations in this manual.

11. Index

# Preface

The H8/3672 Group are single-chip microcomputers made up of the high-speed H8/300H CPU employing Renesas Technology original architecture as their cores, and the peripheral functions required to configure a system. The H8/300H CPU has an instruction set that is compatible with the H8/300 CPU.

Target Users: This manual was written for users who will be using the H8/3672 Group in the design of application systems. Target users are expected to understand the fundamentals of electrical circuits, logical circuits, and microcomputers.

Objective: This manual was written to explain the hardware functions and electrical characteristics of the H8/3672 Group to the target users.
Refer to the H8/300H Series Software Manual for a detailed description of the instruction set.

Notes on reading this manual:

- In order to understand the overall functions of the chip
  Read the manual according to the contents. This manual can be roughly categorized into parts on the CPU, system control functions, peripheral functions and electrical characteristics.
- In order to understand the details of the CPU's functions
  Read the H8/300H Series Software Manual.
- In order to understand the details of a register when its name is known
  Read the index that is the final part of the manual to find the page number of the entry on the register. The addresses, bits, and initial values of the registers are summarized in section 16, List of Registers.
  Example: Bit order: The MSB is on the left and the LSB is on the right.

Notes:

When using the on-chip emulator (E7, E8) for H8/3672 program development and debugging, the following restrictions must be noted.

1. The $\overline{\text{NMI}}$ pin is reserved for the E7 or E8, and cannot be used.
2. Area H'4000 to H'4FFF is used by the E7 or E8, and is not available to the user.
3. Area H'F780 to H'FB7F must on no account be accessed.
4. When the E7 or E8 is used, address breaks can be set as either available to the user or for use by the E7 or E8. If address breaks are set as being used by the E7 or E8, the address break control registers must not be accessed.

5. When the E7 or E8 is used, $\overline{\text{NMI}}$ is an input/output pin (open-drain in output mode).

Related Manuals: The latest versions of all related manuals are available from our web site. Please ensure you have the latest versions of all documents you require. http://www.renesas.com/

H8/3672 Group manuals:

| Document Title | Document No. |
|---|---|
| H8/3672 Group Hardware Manual | This manual |
| H8/300H Series Software Manual | REJ09B0213 |

User's manuals for development tools:

| Document Title | Document No. |
|---|---|
| H8S, H8/300 Series C/C++ Compiler, Assembler, Optimizing Linkage Editor User's Manual | REJ10B0058 |
| Microcomputer Development Environment System H8S, H8/300 Series Simulator/Debugger User's Manual | ADE-702-282 |
| H8S, H8/300 Series High-Performance Embedded Workshop 3, Tutorial | REJ10B0024 |
| H8S, H8/300 Series High-Performance Embedded Workshop 3, User's Manual | REJ10B0026 |

Application notes:

| Document Title | Document No. |
|---|---|
| H8S, H8/300 Series C/C++ Compiler Package Application Note | REJ05B0464 |
| Single Power Supply F-ZTAT™ On-Board Programming | ADE-502-055 |

# Contents


Section 1 Overview ..... 1
1.1 Features ..... 1
1.2 Internal Block Diagram ..... 2
1.3 Pin Arrangement ..... 3
1.4 Pin Functions ..... 5

Section 2 CPU ..... 7
2.1 Address Space and Memory Map ..... 8
2.2 Register Configuration ..... 9
2.2.1 General Registers ..... 10
2.2.2 Program Counter (PC) ..... 11
2.2.3 Condition-Code Register (CCR) ..... 11
2.3 Data Formats ..... 13
2.3.1 General Register Data Formats ..... 13
2.3.2 Memory Data Formats ..... 15
2.4 Instruction Set ..... 16
2.4.1 Table of Instructions Classified by Function ..... 16
2.4.2 Basic Instruction Formats ..... 25
2.5 Addressing Modes and Effective Address Calculation ..... 27
2.5.1 Addressing Modes ..... 27
2.5.2 Effective Address Calculation ..... 30
2.6 Basic Bus Cycle ..... 32
2.6.1 Access to On-Chip Memory (RAM, ROM) ..... 32
2.6.2 On-Chip Peripheral Modules ..... 33
2.7 CPU States ..... 34
2.8 Usage Notes ..... 35
2.8.1 Notes on Data Access to Empty Areas ..... 35
2.8.2 EEPMOV Instruction ..... 35
2.8.3 Bit Manipulation Instruction ..... 35

Section 3 Exception Handling ..... 41
3.1 Exception Sources and Vector Address ..... 42
3.2 Register Descriptions ..... 43
3.2.1 Interrupt Edge Select Register 1 (IEGR1) ..... 43
3.2.2 Interrupt Edge Select Register 2 (IEGR2) ..... 44
3.2.3 Interrupt Enable Register 1 (IENR1) ..... 45
3.2.4 Interrupt Flag Register 1 (IRR1) ..... 46

3.2.5 Wakeup Interrupt Flag Register (IWPR) ..... 47
3.3 Reset Exception Handling ..... 48
3.4 Interrupt Exception Handling ..... 48
3.4.1 External Interrupts ..... 48
3.4.2 Internal Interrupts ..... 49
3.4.3 Interrupt Handling Sequence ..... 50
3.4.4 Interrupt Response Time ..... 51
3.5 Usage Notes ..... 53
3.5.1 Interrupts after Reset ..... 53
3.5.2 Notes on Stack Area Use ..... 53
3.5.3 Notes on Rewriting Port Mode Registers ..... 53

Section 4 Address Break ..... 55
4.1 Register Descriptions ..... 55
4.1.1 Address Break Control Register (ABRKCR) ..... 56
4.1.2 Address Break Status Register (ABRKSR) ..... 57
4.1.3 Break Address Registers (BARH, BARL) ..... 58
4.1.4 Break Data Registers (BDRH, BDRL) ..... 58
4.2 Operation ..... 59
4.3 Usage Notes ..... 61

Section 5 Clock Pulse Generators ..... 65
5.1 System Clock Generator ..... 65
5.1.1 Connecting Crystal Resonator ..... 66
5.1.2 Connecting Ceramic Resonator ..... 66
5.1.3 External Clock Input Method ..... 67
5.2 Prescalers ..... 67
5.2.1 Prescaler S ..... 67
5.3 Usage Notes ..... 67
5.3.1 Note on Resonators ..... 67
5.3.2 Notes on Board Design ..... 68

Section 6 Power-Down Modes ..... 69
6.1 Register Descriptions ..... 70
6.1.1 System Control Register 1 (SYSCR1) ..... 70
6.1.2 System Control Register 2 (SYSCR2) ..... 72
6.1.3 Module Standby Control Register 1 (MSTCR1) ..... 73
6.2 Mode Transitions and States of LSI ..... 74
6.2.1 Sleep Mode ..... 76
6.2.2 Standby Mode ..... 76

6.2.3 Subsleep Mode ... 76
6.3 Operating Frequency in Active Mode ... 77
6.4 Direct Transition ... 77
6.5 Module Standby Function ... 77

Section 7 ROM ... 79
7.1 Block Configuration ... 79
7.2 Register Descriptions ... 80
7.2.1 Flash Memory Control Register 1 (FLMCR1) ... 81
7.2.2 Flash Memory Control Register 2 (FLMCR2) ... 82
7.2.3 Erase Block Register 1 (EBR1) ... 82
7.2.4 Flash Memory Enable Register (FENR) ... 83
7.3 On-Board Programming Modes ... 83
7.3.1 Boot Mode ... 84
7.3.2 Programming/Erasing in User Program Mode ... 86
7.4 Flash Memory Programming/Erasing ... 87
7.4.1 Program/Program-Verify ... 87
7.4.2 Erase/Erase-Verify ... 89
7.4.3 Interrupt Handling when Programming/Erasing Flash Memory ... 90
7.5 Program/Erase Protection ... 92
7.5.1 Hardware Protection ... 92
7.5.2 Software Protection ... 92
7.5.3 Error Protection ... 92

Section 8 RAM ... 93

Section 9 I/O Ports ... 95
9.1 Port 1 ... 95
9.1.1 Port Mode Register 1 (PMR1) ... 96
9.1.2 Port Control Register 1 (PCR1) ... 97
9.1.3 Port Data Register 1 (PDR1) ... 97
9.1.4 Port Pull-Up Control Register 1 (PUCR1) ... 98
9.1.5 Pin Functions ... 98
9.2 Port 2 ... 100
9.2.1 Port Control Register 2 (PCR2) ... 100
9.2.2 Port Data Register 2 (PDR2) ... 101
9.2.3 Pin Functions ... 101
9.3 Port 5 ... 102
9.3.1 Port Mode Register 5 (PMR5) ... 103
9.3.2 Port Control Register 5 (PCR5) ... 104

9.3.3 Port Data Register 5 (PDR5) ... 104
9.3.4 Port Pull-Up Control Register 5 (PUCR5) ... 105
9.3.5 Pin Functions ... 105
9.4 Port 7 ... 107
9.4.1 Port Control Register 7 (PCR7) ... 108
9.4.2 Port Data Register 7 (PDR7) ... 108
9.4.3 Pin Functions ... 109
9.5 Port 8 ... 110
9.5.1 Port Control Register 8 (PCR8) ... 110
9.5.2 Port Data Register 8 (PDR8) ... 111
9.5.3 Pin Functions ... 111
9.6 Port B ... 113
9.6.1 Port Data Register B (PDRB) ... 114

Section 10 Timer V ... 115
10.1 Features ... 115
10.2 Input/Output Pins ... 117
10.3 Register Descriptions ... 117
10.3.1 Timer Counter V (TCNTV) ... 117
10.3.2 Time Constant Registers A and B (TCORA, TCORB) ... 118
10.3.3 Timer Control Register V0 (TCRV0) ... 118
10.3.4 Timer Control/Status Register V (TCSRV) ... 120
10.3.5 Timer Control Register V1 (TCRV1) ... 121
10.4 Operation ... 122
10.4.1 Timer V Operation ... 122
10.5 Timer V Application Examples ... 126
10.5.1 Pulse Output with Arbitrary Duty Cycle ... 126
10.5.2 Pulse Output with Arbitrary Pulse Width and Delay from TRGV Input ... 127
10.6 Usage Notes ... 128

Section 11 Timer W ... 131
11.1 Features ... 131
11.2 Input/Output Pins ... 134
11.3 Register Descriptions ... 134
11.3.1 Timer Mode Register W (TMRW) ... 135
11.3.2 Timer Control Register W (TCRW) ... 136
11.3.3 Timer Interrupt Enable Register W (TIERW) ... 137
11.3.4 Timer Status Register W (TSRW) ... 137
11.3.5 Timer I/O Control Register 0 (TIOR0) ... 139
11.3.6 Timer I/O Control Register 1 (TIOR1) ... 140

11.3.7 Timer Counter (TCNT) ..... 141
11.3.8 General Registers A to D (GRA to GRD) ..... 141
11.4 Operation ..... 142
11.4.1 Normal Operation ..... 142
11.4.2 PWM Operation ..... 146
11.5 Operation Timing ..... 151
11.5.1 TCNT Count Timing ..... 151
11.5.2 Output Compare Output Timing ..... 151
11.5.3 Input Capture Timing ..... 152
11.5.4 Timing of Counter Clearing by Compare Match ..... 153
11.5.5 Buffer Operation Timing ..... 153
11.5.6 Timing of IMFA to IMFD Flag Setting at Compare Match ..... 154
11.5.7 Timing of IMFA to IMFD Setting at Input Capture ..... 155
11.5.8 Timing of Status Flag Clearing ..... 155
11.6 Usage Notes ..... 156

Section 12 Watchdog Timer ..... 159
12.1 Features ..... 159
12.2 Register Descriptions ..... 159
12.2.1 Timer Control/Status Register WD (TCSRWD) ..... 160
12.2.2 Timer Counter WD (TCWD) ..... 161
12.2.3 Timer Mode Register WD (TMWD) ..... 161
12.3 Operation ..... 162

Section 13 Serial Communication Interface 3 (SCI3) ..... 163
13.1 Features ..... 163
13.2 Input/Output Pins ..... 165
13.3 Register Descriptions ..... 165
13.3.1 Receive Shift Register (RSR) ..... 166
13.3.2 Receive Data Register (RDR) ..... 166
13.3.3 Transmit Shift Register (TSR) ..... 166
13.3.4 Transmit Data Register (TDR) ..... 166
13.3.5 Serial Mode Register (SMR) ..... 167
13.3.6 Serial Control Register 3 (SCR3) ..... 168
13.3.7 Serial Status Register (SSR) ..... 170
13.3.8 Bit Rate Register (BRR) ..... 172
13.4 Operation in Asynchronous Mode ..... 177
13.4.1 Clock ..... 177
13.4.2 SCI3 Initialization ..... 178
13.4.3 Data Transmission ..... 179

13.4.4 Serial Data Reception ..... 181
13.5 Operation in Clocked Synchronous Mode ..... 185
13.5.1 Clock ..... 185
13.5.2 SCI3 Initialization ..... 185
13.5.3 Serial Data Transmission ..... 186
13.5.4 Serial Data Reception (Clocked Synchronous Mode) ..... 189
13.5.5 Simultaneous Serial Data Transmission and Reception ..... 191
13.6 Multiprocessor Communication Function ..... 193
13.6.1 Multiprocessor Serial Data Transmission ..... 195
13.6.2 Multiprocessor Serial Data Reception ..... 196
13.7 Interrupts ..... 200
13.8 Usage Notes ..... 201
13.8.1 Break Detection and Processing ..... 201
13.8.2 Mark State and Break Sending ..... 201
13.8.3 Receive Error Flags and Transmit Operations (Clocked Synchronous Mode Only) ..... 201
13.8.4 Receive Data Sampling Timing and Reception Margin in Asynchronous Mode ..... 202

Section 14 A/D Converter ..... 203
14.1 Features ..... 203
14.2 Input/Output Pins ..... 205
14.3 Register Description ..... 206
14.3.1 A/D Data Registers A to D (ADDRA to ADDRD) ..... 206
14.3.2 A/D Control/Status Register (ADCSR) ..... 207
14.3.3 A/D Control Register (ADCR) ..... 208
14.4 Operation ..... 209
14.4.1 Single Mode ..... 209
14.4.2 Scan Mode ..... 209
14.4.3 Input Sampling and A/D Conversion Time ..... 210
14.4.4 External Trigger Input Timing ..... 211
14.5 A/D Conversion Accuracy Definitions ..... 212
14.6 Usage Notes ..... 214
14.6.1 Permissible Signal Source Impedance ..... 214
14.6.2 Influences on Absolute Accuracy ..... 214

Section 15 Power Supply Circuit ..... 215
15.1 When Using Internal Power Supply Step-Down Circuit ..... 215
15.2 When Not Using Internal Power Supply Step-Down Circuit ..... 216

Section 16 List of Registers ..... 217
16.1 Register Addresses (Address Order) ..... 218
16.2 Register Bits ..... 221
16.3 Register States in Each Operating Mode ..... 224

Section 17 Electrical Characteristics ..... 227
17.1 Absolute Maximum Ratings ..... 227
17.2 Electrical Characteristics ..... 227
17.2.1 Power Supply Voltage and Operating Ranges ..... 227
17.2.2 DC Characteristics ..... 230
17.2.3 AC Characteristics ..... 235
17.2.4 A/D Converter Characteristics ..... 238
17.2.5 Watchdog Timer ..... 239
17.2.6 Flash Memory Characteristics ..... 240
17.3 Operation Timing ..... 242
17.4 Output Load Condition ..... 244

Appendix A Instruction Set ..... 245
A.1 Instruction List ..... 245
A.2 Operation Code Map ..... 260
A.3 Number of Execution States ..... 263
A.4 Combinations of Instructions and Addressing Modes ..... 274

Appendix B I/O Port Block Diagrams ..... 276
B.1 I/O Port Block ..... 276
B.2 Port States in Each Operating State ..... 291

Appendix C Product Code Lineup ..... 292

Appendix D Package Dimensions ..... 293

Main Revisions and Additions in this Edition ..... 297

Index ..... 301

# Figures

**Section 1 Overview**
Figure 1.1 Internal Block Diagram ..... 2
Figure 1.2 Pin Arrangement (FP-64E) ..... 3
Figure 1.3 Pin Arrangement (FP-48F, FP-48B) ..... 4

**Section 2 CPU**
Figure 2.1 Memory Map ..... 8
Figure 2.2 CPU Registers ..... 9
Figure 2.3 Usage of General Registers ..... 10
Figure 2.4 Relationship between Stack Pointer and Stack Area ..... 11
Figure 2.5 General Register Data Formats (1) ..... 13
Figure 2.5 General Register Data Formats (2) ..... 14
Figure 2.6 Memory Data Formats ..... 15
Figure 2.7 Instruction Formats ..... 26
Figure 2.8 Branch Address Specification in Memory Indirect Mode ..... 30
Figure 2.9 On-Chip Memory Access Cycle ..... 32
Figure 2.10 On-Chip Peripheral Module Access Cycle (3-State Access) ..... 33
Figure 2.11 CPU Operation States ..... 34
Figure 2.12 State Transitions ..... 35
Figure 2.13 Example of Timer Configuration with Two Registers Allocated to Same Address ..... 36

**Section 3 Exception Handling**
Figure 3.1 Reset Sequence ..... 49
Figure 3.2 Stack Status after Exception Handling ..... 51
Figure 3.3 Interrupt Sequence ..... 52
Figure 3.4 Port Mode Register Setting and Interrupt Request Flag Clearing Procedure ..... 53

**Section 4 Address Break**
Figure 4.1 Block Diagram of Address Break ..... 55
Figure 4.2 Address Break Interrupt Operation Example (1) ..... 59
Figure 4.2 Address Break Interrupt Operation Example (2) ..... 60
Figure 4.3 Operation when Condition is not Satisfied in Branch Instruction ..... 61
Figure 4.4 Operation when Another Interrupt is Accepted at Address Break Setting Instruction ..... 62
Figure 4.5 Operation when the Instruction Set is not Executed and does not Branch due to Conditions not Being Satisfied ..... 63

**Section 5 Clock Pulse Generators**

Figure 5.1 Block Diagram of Clock Pulse Generators ..... 65
Figure 5.2 Block Diagram of System Clock Generator ..... 65
Figure 5.3 Typical Connection to Crystal Resonator ..... 66
Figure 5.4 Equivalent Circuit of Crystal Resonator ..... 66
Figure 5.5 Typical Connection to Ceramic Resonator ..... 66
Figure 5.6 Example of External Clock Input ..... 67
Figure 5.7 Example of Incorrect Board Design ..... 68

**Section 6 Power-Down Modes**

Figure 6.1 Mode Transition Diagram ..... 74

**Section 7 ROM**

Figure 7.1 Flash Memory Block Configuration ..... 80
Figure 7.2 Programming/Erasing Flowchart Example in User Program Mode ..... 86
Figure 7.3 Program/Program-Verify Flowchart ..... 88
Figure 7.4 Erase/Erase-Verify Flowchart ..... 91

**Section 9 I/O Ports**

Figure 9.1 Port 1 Pin Configuration ..... 95
Figure 9.2 Port 2 Pin Configuration ..... 100
Figure 9.3 Port 5 Pin Configuration ..... 102
Figure 9.4 Port 7 Pin Configuration ..... 107
Figure 9.5 Port 8 Pin Configuration ..... 110
Figure 9.6 Port B Pin Configuration ..... 113

**Section 10 Timer V**

Figure 10.1 Block Diagram of Timer V ..... 116
Figure 10.2 Increment Timing with Internal Clock ..... 123
Figure 10.3 Increment Timing with External Clock ..... 123
Figure 10.4 OVF Set Timing ..... 123
Figure 10.5 CMFA and CMFB Set Timing ..... 124
Figure 10.6 TMOV Output Timing ..... 124
Figure 10.7 Clear Timing by Compare Match ..... 124
Figure 10.8 Clear Timing by TMRIV Input ..... 125
Figure 10.9 Pulse Output Example ..... 126
Figure 10.10 Example of Pulse Output Synchronized to TRGV Input ..... 127
Figure 10.11 Contention between TCNTV Write and Clear ..... 128
Figure 10.12 Contention between TCORA Write and Compare Match ..... 129
Figure 10.13 Internal Clock Switching and TCNTV Operation ..... 129

**Section 11 Timer W**

Figure 11.1 Timer W Block Diagram ..... 133

Figure 11.2 Free-Running Counter Operation ..... 142
Figure 11.3 Periodic Counter Operation ..... 143
Figure 11.4 0 and 1 Output Example (TOA = 0, TOB = 1) ..... 143
Figure 11.5 Toggle Output Example (TOA = 0, TOB = 1) ..... 144
Figure 11.6 Toggle Output Example (TOA = 0, TOB = 1) ..... 144
Figure 11.7 Input Capture Operating Example ..... 145
Figure 11.8 Buffer Operation Example (Input Capture) ..... 146
Figure 11.9 PWM Mode Example (1) ..... 147
Figure 11.10 PWM Mode Example (2) ..... 147
Figure 11.11 Buffer Operation Example (Output Compare) ..... 148
Figure 11.12 PWM Mode Example
(TOB, TOC, and TOD = 0: initial output values are set to 0) ..... 149
Figure 11.13 PWM Mode Example
(TOB, TOC, and TOD = 1: initial output values are set to 1) ..... 150
Figure 11.14 Count Timing for Internal Clock Source ..... 151
Figure 11.15 Count Timing for External Clock Source ..... 151
Figure 11.16 Output Compare Output Timing ..... 152
Figure 11.17 Input Capture Input Signal Timing ..... 152
Figure 11.18 Timing of Counter Clearing by Compare Match ..... 153
Figure 11.19 Buffer Operation Timing (Compare Match) ..... 153
Figure 11.20 Buffer Operation Timing (Input Capture) ..... 154
Figure 11.21 Timing of IMFA to IMFD Flag Setting at Compare Match ..... 154
Figure 11.22 Timing of IMFA to IMFD Flag Setting at Input Capture ..... 155
Figure 11.23 Timing of Status Flag Clearing by CPU ..... 155
Figure 11.24 Contention between TCNT Write and Clear ..... 156
Figure 11.25 Internal Clock Switching and TCNT Operation ..... 157
Figure 11.26 When Compare Match and Bit Manipulation Instruction to TCRW
Occur at the Same Timing ..... 158

**Section 12 Watchdog Timer**

Figure 12.1 Block Diagram of Watchdog Timer ..... 159
Figure 12.2 Watchdog Timer Operation Example ..... 162

**Section 13 Serial Communication Interface 3 (SCI3)**

Figure 13.1 Block Diagram of SCI3 ..... 164
Figure 13.2 Data Format in Asynchronous Communication ..... 177
Figure 13.3 Relationship between Output Clock and Transfer Data Phase
(Asynchronous Mode) (Example with 8-Bit Data, Parity, Two Stop Bits) ..... 177
Figure 13.4 Sample SCI3 Initialization Flowchart ..... 178
Figure 13.5 Example SCI3 Operation in Transmission in Asynchronous Mode
(8-Bit Data, Parity, One Stop Bit) ..... 179

Figure 13.6 Sample Serial Transmission Flowchart (Asynchronous Mode) ..... 180
Figure 13.7 Example SCI3 Operation in Reception in Asynchronous Mode (8-Bit Data, Parity, One Stop Bit) ..... 181
Figure 13.8 Sample Serial Data Reception Flowchart (Asynchronous mode) (1) ..... 183
Figure 13.8 Sample Serial Reception Data Flowchart (2) ..... 184
Figure 13.9 Data Format in Clocked Synchronous Communication ..... 185
Figure 13.10 Example of SCI3 Operation in Transmission in Clocked Synchronous Mode ..... 187
Figure 13.11 Sample Serial Transmission Flowchart (Clocked Synchronous Mode) ..... 188
Figure 13.12 Example of SCI3 Reception Operation in Clocked Synchronous Mode ..... 189
Figure 13.13 Sample Serial Reception Flowchart (Clocked Synchronous Mode) ..... 190
Figure 13.14 Sample Flowchart of Simultaneous Serial Transmit and Receive Operations (Clocked Synchronous Mode) ..... 192
Figure 13.15 Example of Communication Using Multiprocessor Format (Transmission of Data H'AA to Receiving Station A) ..... 194
Figure 13.16 Sample Multiprocessor Serial Transmission Flowchart ..... 195
Figure 13.17 Sample Multiprocessor Serial Reception Flowchart (1) ..... 197
Figure 13.17 Sample Multiprocessor Serial Reception Flowchart (2) ..... 198
Figure 13.18 Example of SCI3 Operation in Reception Using Multiprocessor Format (Example with 8-Bit Data, Multiprocessor Bit, One Stop Bit) ..... 199
Figure 13.19 Receive Data Sampling Timing in Asynchronous Mode ..... 202

**Section 14 A/D Converter**

Figure 14.1 Block Diagram of A/D Converter ..... 204
Figure 14.2 A/D Conversion Timing ..... 210
Figure 14.3 External Trigger Input Timing ..... 211
Figure 14.4 A/D Conversion Accuracy Definitions (1) ..... 213
Figure 14.5 A/D Conversion Accuracy Definitions (2) ..... 213
Figure 14.6 Analog Input Circuit Example ..... 214

**Section 15 Power Supply Circuit**

Figure 15.1 Power Supply Connection when Internal Step-Down Circuit is Used ..... 215
Figure 15.2 Power Supply Connection when Internal Step-Down Circuit is Not Used ..... 216

**Section 17 Electrical Characteristics**

Figure 17.1 System Clock Input Timing ..... 242
Figure 17.2 $\overline{\text{RES}}$ Low Width Timing ..... 242
Figure 17.3 Input Timing ..... 242
Figure 17.4 SCK3 Input Clock Timing ..... 243
Figure 17.5 SCI3 Input/Output Timing in Clocked Synchronous Mode ..... 243
Figure 17.6 Output Load Circuit ..... 244

**Appendix B I/O Port Block Diagrams**

Figure B.1 Port 1 Block Diagram (P17) ..... 276
Figure B.2 Port 1 Block Diagram (P14) ..... 277
Figure B.3 Port 1 Block Diagram (P16, P15, P12, P10) ..... 278
Figure B.4 Port 1 Block Diagram (P11) ..... 279
Figure B.5 Port 2 Block Diagram (P22) ..... 280
Figure B.6 Port 2 Block Diagram (P21) ..... 281
Figure B.7 Port 2 Block Diagram (P20) ..... 282
Figure B.8 Port 5 Block Diagram (P57, P56) ..... 283
Figure B.9 Port 5 Block Diagram (P55) ..... 284
Figure B.10 Port 5 Block Diagram (P54 to P50) ..... 285
Figure B.11 Port 7 Block Diagram (P76) ..... 286
Figure B.12 Port 7 Block Diagram (P75) ..... 287
Figure B.13 Port 7 Block Diagram (P74) ..... 288
Figure B.14 Port 8 Block Diagram (P84 to P81) ..... 289
Figure B.15 Port 8 Block Diagram (P80) ..... 290
Figure B.16 Port B Block Diagram (PB3 to PB0) ..... 291

**Appendix D Package Dimensions**

Figure D.1 FP-64E Package Dimensions ..... 293
Figure D.2 FP-48F Package Dimensions ..... 294
Figure D.3 FP-48B Package Dimensions ..... 295

# Tables

**Section 1 Overview**
Table 1.1 Pin Functions ..... 5

**Section 2 CPU**
Table 2.1 Operation Notation ..... 16
Table 2.2 Data Transfer Instructions ..... 17
Table 2.3 Arithmetic Operations Instructions (1) ..... 18
Table 2.3 Arithmetic Operations Instructions (2) ..... 19
Table 2.4 Logic Operations Instructions ..... 20
Table 2.5 Shift Instructions ..... 20
Table 2.6 Bit Manipulation Instructions (1) ..... 21
Table 2.6 Bit Manipulation Instructions (2) ..... 22
Table 2.7 Branch Instructions ..... 23
Table 2.8 System Control Instructions ..... 24
Table 2.9 Block Data Transfer Instructions ..... 25
Table 2.10 Addressing Modes ..... 27
Table 2.11 Absolute Address Access Ranges ..... 29
Table 2.12 Effective Address Calculation (1) ..... 30
Table 2.12 Effective Address Calculation (2) ..... 31

**Section 3 Exception Handling**
Table 3.1 Exception Sources and Vector Address ..... 42
Table 3.2 Interrupt Wait States ..... 51

**Section 4 Address Break**
Table 4.1 Access and Data Bus Used ..... 57

**Section 5 Clock Pulse Generators**
Table 5.1 Crystal Resonator Parameters ..... 66

**Section 6 Power-Down Modes**
Table 6.1 Operating Frequency and Waiting Time ..... 71
Table 6.2 Transition Mode after SLEEP Instruction Execution and Interrupt Handling ..... 75
Table 6.3 Internal State in Each Operating Mode ..... 75

**Section 7 ROM**
Table 7.1 Setting Programming Modes ..... 83
Table 7.2 Boot Mode Operation ..... 85
Table 7.3 System Clock Frequencies for which Automatic Adjustment of LSI Bit Rate is Possible ..... 86
Table 7.4 Reprogram Data Computation Table ..... 89

Table 7.5 Additional-Program Data Computation Table ..... 89
Table 7.6 Programming Time ..... 89

**Section 10 Timer V**
Table 10.1 Pin Configuration ..... 117
Table 10.2 Clock Signals to Input to TCNTV and Counting Conditions ..... 119

**Section 11 Timer W**
Table 11.1 Timer W Functions ..... 132
Table 11.2 Pin Configuration ..... 134

**Section 13 Serial Communication Interface 3 (SCI3)**
Table 13.1 Pin Configuration ..... 165
Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (1) ..... 173
Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (2) ..... 174
Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (3) ..... 175
Table 13.3 Maximum Bit Rate for Each Frequency (Asynchronous Mode) ..... 175
Table 13.4 BRR Settings for Various Bit Rates (Clocked Synchronous Mode) ..... 176
Table 13.5 SSR Status Flags and Receive Data Handling ..... 182
Table 13.6 SCI3 Interrupt Requests ..... 200

**Section 14 A/D Converter**
Table 14.1 Pin Configuration ..... 205
Table 14.2 Analog Input Channels and Corresponding ADDR Registers ..... 206
Table 14.3 A/D Conversion Time (Single Mode) ..... 211

**Section 17 Electrical Characteristics**
Table 17.1 Absolute Maximum Ratings ..... 227
Table 17.2 DC Characteristics (1) ..... 230
Table 17.2 DC Characteristics (2) ..... 234
Table 17.3 AC Characteristics ..... 235
Table 17.4 Serial Interface (SCI3) Timing ..... 237
Table 17.5 A/D Converter Characteristics ..... 238
Table 17.6 Watchdog Timer Characteristics ..... 239
Table 17.7 Flash Memory Characteristics ..... 240

**Appendix A Instruction Set**
Table A.1 Instruction Set ..... 247
Table A.2 Operation Code Map (1) ..... 260
Table A.2 Operation Code Map (2) ..... 261
Table A.2 Operation Code Map (3) ..... 262
Table A.3 Number of Cycles in Each Instruction ..... 264
Table A.4 Number of Cycles in Each Instruction ..... 265
Table A.5 Combinations of Instructions and Addressing Modes ..... 275

# Section 1 Overview

## 1.1 Features

- High-speed H8/300H central processing unit with an internal 16-bit architecture
  - Upward-compatible with H8/300 CPU on an object level
  - Sixteen 16-bit general registers
  - 62 basic instructions
- Various peripheral functions
  - Timer V (8-bit timer)
  - Timer W (16-bit timer)
  - Watchdog timer
  - SCI3 (Asynchronous or clocked synchronous serial communication interface)
  - 10-bit A/D converter
- On-chip memory

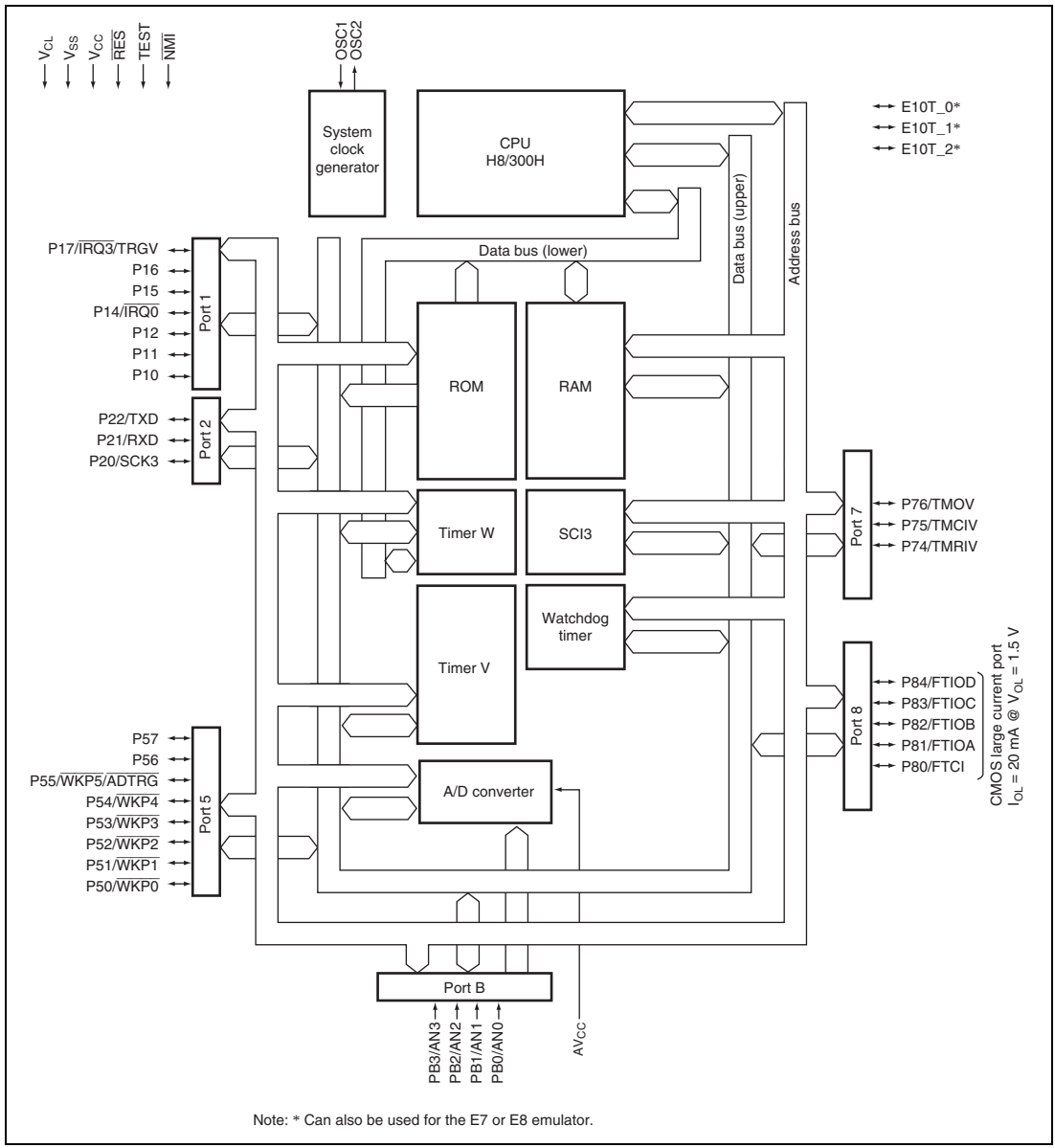
| Product Classification | | Model | ROM | RAM |
|---|---|---|---|---|
| Flash memory version | H8/3672 | HD64F3672 | 16 kbytes | 2,048 bytes |
| (F-ZTAT™ version) | H8/3670 | HD64F3670 | 8 kbytes | 2,048 bytes |

- General I/O ports
  - I/O pins: 26 I/O pins, including 5 large current ports ($I_{OL}$ = 20 mA, @$V_{OL}$ = 1.5 V)
  - Input-only pins: 4 input pins (also used for analog input)
- Supports various power-down modes

  Note: F-ZTAT™ is a trademark of Renesas Technology Corp.
- Compact package

| Package | Code | Body Size | Pin Pitch |
|---|---|---|---|
| LQFP-64 | FP-64E | 10.0 × 10.0 mm | 0.5 mm |
| LQFP-48 | FP-48F | 10.0 × 10.0 mm | 0.65 mm |
| LQFP-48 | FP-48B | 7.0 × 7.0 mm | 0.5 mm |

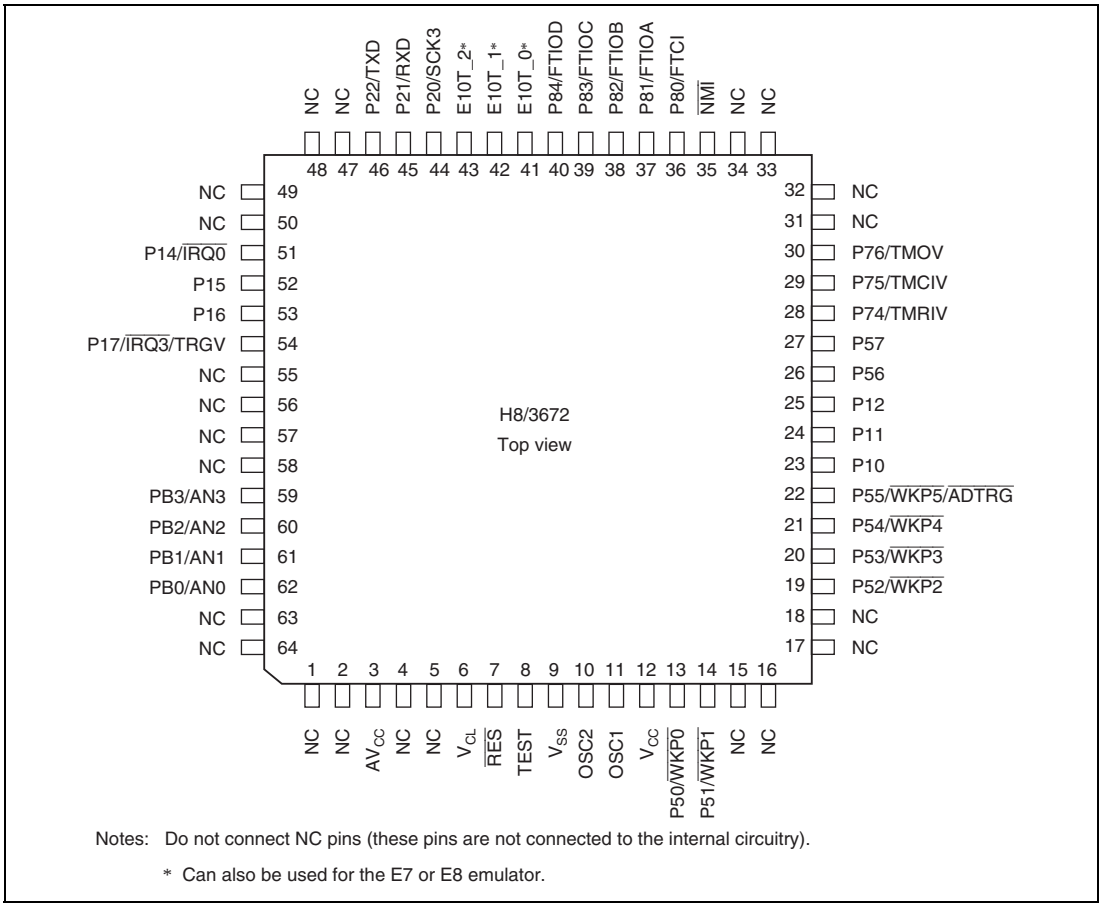
## 1.2 Internal Block Diagram




Note: * Can also be used for the E7 or E8 emulator.

**Figure 1.1 Internal Block Diagram**

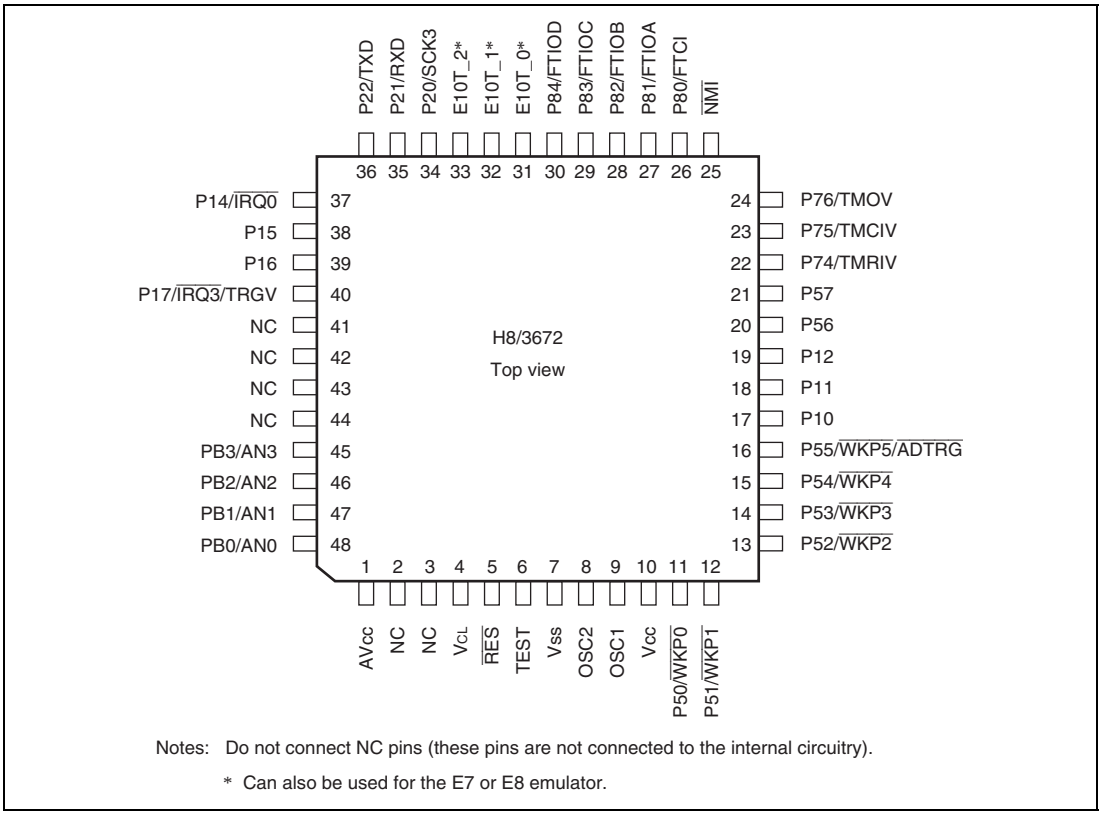
## 1.3 Pin Arrangement




Notes: Do not connect NC pins (these pins are not connected to the internal circuitry).

* Can also be used for the E7 or E8 emulator.

**Figure 1.2 Pin Arrangement (FP-64E)**




Notes: Do not connect NC pins (these pins are not connected to the internal circuitry).

* Can also be used for the E7 or E8 emulator.

**Figure 1.3 Pin Arrangement (FP-48F, FP-48B)**

## 1.4 Pin Functions

**Table 1.1 Pin Functions**

| Type | Symbol | Pin No. FP-64E | Pin No. FP-48F, FP-48B | I/O | Functions |
|---|---|---|---|---|---|
| Power source pins | $V_{CC}$ | 12 | 10 | Input | Power supply pin. Connect this pin to the system power supply. |
| | $V_{SS}$ | 9 | 7 | Input | Ground pin. Connect this pin to the system power supply (0V). |
| | $AV_{CC}$ | 3 | 1 | Input | Analog power supply pin for the A/D converter. When the A/D converter is not used, connect this pin to the system power supply. |
| | $V_{CL}$ | 6 | 4 | Input | Internal step-down power supply pin. Connect a capacitor of around 0.1 μF between this pin and the Vss pin for stabilization. |
| Clock pins | OSC1 | 11 | 9 | Input | These pins connect to a crystal or ceramic resonator for system clocks, or can be used to input an external clock.<br>See section 5, Clock Pulse Generators, for a typical connection. |
| | OSC2 | 10 | 8 | Output | |
| System control | $\overline{RES}$ | 7 | 5 | Input | Reset pin. When this driven low, the chip is reset. |
| | TEST | 8 | 6 | Input | Test pin. Connect this pin to Vss. |
| Interrupt pins | $\overline{NMI}$ | 35 | 25 | Input | Non-maskable interrupt request input pin. Be sure to pull-up by a pull-up resistor. |
| | $\overline{IRQ0}$, $\overline{IRQ3}$ | 51, 54 | 37, 40 | Input | External interrupt request input pins. Can select the rising or falling edge. |
| | $\overline{WKP0}$ to $\overline{WKP5}$ | 13, 14, 19 to 22 | 11 to 16 | Input | External interrupt request input pins. Can select the rising or falling edge. |

| Type | Symbol | Pin No. FP-64E | Pin No. FP-48F, FP-48B | I/O | Functions |
|---|---|---|---|---|---|
| Timer V | TMOV | 30 | 24 | Output | This is an output pin for waveforms generated by the output compare function. |
| | TMCIV | 29 | 23 | Input | External event input pin. |
| | TMRIV | 28 | 22 | Input | Counter reset input pin. |
| | TRGV | 54 | 40 | Input | Counter start trigger input pin. |
| Timer W | FTCI | 36 | 26 | Input | External event input pin. |
| | FTIOA to FTIOD | 37 to 40 | 27 to 30 | I/O | Output compare output/input capture input/PWM output pin |
| Serial communication interface (SCI) | TXD | 46 | 36 | Output | Transmit data output pin |
| | RXD | 45 | 35 | Input | Receive data input pin |
| | SCK3 | 44 | 34 | I/O | Clock I/O pin |
| A/D converter | AN3 to AN0 | 59 to 62 | 45 to 48 | Input | Analog input pin |
| | $\overline{\text{ADTRG}}$ | 22 | 16 | Input | A/D converter trigger input pin. |
| I/O ports | PB3 to PB0 | 59 to 62 | 45 to 48 | Input | 4-bit input port. |
| | P17 to P14, P12 to P10 | 54 to 51, 25 to 23 | 40 to 37, 19 to 17 | I/O | 7-bit I/O port. |
| | P22 to P20 | 46 to 44 | 36 to 34 | I/O | 3-bit I/O port. |
| | P57 to P50 | 27, 26, 22 to 19, 14, 13 | 21, 20, 16 to 11 | I/O | 8-bit I/O port |
| | P76 to P74 | 30 to 28 | 24 to 22 | I/O | 3-bit I/O port |
| | P84 to P80 | 40 to 36 | 30 to 26 | I/O | 5-bit I/O port. |
| E10T | E10T_0, E10T_1, E10T_2 | 41, 42, 43 | 31, 32, 33 | | Interface pin for the E10T, E7, or E8 emulator |

# Section 2 CPU

This LSI has an H8/300H CPU with an internal 32-bit architecture that is upward-compatible with the H8/300CPU, 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 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 address space of this LSI is 64 kbytes, which includes the program area and the data area. Figure 2.1 shows the memory map.

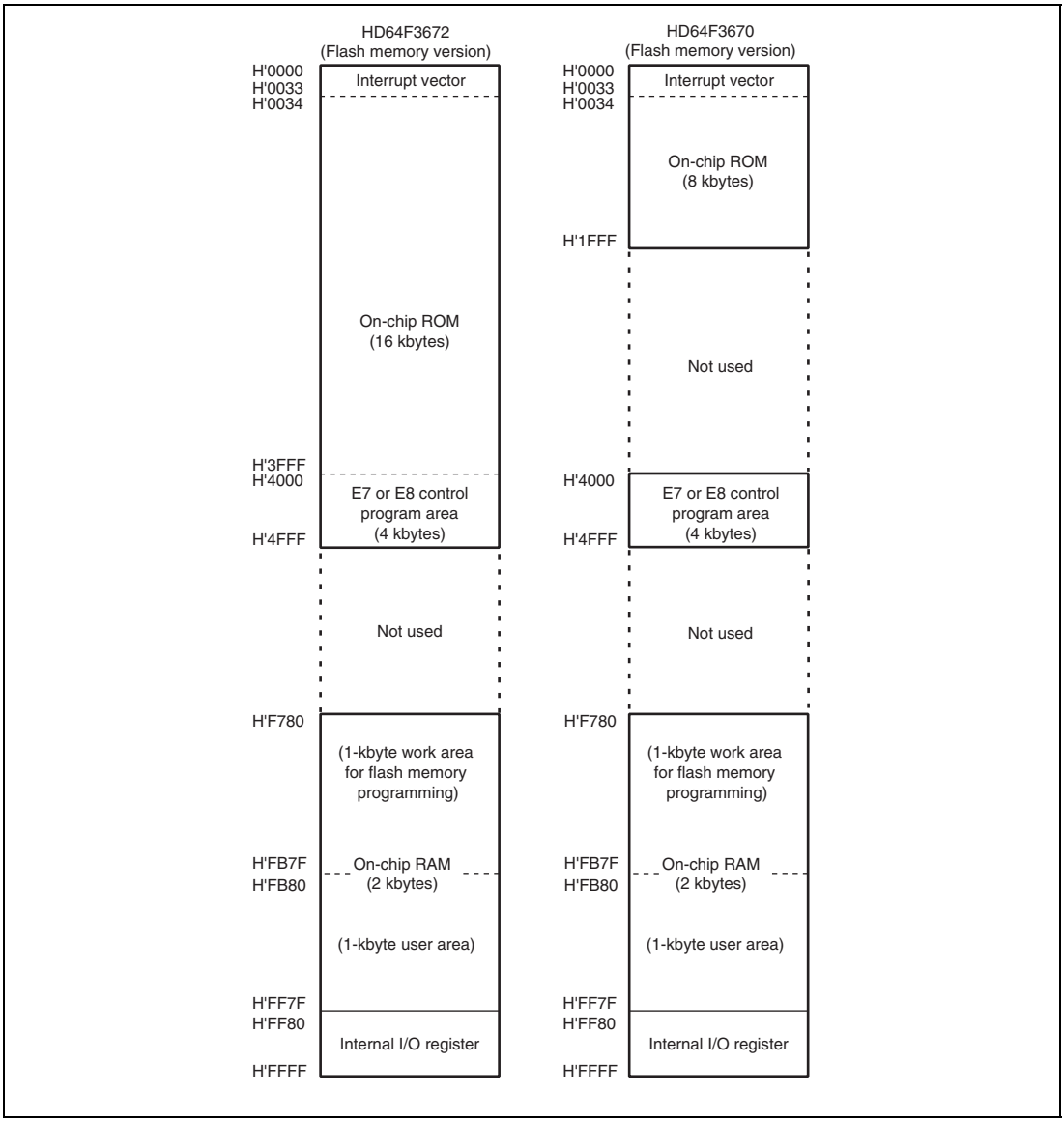



**Figure 2.1 Memory Map**

## 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).

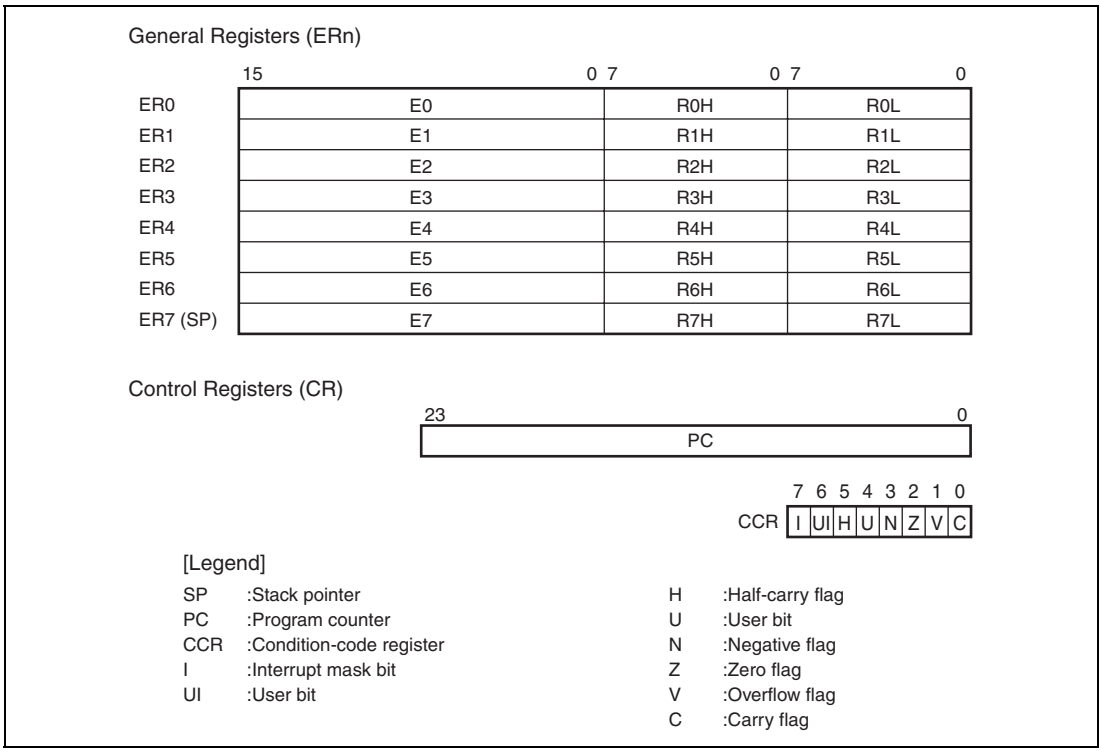



**Figure 2.2 CPU Registers**

### 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.

General register ER7 has the function of 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 stack.

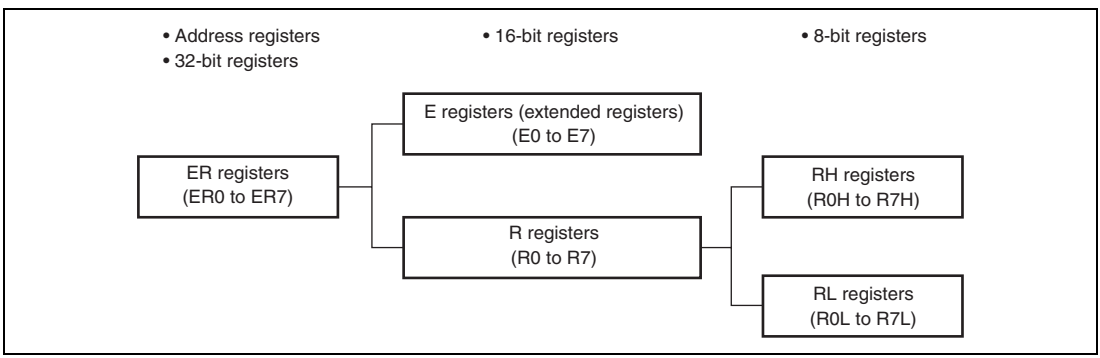



**Figure 2.3 Usage of General Registers**

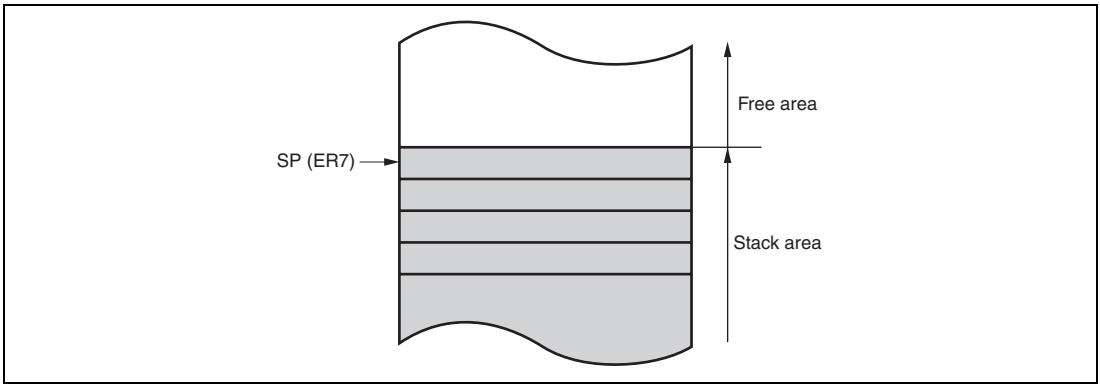



**Figure 2.4 Relationship between Stack Pointer and Stack Area**

### 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.

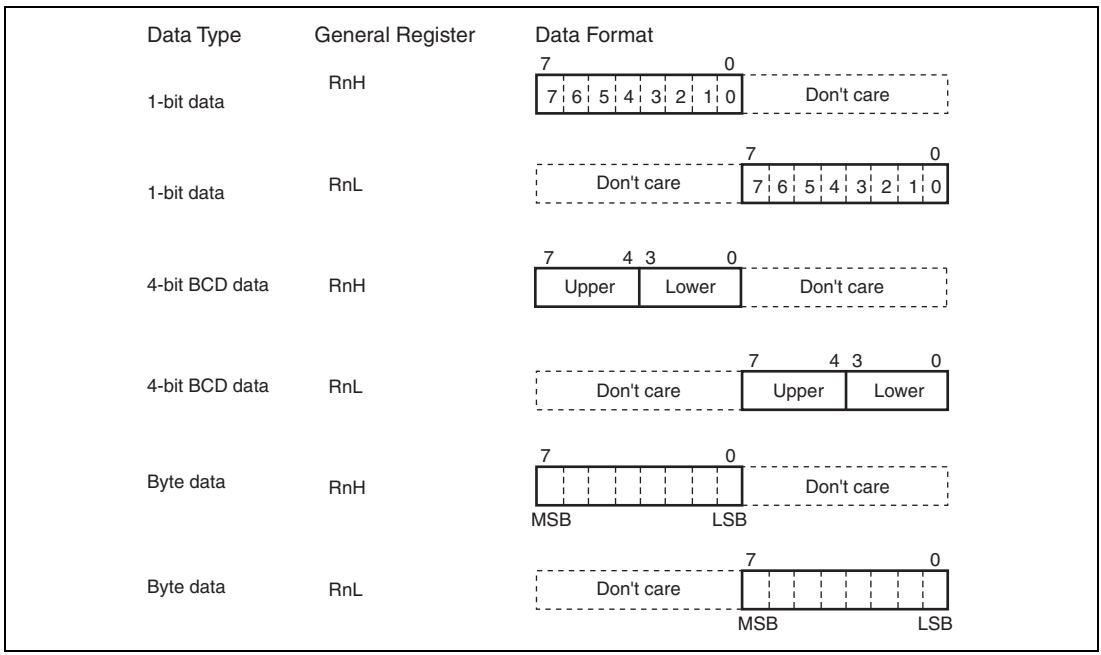



**Figure 2.5 General Register Data Formats (1)**

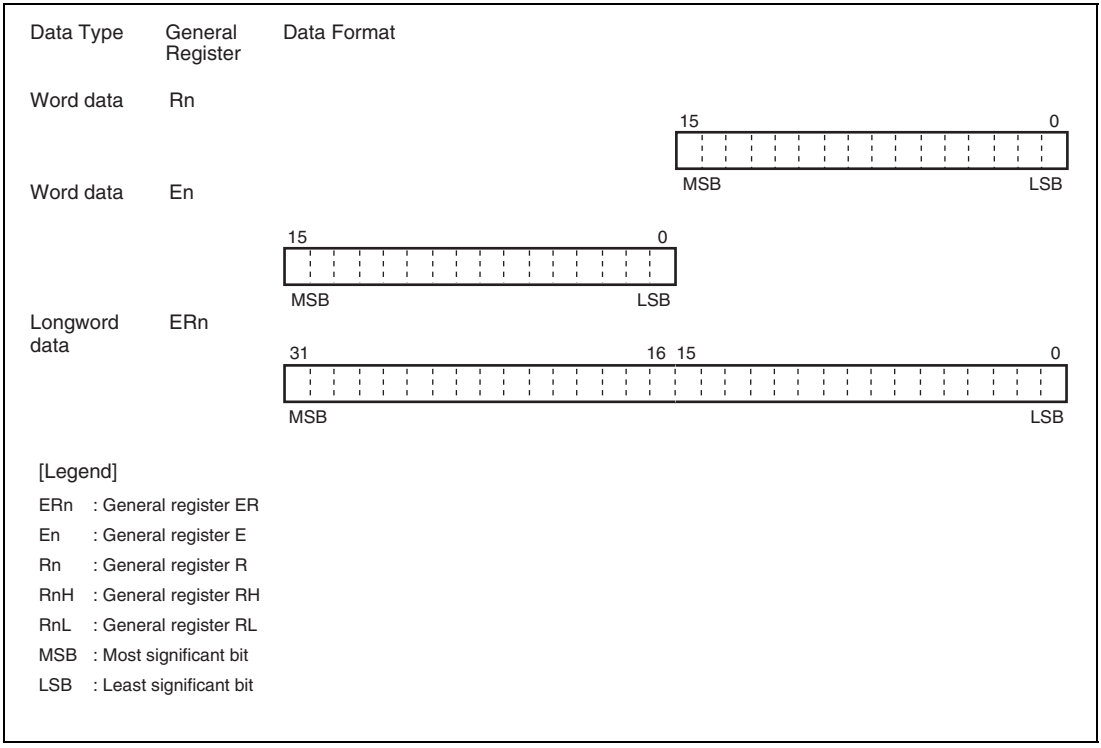



**Figure 2.5 General Register Data Formats (2)**

### 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, the operand size should be word or longword.

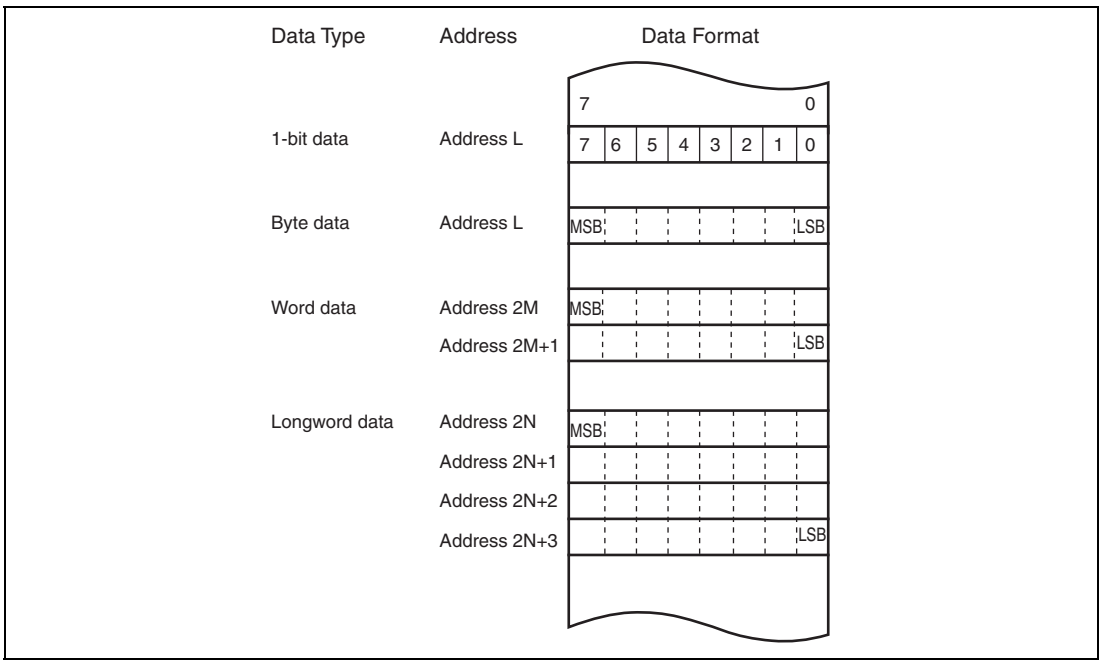



**Figure 2.6 Memory Data Formats**

## 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 below.

**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) |
| :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 registers (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, Cannot be used in this LSI. |
| MOVTPE | B | Rs → (EAs) 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 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. 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. 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. 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. 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. 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 |
|---|---|---|
| TRAPA | — | Starts trap-instruction exception handling. |
| 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), EXR → (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, EXR ∧ #IMM → EXR<br>Logically ANDs the CCR with immediate data. |
| ORC | B | CCR ∨ #IMM → CCR, EXR ∨ #IMM → EXR<br>Logically ORs the CCR with immediate data. |
| XORC | B | CCR ⊕ #IMM → CCR, EXR ⊕ #IMM → EXR<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 field), a register field (r field), an effective address extension (EA field), and a condition field (cc).

Figure 2.7 shows examples of instruction formats.

- 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.

- 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.

- 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).

- Condition Field

  Specifies the branching condition of Bcc instructions.

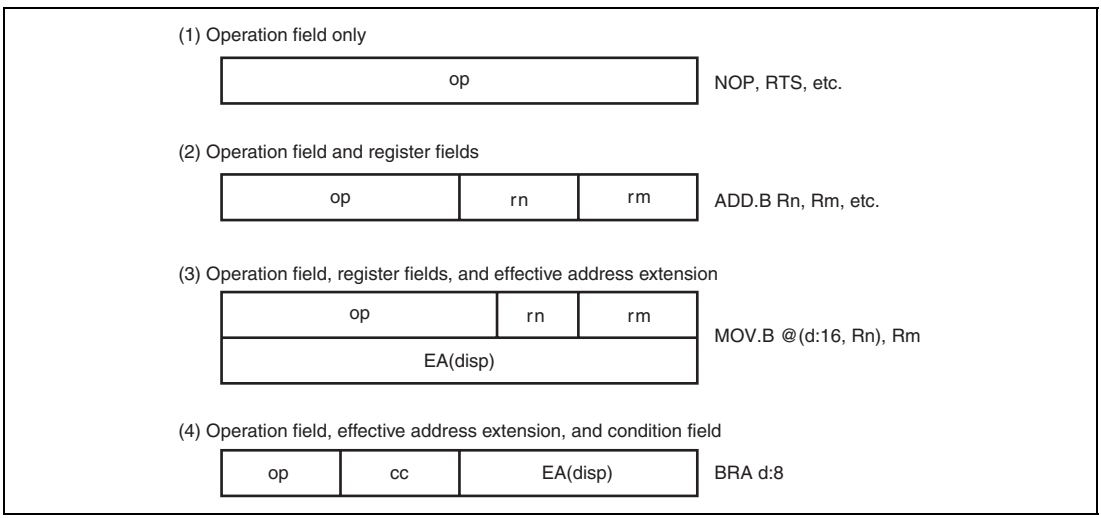



**Figure 2.7 Instruction Formats**

## 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 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 |

**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.

**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.

**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.

**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.

**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 the series of 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 |

**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. The TRAPA instruction contains 2-bit immediate data in its instruction code, specifying a vector address.

**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.

**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 by longword access. The first byte of the memory operand is ignored, generating a 24-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.

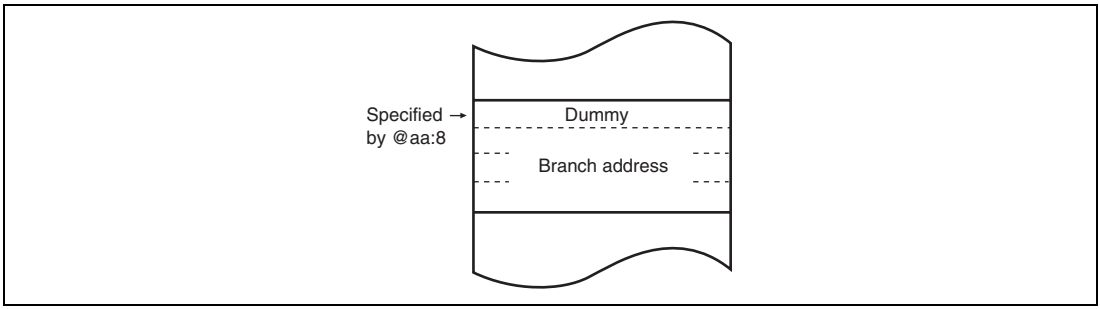



**Figure 2.8 Branch Address Specification in Memory Indirect Mode**

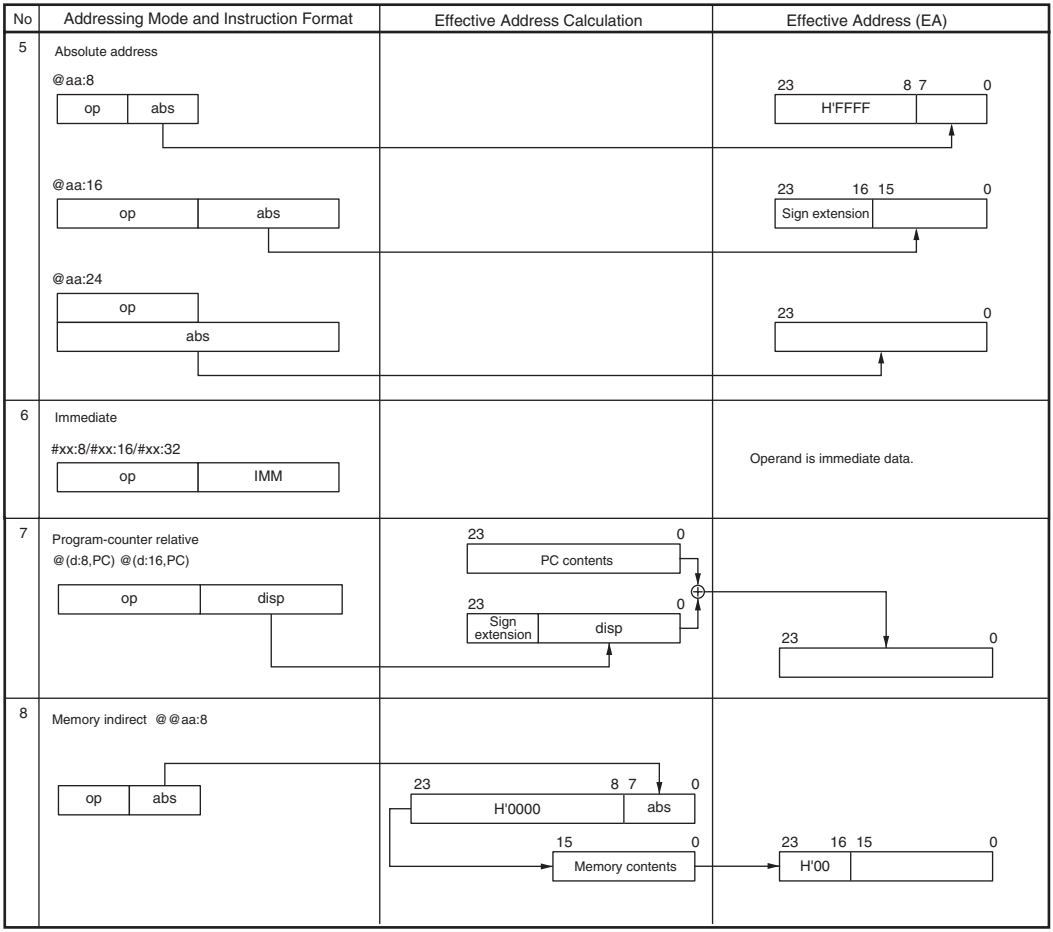
### 2.5.2 Effective Address Calculation

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<br>General register contents | 23 0 |
| 3 | Register indirect with displacement<br>@(d:16,ERn) or @(d:24,ERn)<br>op r disp | 31 0<br>General register contents<br>31 0<br>Sign extension disp | 23 0 |
| 4 | Register indirect with post-increment or pre-decrement<br>•Register indirect with post-increment @ERn+<br>op r<br>•Register indirect with pre-decrement @-ERn<br>op r | 31 0<br>General register contents<br>1, 2, or 4<br>31 0<br>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<br>23 0 |

## Table 2.12 Effective Address Calculation (2)




[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.

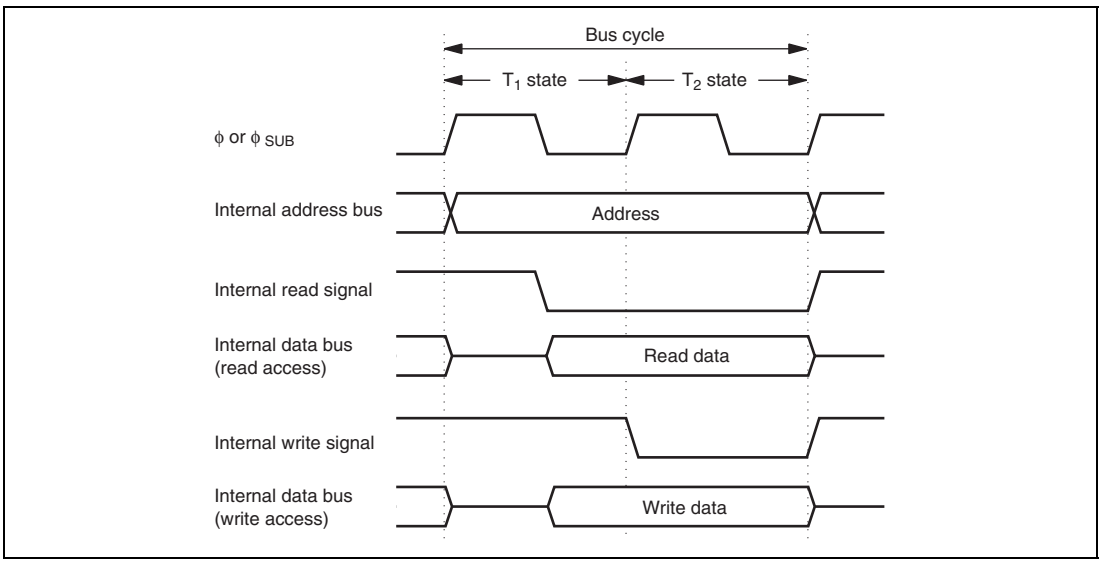



**Figure 2.9 On-Chip Memory Access Cycle**

### 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 16.1, 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, access is completed in two cycles. 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.

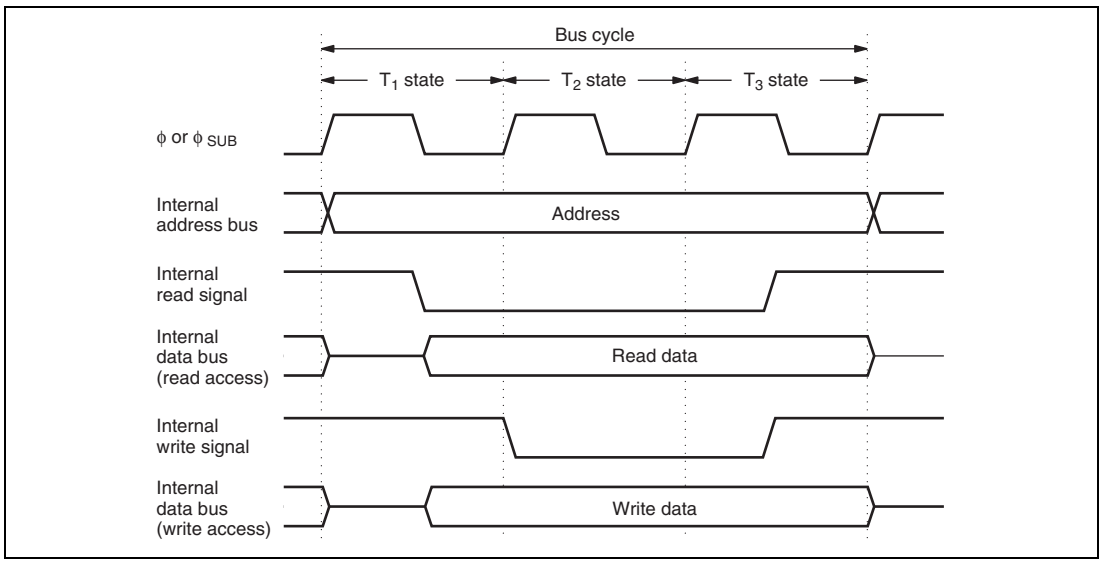



**Figure 2.10 On-Chip Peripheral Module Access Cycle (3-State Access)**

## 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 mode. In the program halt state there are a sleep mode, and standby 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 6, Power-Down Modes. For details on exception processing, refer to section 3, Exception Handling.

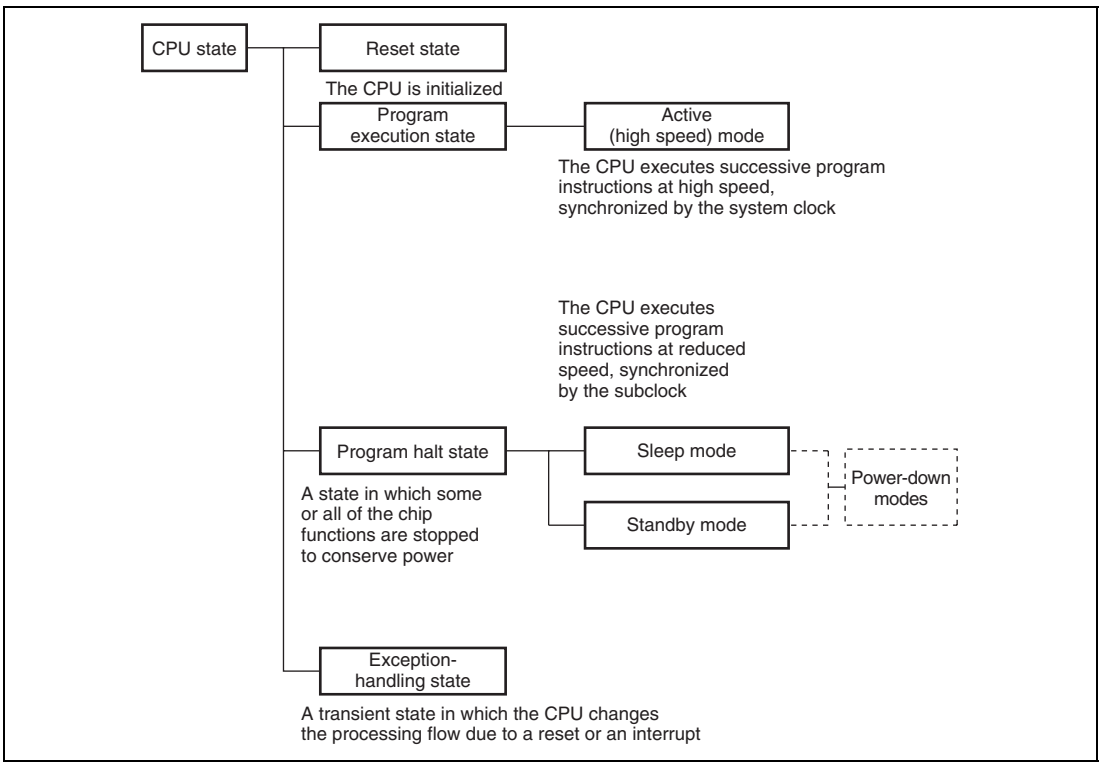



**Figure 2.11 CPU Operation States**

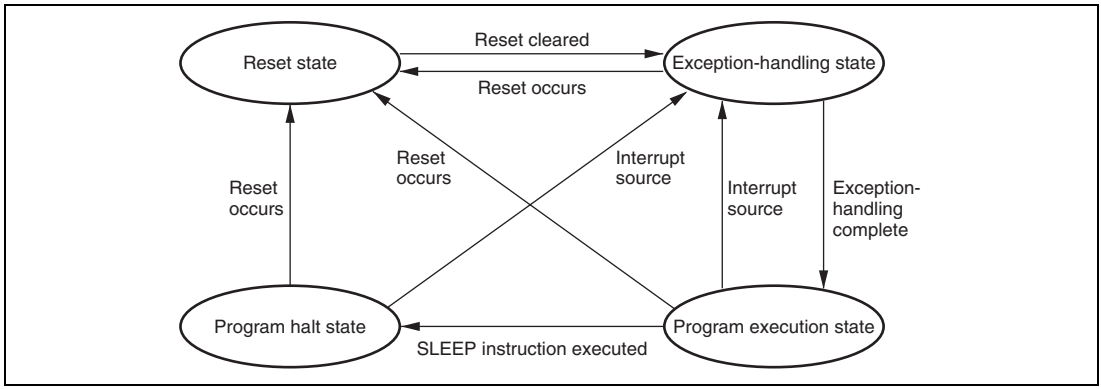



**Figure 2.12 State Transitions**

## 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, because this may rewrite data of a bit other than the bit to be manipulated.

**Bit manipulation for two registers assigned to the same address**

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

**(Applicable for timer B and timer C, not for the series of this LSI.)**

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.

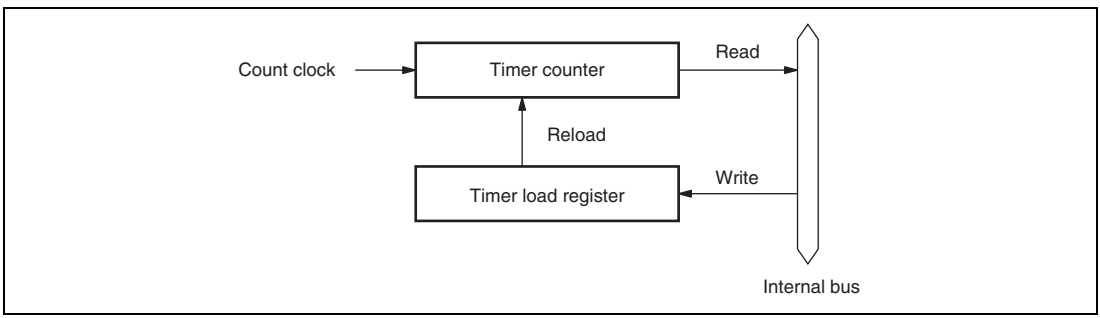



**Figure 2.13 Example of Timer Configuration with Two Registers Allocated to Same Address**

**Example 2: 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 instruction

```
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.
   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   #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 |

**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 PCR5 data in a work area in memory and manipulate data of the bit in the work area, then write this data to PCR5.

- Prior to executing BCLR instruction

```
MOV.B   #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 |

# Section 3 Exception Handling

Exception handling may be caused by a reset, a trap instruction (TRAPA), or interrupts.

- Reset

A reset has the highest exception priority. Exception handling starts as soon as the reset is cleared by the $\overline{\text{RES}}$ pin. The chip is also reset when the watchdog timer overflows, and exception handling starts. Exception handling is the same as exception handling by the $\overline{\text{RES}}$ pin.

- Trap Instruction

Exception handling starts when a trap instruction (TRAPA) is executed. The TRAPA instruction generates a vector address corresponding to a vector number from 0 to 3, as specified in the instruction code. Exception handling can be executed at all times in the program execution state.

- Interrupts

External interrupts other than NMI and internal interrupts other than address break are masked by the I bit in CCR, and kept masked while the I bit is set to 1. Exception handling starts when the current instruction or exception handling ends, if an interrupt request has been issued.

## 3.1 Exception Sources and Vector Address

Table 3.1 shows the vector addresses and priority of each exception handling. When more than one interrupt is requested, handling is performed from the interrupt with the highest priority.

**Table 3.1 Exception Sources and Vector Address**

| Relative Module | Exception Sources | Vector Number | Vector Address | Priority |
|---|---|---|---|---|
| $\overline{\text{RES}}$ pin | Reset | 0 | H'0000 to H'0001 | High |
| Watchdog timer | | | | |
| — | Reserved for system use | 1 to 6 | H'0002 to H'000D | |
| External interrupt pin | NMI | 7 | H'000E to H'000F | |
| CPU | Trap instruction (#0) | 8 | H'0010 to H'0011 | |
| | (#1) | 9 | H'0012 to H'0013 | |
| | (#2) | 10 | H'0014 to H'0015 | |
| | (#3) | 11 | H'0016 to H'0017 | |
| Address break | Break conditions satisfied | 12 | H'0018 to H'0019 | |
| CPU | Direct transition by executing the SLEEP instruction | 13 | H'001A to H'001B | |
| External interrupt pin | IRQ0 | 14 | H'001C to H'001D | |
| | IRQ3 | 17 | H'0022 to H'0023 | |
| | WKP | 18 | H'0024 to H'0025 | |
| — | Reserved for system use | 20 | H'0028 to H'0029 | |
| Timer W | Input capture A/compare match A<br>Input capture B/compare match B<br>Input capture C/compare match C<br>Input capture D/compare match D<br>Timer W overflow | 21 | H'002A to H'002B | |
| Timer V | Timer V compare match A<br>Timer V compare match B<br>Timer V overflow | 22 | H'002C to H'002D | |
| SCI3 | SCI3 receive data full<br>SCI3 transmit data empty<br>SCI3 transmit end<br>SCI3 receive error | 23 | H'002E to H'002F | |
| A/D converter | A/D conversion end | 25 | H'0032 to H'0033 | Low |

## 3.2 Register Descriptions

Interrupts are controlled by the following registers.

- Interrupt edge select register 1 (IEGR1)
- Interrupt edge select register 2 (IEGR2)
- Interrupt enable register 1 (IENR1)
- Interrupt flag register 1 (IRR1)
- Wakeup interrupt flag register (IWPR)

### 3.2.1 Interrupt Edge Select Register 1 (IEGR1)

IEGR1 selects the direction of an edge that generates interrupt requests of pins and $\overline{\text{IRQ3}}$ and $\overline{\text{IRQ0}}$.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved<br>This bit is always read as 0. |
| 6 | — | 1 | — | Reserved |
| 5 | — | 1 | — | These bits are always read as 1. |
| 4 | — | 1 | — | |
| 3 | IEG3 | 0 | R/W | IRQ3 Edge Select<br>0: Falling edge of $\overline{\text{IRQ3}}$ pin input is detected<br>1: Rising edge of $\overline{\text{IRQ3}}$ pin input is detected |
| 2 | — | 0 | — | Reserved |
| 1 | — | 0 | — | These bits are always read as 0. |
| 0 | IEG0 | 0 | R/W | IRQ0 Edge Select<br>0: Falling edge of $\overline{\text{IRQ0}}$ pin input is detected<br>1: Rising edge of $\overline{\text{IRQ0}}$ pin input is detected |

### 3.2.2 Interrupt Edge Select Register 2 (IEGR2)

IEGR2 selects the direction of an edge that generates interrupt requests of the pins $\overline{\text{ADTRG}}$ and $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved |
| 6 | — | 1 | — | These bits are always read as 1. |
| 5 | WPEG5 | 0 | R/W | WKP5 Edge Select<br>0: Falling edge of $\overline{\text{WKP5}}$ ($\overline{\text{ADTRG}}$) pin input is detected<br>1: Rising edge of $\overline{\text{WKP5}}$ ($\overline{\text{ADTRG}}$) pin input is detected |
| 4 | WPEG4 | 0 | R/W | WKP4 Edge Select<br>0: Falling edge of $\overline{\text{WKP4}}$ pin input is detected<br>1: Rising edge of $\overline{\text{WKP4}}$ pin input is detected |
| 3 | WPEG3 | 0 | R/W | WKP3 Edge Select<br>0: Falling edge of $\overline{\text{WKP3}}$ pin input is detected<br>1: Rising edge of $\overline{\text{WKP3}}$ pin input is detected |
| 2 | WPEG2 | 0 | R/W | WKP2 Edge Select<br>0: Falling edge of $\overline{\text{WKP2}}$ pin input is detected<br>1: Rising edge of $\overline{\text{WKP2}}$ pin input is detected |
| 1 | WPEG1 | 0 | R/W | WKP1Edge Select<br>0: Falling edge of $\overline{\text{WKP1}}$ pin input is detected<br>1: Rising edge of $\overline{\text{WKP1}}$ pin input is detected |
| 0 | WPEG0 | 0 | R/W | WKP0 Edge Select<br>0: Falling edge of $\overline{\text{WKP0}}$ pin input is detected<br>1: Rising edge of $\overline{\text{WKP0}}$ pin input is detected |

### 3.2.3 Interrupt Enable Register 1 (IENR1)

IENR1 enables direct transition interrupts, and external pin interrupts.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | IENDT | 0 | R/W | Direct Transfer Interrupt Enable<br>When this bit is set to 1, direct transition interrupt requests are enabled. |
| 6 | — | 0 | — | Reserved<br>This bit is always read as 0. |
| 5 | IENWP | 0 | R/W | Wakeup Interrupt Enable<br>This bit is an enable bit, which is common to the pins $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$. When the bit is set to 1, interrupt requests are enabled. |
| 4 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 3 | IEN3 | 0 | R/W | IRQ3 Interrupt Enable<br>When this bit is set to 1, interrupt requests of the $\overline{\text{IRQ3}}$ pin are enabled. |
| 2 | — | 0 | — | Reserved |
| 1 | — | 0 | — | These bits are always read as 0. |
| 0 | IEN0 | 0 | R/W | IRQ0 Interrupt Enable<br>When this bit is set to 1, interrupt requests of the $\overline{\text{IRQ0}}$ pin are enabled. |

When disabling interrupts by clearing bits in an interrupt enable register, or when clearing bits in an interrupt flag register, always do so while interrupts are masked (I = 1). 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 handling for the interrupt will be executed after the clear instruction has been executed.

### 3.2.4 Interrupt Flag Register 1 (IRR1)

IRR1 is a status flag register for direct transition interrupts, and $\overline{\text{IRQ3}}$ and $\overline{\text{IRQ0}}$ interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | IRRDT | 0 | R/W | Direct Transfer Interrupt Request Flag<br>[Setting condition]<br>When a direct transfer is made by executing a SLEEP instruction while DTON in SYSCR2 is set to 1.<br>[Clearing condition]<br>When IRRDT is cleared by writing 0 |
| 6 | — | 0 | — | Reserved<br>This bit is always read as 0. |
| 5 | — | 1 | — | Reserved |
| 4 | — | 1 | — | These bits are always read as 1. |
| 3 | IRRI3 | 0 | R/W | IRQ3 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{IRQ3}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IRRI3 is cleared by writing 0 |
| 2 | — | 0 | — | Reserved |
| 1 | — | 0 | — | These bits are always read as 0. |
| 0 | IRRI0 | 0 | R/W | IRQ0 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{IRQ0}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IRRI0 is cleared by writing 0 |

### 3.2.5 Wakeup Interrupt Flag Register (IWPR)

IWPR is a status flag register for $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$ interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved |
| 6 | — | 1 | — | These bits are always read as 1. |
| 5 | IWPF5 | 0 | R/W | WKP5 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP5}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF5 is cleared by writing 0. |
| 4 | IWPF4 | 0 | R/W | WKP4 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP4}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF4 is cleared by writing 0. |
| 3 | IWPF3 | 0 | R/W | WKP3 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP3}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF3 is cleared by writing 0. |
| 2 | IWPF2 | 0 | R/W | WKP2 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP2}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF2 is cleared by writing 0. |
| 1 | IWPF1 | 0 | R/W | WKP1 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP1}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF1 is cleared by writing 0. |
| 0 | IWPF0 | 0 | R/W | WKP0 Interrupt Request Flag<br>[Setting condition]<br>When $\overline{\text{WKP0}}$ pin is designated for interrupt input and the designated signal edge is detected.<br>[Clearing condition]<br>When IWPF0 is cleared by writing 0. |

## 3.3 Reset Exception Handling

When the $\overline{\text{RES}}$ pin goes low, all processing halts and this LSI enters the reset. The internal state of the CPU and the registers of the on-chip peripheral modules are initialized by the reset. To ensure that this LSI is reset at power-up, hold the $\overline{\text{RES}}$ pin low until the clock pulse generator output stabilizes. To reset the chip during operation, hold the $\overline{\text{RES}}$ pin low for at least 10 system clock cycles. When the $\overline{\text{RES}}$ pin goes high after being held low for the necessary time, this LSI starts reset exception handling. The reset exception handling sequence is shown in figure 3.1. The reset exception handling sequence is as follows:

1. Set the I bit in the condition code register (CCR) to 1.
2. The CPU generates a reset exception handling vector address (from H'0000 to H'0001), the data in that address is sent to the program counter (PC) as the start address, and program execution starts from that address.

## 3.4 Interrupt Exception Handling

### 3.4.1 External Interrupts

There are external interrupts, NMI, IRQ3, IRQ0, and WKP.

**NMI**

NMI interrupt is requested by input falling edge to pin $\overline{\text{NMI}}$.

NMI is the highest interrupt, and can always be accepted without depending on the I bit value in CCR.

**IRQ3 to IRQ0 Interrupts**

IRQ3 to IRQ0 interrupts are requested by input signals to pins $\overline{\text{IRQ3}}$ to $\overline{\text{IRQ0}}$. These four interrupts are given different vector addresses, and are detected individually by either rising edge sensing or falling edge sensing, depending on the settings of bits IEG3 to IEG0 in IEGR1.

When pins $\overline{\text{IRQ3}}$ to $\overline{\text{IRQ0}}$ are designated for interrupt input in PMR1 and the designated signal edge is input, the corresponding bit in IRR1 is set to 1, requesting the CPU of an interrupt. When IRQ3 to IRQ0 interrupt is accepted, the I bit is set to 1 in CCR. These interrupts can be masked by setting bits IEN3 to IEN0 in IENR1.

### WKP5 to WKP0 Interrupts

WKP5 to WKP0 interrupts are requested by input signals to pins $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$. These six interrupts have the same vector addresses, and are detected individually by either rising edge sensing or falling edge sensing, depending on the settings of bits WPEG5 to WPEG0 in IEGR2.

When pins $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$ are designated for interrupt input in PMR5 and the designated signal edge is input, the corresponding bit in IWPR is set to 1, requesting the CPU of an interrupt. These interrupts can be masked by setting bit IENWP in IENR1.

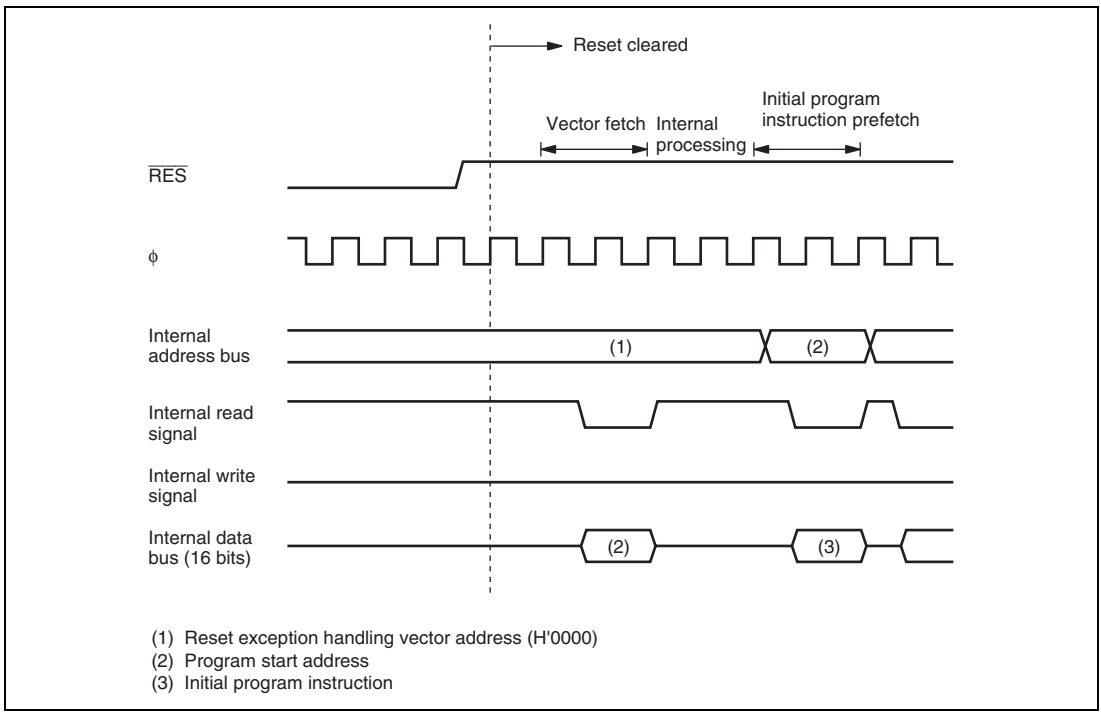



**Figure 3.1 Reset Sequence**

### 3.4.2 Internal Interrupts

Each on-chip peripheral module has a flag to show the interrupt request status and the enable bit to enable or disable the interrupt. For direct transfer interrupt requests generated by execution of a SLEEP instruction, this function is included in IRR1 and IENR1.

When an on-chip peripheral module requests an interrupt, the corresponding interrupt request status flag is set to 1, requesting the CPU of an interrupt. When this interrupt is accepted, the I bit is set to 1 in CCR. These interrupts can be masked by writing 0 to clear the corresponding enable bit.

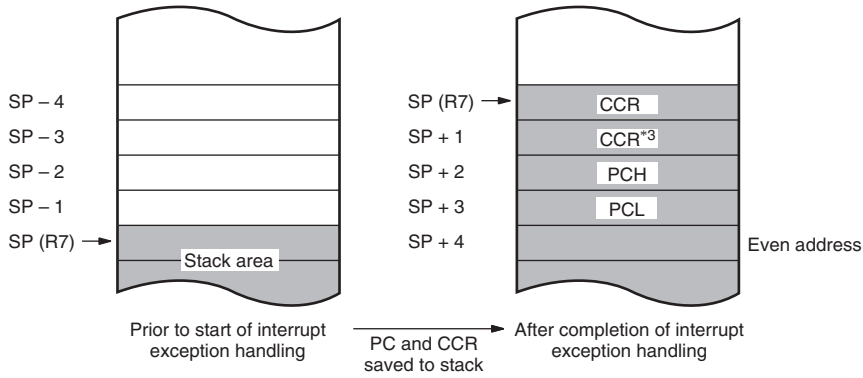
### 3.4.3 Interrupt Handling Sequence

Interrupts are controlled by an interrupt controller.

Interrupt operation is described as follows.

1. If an interrupt occurs while the NMI or interrupt enable bit is set to 1, an interrupt request signal is sent to the interrupt controller.
2. When multiple interrupt requests are generated, the interrupt controller requests to the CPU for the interrupt handling with the highest priority at that time according to table 3.1. Other interrupt requests are held pending.
3. The CPU accepts the NMI or address break without depending on the I bit value. Other interrupt requests are accepted, if the I bit is cleared to 0 in CCR; if the I bit is set to 1, the interrupt request is held pending.
4. If the CPU accepts the interrupt after processing of the current instruction is completed, interrupt exception handling will begin. First, both PC and CCR are pushed onto the stack. The state of the stack at this time is shown in figure 3.2. The PC value pushed onto the stack is the address of the first instruction to be executed upon return from interrupt handling.
5. Then, the I bit of CCR is set to 1, masking further interrupts excluding the NMI and address break. Upon return from interrupt handling, the values of I bit and other bits in CCR will be restored and returned to the values prior to the start of interrupt exception handling.
6. Next, the CPU generates the vector address corresponding to the accepted interrupt, and transfers the address to PC as a start address of the interrupt handling-routine. Then a program starts executing from the address indicated in PC.

Figure 3.3 shows a typical interrupt sequence where the program area is in the on-chip ROM and the stack area is in the on-chip RAM.




[Legend]
PCH: Upper 8 bits of program counter (PC)
PCL: 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 length, starting from an even-numbered address.
3. Ignored when returning from the interrupt handling routine.

**Figure 3.2 Stack Status after Exception Handling**

### 3.4.4 Interrupt Response Time

Table 3.2 shows the number of wait states after an interrupt request flag is set until the first instruction of the interrupt handling-routine is executed.

**Table 3.2 Interrupt Wait States**

| Item | States | Total |
|---|---|---|
| Waiting time for completion of executing instruction* | 1 to 23 | 15 to 37 |
| Saving of PC and CCR to stack | 4 | |
| Vector fetch | 2 | |
| Instruction fetch | 4 | |
| Internal processing | 4 | |

Note: * Not including EEPMOV instruction.

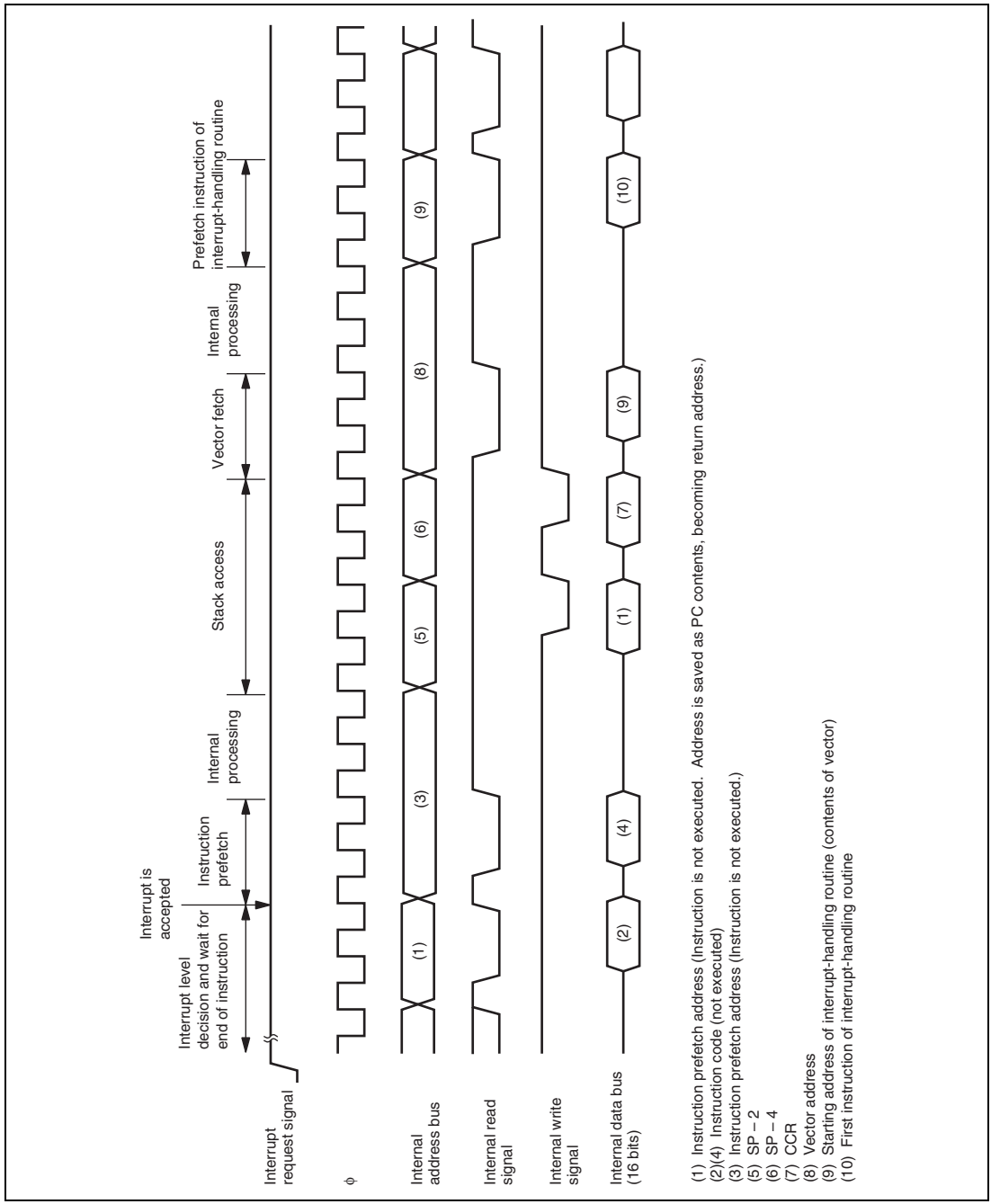



**Figure 3.3 Interrupt Sequence**

## 3.5 Usage Notes

### 3.5.1 Interrupts after Reset

If an interrupt is accepted after a reset and before the stack pointer (SP) is initialized, the PC and CCR will not be saved correctly, leading to a program crash. To prevent this, all interrupt requests, including NMI, are disabled immediately after a reset. Since the first instruction of a program is always executed immediately after the reset state ends, make sure that this instruction initializes the stack pointer (example: MOV.W #xx: 16, SP).

### 3.5.2 Notes on Stack Area Use

When word data is accessed, 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.

### 3.5.3 Notes on Rewriting Port Mode Registers

When a port mode register is rewritten to switch the functions of external interrupt pins, $\overline{\text{IRQ3}}$ to $\overline{\text{IRQ0}}$, and $\overline{\text{WKP5}}$ to $\overline{\text{WKP0}}$, the interrupt request flag may be set to 1.

Figure 3.4 shows a port mode register setting and interrupt request flag clearing procedure.

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.

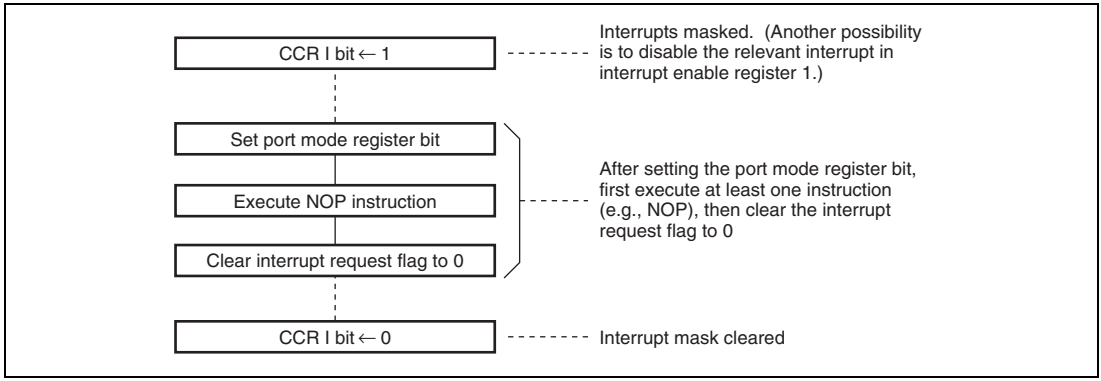



**Figure 3.4 Port Mode Register Setting and Interrupt Request Flag Clearing Procedure**

# Section 4 Address Break

The address break simplifies on-board program debugging. It requests an address break interrupt when the set break condition is satisfied. The interrupt request is not affected by the I bit of CCR. Break conditions that can be set include instruction execution at a specific address and a combination of access and data at a specific address. With the address break function, the execution start point of a program containing a bug is detected and execution is branched to the correcting program. Figure 4.1 shows a block diagram of the address break.

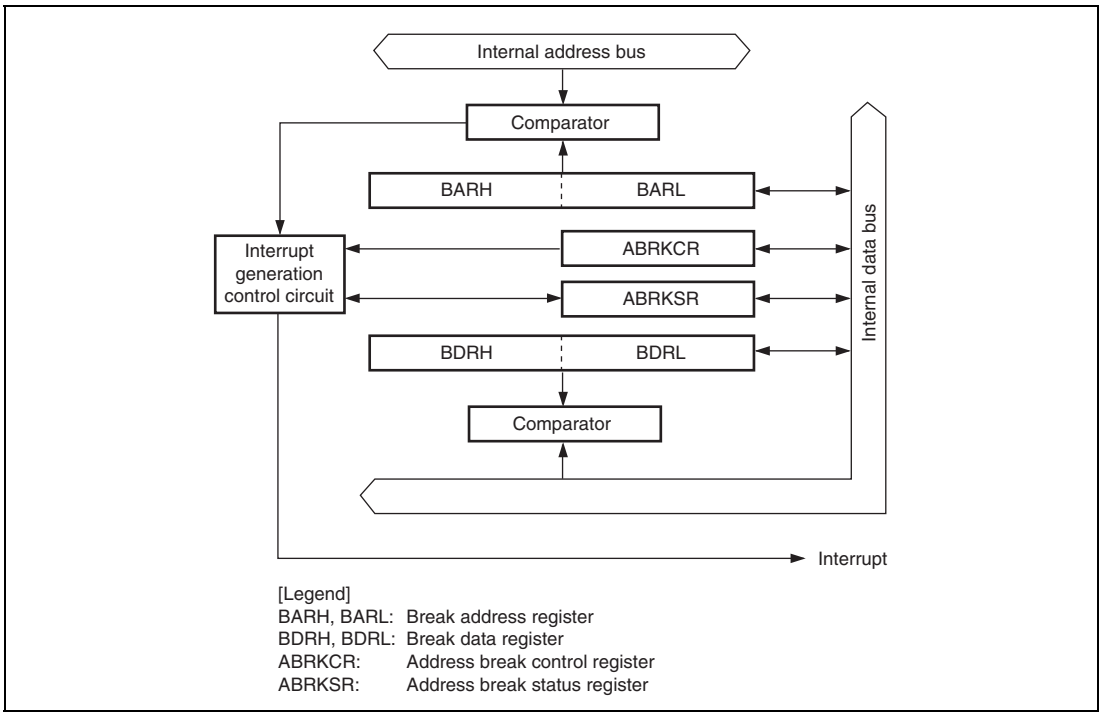



**Figure 4.1 Block Diagram of Address Break**

## 4.1 Register Descriptions

Address break has the following registers.

- Address break control register (ABRKCR)
- Address break status register (ABRKSR)
- Break address register (BARH, BARL)
- Break data register (BDRH, BDRL)

### 4.1.1 Address Break Control Register (ABRKCR)

ABRKCR sets address break conditions.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | RTINTE | 1 | R/W | RTE Interrupt Enable<br>When this bit is 0, the interrupt immediately after executing RTE is masked and then one instruction must be executed. When this bit is 1, the interrupt is not masked. |
| 6 | CSEL1 | 0 | R/W | Condition Select 1 and 0 |
| 5 | CSEL0 | 0 | R/W | These bits set address break conditions.<br>00: Instruction execution cycle<br>01: CPU data read cycle<br>10: CPU data write cycle<br>11: CPU data read/write cycle |
| 4 | ACMP2 | 0 | R/W | Address Compare Condition Select 2 to 0 |
| 3 | ACMP1 | 0 | R/W | These bits comparison condition between the address set in BAR and the internal address bus.<br>000: Compares 16-bit addresses<br>001: Compares upper 12-bit addresses<br>010: Compares upper 8-bit addresses<br>011: Compares upper 4-bit addresses<br>1XX: Reserved (setting prohibited) |
| 2 | ACMP0 | 0 | R/W | |
| 1 | DCMP1 | 0 | R/W | Data Compare Condition Select 1 and 0 |
| 0 | DCMP0 | 0 | R/W | These bits set the comparison condition between the data set in BDR and the internal data bus.<br>00: No data comparison<br>01: Compares lower 8-bit data between BDRL and data bus<br>10: Compares upper 8-bit data between BDRH and data bus<br>11: Compares 16-bit data between BDR and data bus |

Legend: X: Don't care.

When an address break is set in the data read cycle or data write cycle, the data bus used will depend on the combination of the byte/word access and address. Table 4.1 shows the access and data bus used. When an I/O register space with an 8-bit data bus width is accessed in word size, a byte access is generated twice. For details on data widths of each register, see section 16.1, Register Addresses (Address Order).

**Table 4.1 Access and Data Bus Used**

| | Word Access | | Byte Access | |
|---|---|---|---|---|
| | **Even Address** | **Odd Address** | **Even Address** | **Odd Address** |
| ROM space | Upper 8 bits | Lower 8 bits | Upper 8 bits | Upper 8 bits |
| RAM space | Upper 8 bits | Lower 8 bits | Upper 8 bits | Upper 8 bits |
| I/O register with 8-bit data bus width | Upper 8 bits | Upper 8 bits | Upper 8 bits | Upper 8 bits |
| I/O register with 16-bit data bus width | Upper 8 bits | Lower 8 bits | — | — |

### 4.1.2 Address Break Status Register (ABRKSR)

ABRKSR consists of the address break interrupt flag and the address break interrupt enable bit.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | ABIF | 0 | R/W | Address Break Interrupt Flag<br>[Setting condition]<br>When the condition set in ABRKCR is satisfied<br>[Clearing condition]<br>When 0 is written after ABIF=1 is read |
| 6 | ABIE | 0 | R/W | Address Break Interrupt Enable<br>When this bit is 1, an address break interrupt request is enabled. |
| 5 to 0 | — | All 1 | — | Reserved<br>These bits are always read as 1. |

### 4.1.3 Break Address Registers (BARH, BARL)

BARH and BARL are 16-bit read/write registers that set the address for generating an address break interrupt. When setting the address break condition to the instruction execution cycle, set the first byte address of the instruction. The initial value of this register is H'FFFF.

### 4.1.4 Break Data Registers (BDRH, BDRL)

BDRH and BDRL are 16-bit read/write registers that set the data for generating an address break interrupt. BDRH is compared with the upper 8-bit data bus. BDRL is compared with the lower 8-bit data bus. When memory or registers are accessed by byte, the upper 8-bit data bus is used for even and odd addresses in the data transmission. Therefore, comparison data must be set in BDRH for byte access. For word access, the data bus used depends on the address. See section 4.1.1, Address Break Control Register (ABRKCR), for details. The initial value of this register is undefined.

## 4.2 Operation

When the ABIF and ABIE bits in ABRKSR are set to 1, the address break function generates an interrupt request to the CPU. The ABIF bit in ABRKSR is set to 1 by the combination of the address set in BAR, the data set in BDR, and the conditions set in ABRKCR. When the interrupt request is accepted, interrupt exception handling starts after the instruction being executed ends. The address break interrupt is not masked because of the I bit in CCR of the CPU.

Figures 4.2 show the operation examples of the address break interrupt setting.

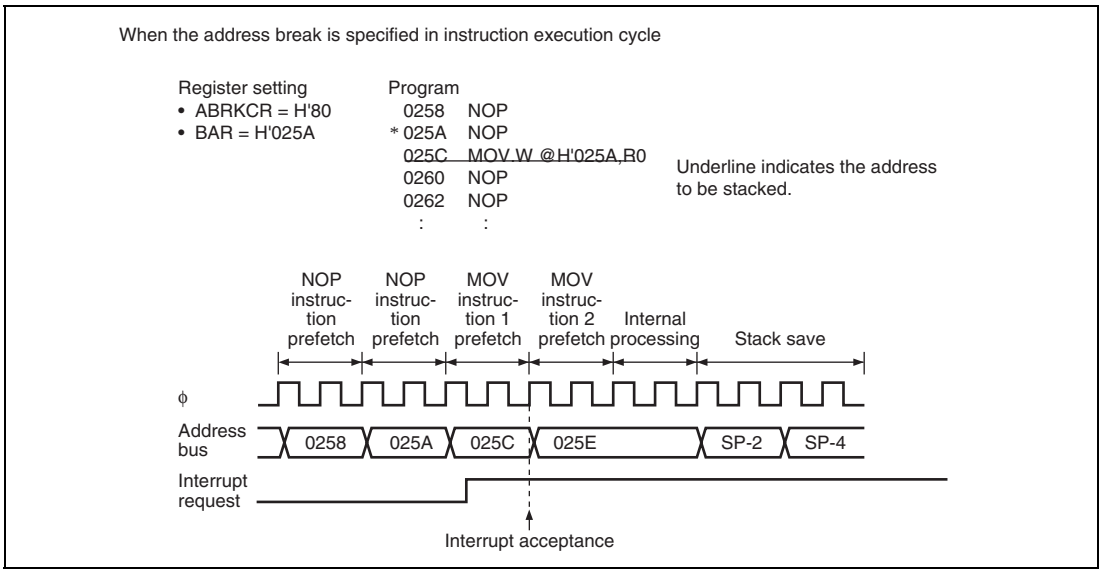



**Figure 4.2 Address Break Interrupt Operation Example (1)**

When the address break is specified in the data read cycle

Register setting
- ABRKCR = H'A0
- BAR = H'025A

Program

```
  0258  NOP
  025A  NOP
* 025C  MOV.W @H'025A,R0
  0260  NOP
  0262  NOP
    :     :
```

Underline indicates the address to be stacked.

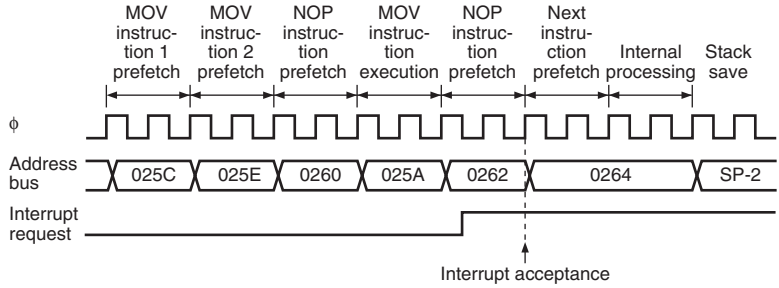



**Figure 4.2 Address Break Interrupt Operation Example (2)**

## 4.3 Usage Notes

When an address break is set to an instruction after a conditional branch instruction, and the instruction set when the condition of the branch instruction is not satisfied is executed (see figure 4.3), note that an address break interrupt request is not generated. Therefore an address break must not be set to the instruction after a conditional branch instruction.

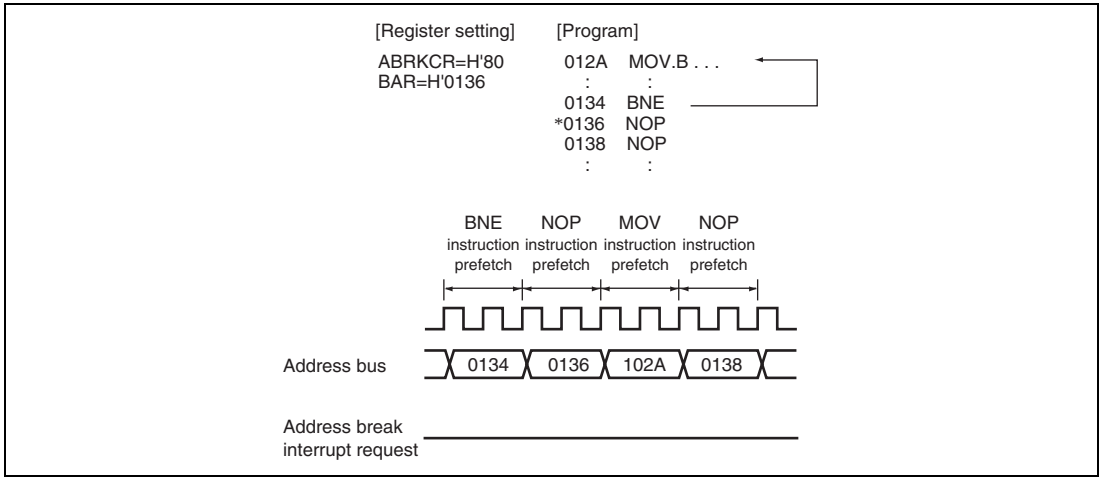



**Figure 4.3 Operation when Condition is not Satisfied in Branch Instruction**

When another interrupt request is accepted before an instruction to which an address break is set is executed, exception handling of an address break interrupt is not executed. However, the ABIF bit is set to 1 (see figure 4.4). Therefore the ABIF bit must be read during exception handling of an address break interrupt.

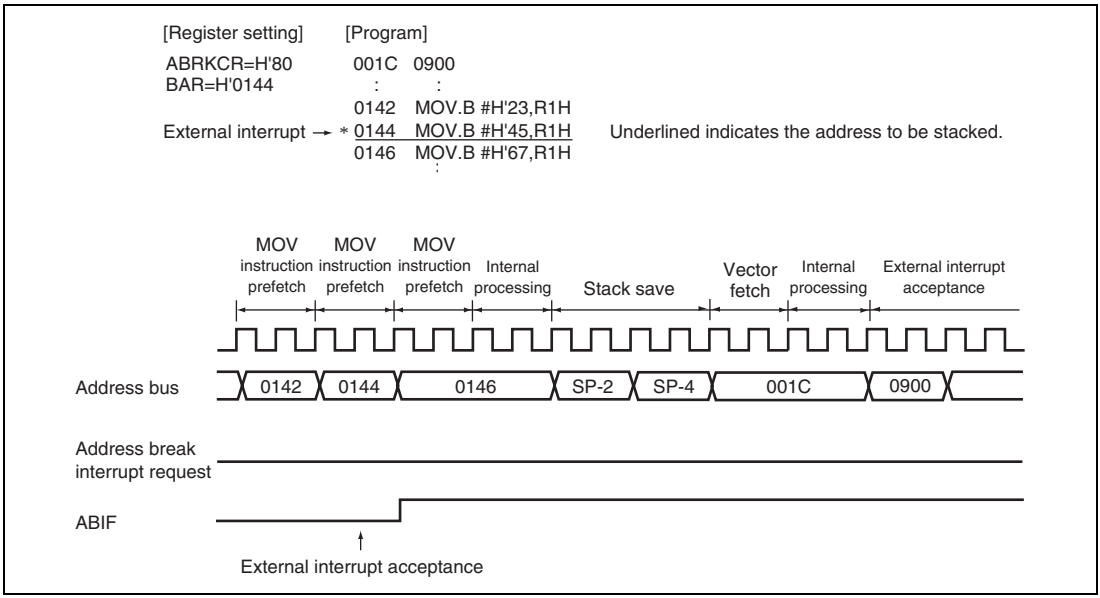



**Figure 4.4 Operation when Another Interrupt is Accepted at Address Break Setting Instruction**

When an address break is set to an instruction as a branch destination of a conditional branch instruction, the instruction set when the condition of the branch instruction is not satisfied is not executed, and an address break is generated. Therefore an address break must not be set to the instruction as a branch destination of a conditional branch instruction.

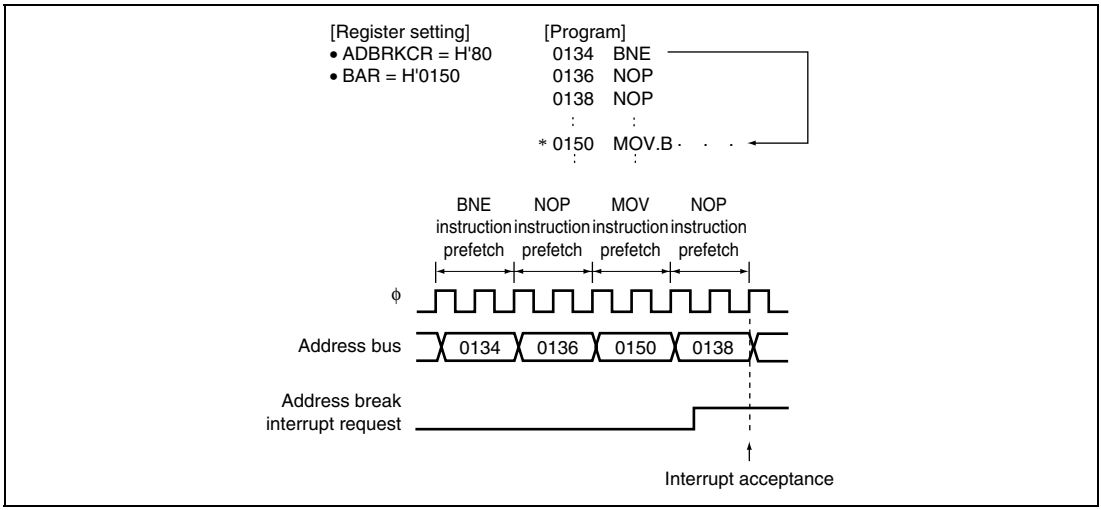



**Figure 4.5 Operation when the Instruction Set is not Executed and does not Branch due to Conditions not Being Satisfied**

# Section 5 Clock Pulse Generators

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

Figure 5.1 shows a block diagram of the clock pulse generators.

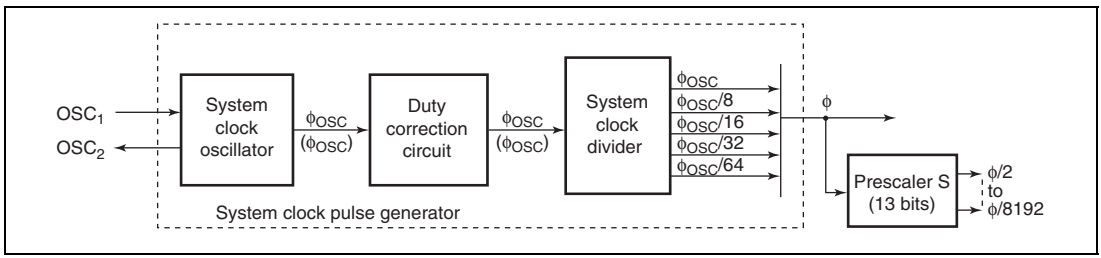



**Figure 5.1 Block Diagram of Clock Pulse Generators**

The basic clock signals that drive the CPU and on-chip peripheral modules are ø.

The system clock is divided into ø/8192 to ø/2 by prescaler S and they are supplied to respective peripheral modules.

## 5.1 System Clock Generator

Clock pulses can be supplied to the system clock divider either by connecting a crystal or ceramic resonator, or by providing external clock input. Figure 5.2 shows a block diagram of the system clock generator.

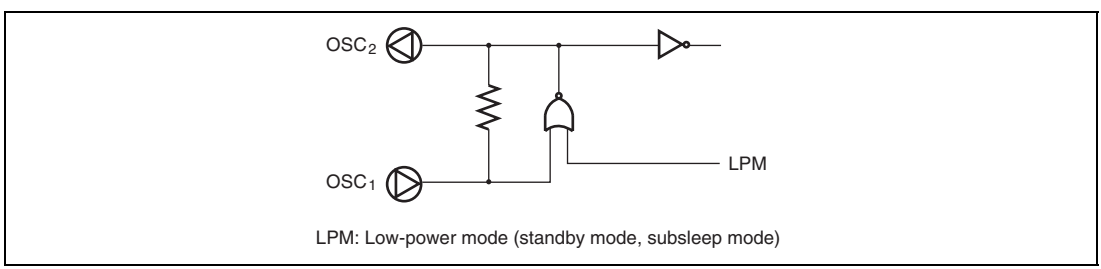



**Figure 5.2 Block Diagram of System Clock Generator**

### 5.1.1 Connecting Crystal Resonator

Figure 5.3 shows a typical method of connecting a crystal resonator. An AT-cut parallel-resonance crystal resonator should be used. Figure 5.4 shows the equivalent circuit of a crystal resonator. A resonator having the characteristics given in table 5.1 should be used.

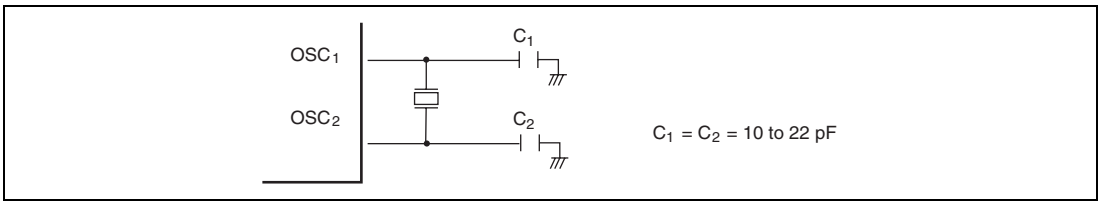



**Figure 5.3 Typical Connection to Crystal Resonator**

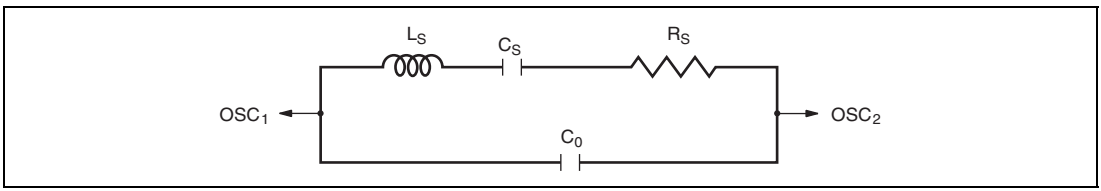



**Figure 5.4 Equivalent Circuit of Crystal Resonator**

**Table 5.1 Crystal Resonator Parameters**

| Frequency (MHz) | 2 | 4 | 8 | 10 | 16 |
|---|---|---|---|---|---|
| $R_s$ (max) | 500 Ω | 120 Ω | 80 Ω | 60 Ω | 50 Ω |
| $C_0$ (max) | 7 pF | 7 pF | 7 pF | 7 pF | 7 pF |

### 5.1.2 Connecting Ceramic Resonator

Figure 5.5 shows a typical method of connecting a ceramic resonator.

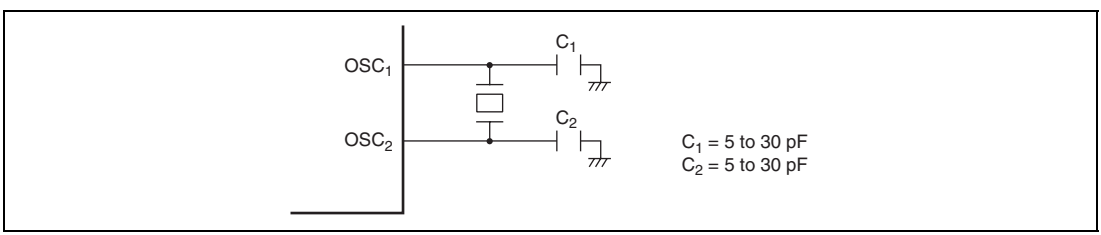



**Figure 5.5 Typical Connection to Ceramic Resonator**

### 5.1.3 External Clock Input Method

Connect an external clock signal to pin $OSC_1$, and leave pin $OSC_2$ open. Figure 5.6 shows a typical connection. The duty cycle of the external clock signal must be 45 to 55%.

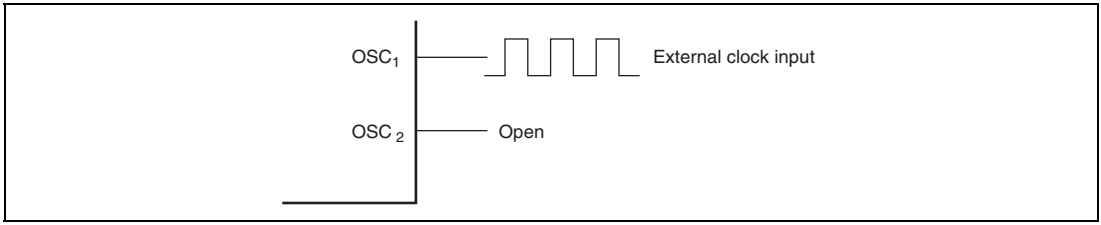



**Figure 5.6 Example of External Clock Input**

## 5.2 Prescalers

### 5.2.1 Prescaler S

Prescaler S is a 13-bit counter using the system clock (ø) 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 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 the on-chip peripheral modules. The divider ratio can be set separately for each on-chip peripheral function. In active mode and sleep mode, the clock input to prescaler S is determined by the division factor designated by MA2 to MA0 in SYSCR2.

## 5.3 Usage Notes

### 5.3.1 Note on Resonators

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

### 5.3.2 Notes on Board Design

When using a crystal resonator (ceramic resonator), place the resonator and its load capacitors as close as possible to the $OSC_1$ and $OSC_2$ pins. Other signal lines should be routed away from the oscillator circuit to prevent induction from interfering with correct oscillation (see figure 5.7).

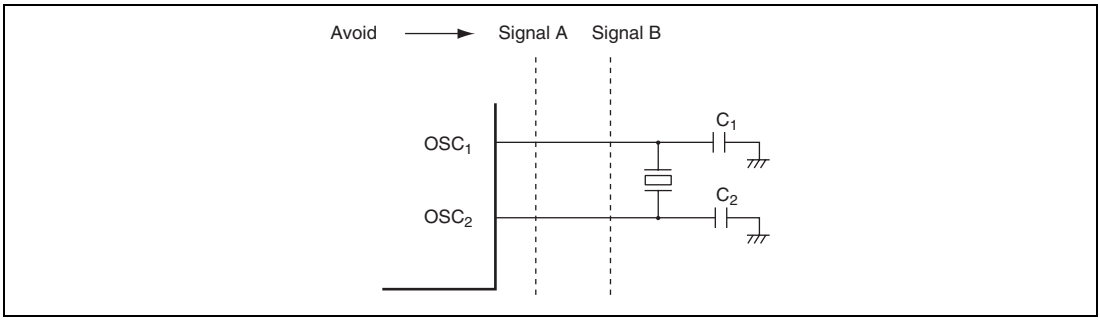



**Figure 5.7 Example of Incorrect Board Design**

# Section 6 Power-Down Modes

This LSI has five modes of operation after a reset. These include a normal active mode and three power-down modes, in which power consumption is significantly reduced. Module standby mode reduces power consumption by selectively halting on-chip module functions.

- Active mode

  The CPU and all on-chip peripheral modules are operable on the system clock. The system clock frequency can be selected from ϕosc, ϕosc/8, ϕosc/16, ϕosc/32, and ϕosc/64.
- Sleep mode

  The CPU halts. On-chip peripheral modules are operable on the system clock.
- Standby mode

  The CPU and all on-chip peripheral modules halt.
- Subsleep mode

  The CPU and all on-chip peripheral modules halt. I/O ports keep the same states as before the transition.
- Module standby mode

  Independent of the above modes, power consumption can be reduced by halting on-chip peripheral modules that are not used in module units.

## 6.1 Register Descriptions

The registers related to power-down modes are listed below.

- System control register 1 (SYSCR1)
- System control register 2 (SYSCR2)
- Module standby control register 1 (MSTCR1)

### 6.1.1 System Control Register 1 (SYSCR1)

SYSCR1 controls the power-down modes, as well as SYSCR2.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | SSBY | 0 | R/W | Software Standby<br>This bit selects the mode to transit after the execution of the SLEEP instruction.<br>0: a transition is made to sleep mode<br>1: a transition is made to standby mode.<br>For details, see table 6.2. |
| 6 | STS2 | 0 | R/W | Standby Timer Select 2 to 0<br>These bits designate the time the CPU and peripheral modules wait for stable clock operation after exiting from standby mode, to active mode or sleep mode due to an interrupt. The designation should be made according to the clock frequency so that the waiting time is at least 6.5 ms. The relationship between the specified value and the number of wait states is shown in table 6.1. When an external clock is to be used, the minimum value (STS2 = STS1 = STS0 = 1) is recommended. |
| 5 | STS1 | 0 | R/W | |
| 4 | STS0 | 0 | R/W | |
| 3 to 0 | — | 0 | — | Reserved<br>These bits are always read as 0. |

**Table 6.1 Operating Frequency and Waiting Time**

| STS2 | STS1 | STS0 | Waiting Time | 16 MHz | 10 MHz | 8 MHz | 4 MHz | 2 MHz | 1 MHz | 0.5 MHz |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 8,192 states | 0.5 | 0.8 | 1.0 | 2.0 | 4.1 | 8.1 | 16.4 |
| | | 1 | 16,384 states | 1.0 | 1.6 | 2.0 | 4.1 | 8.2 | 16.4 | 32.8 |
| | 1 | 0 | 32,768 states | 2.0 | 3.3 | 4.1 | 8.2 | 16.4 | 32.8 | 65.5 |
| | | 1 | 65,536 states | 4.1 | 6.6 | 8.2 | 16.4 | 32.8 | 65.5 | 131.1 |
| 1 | 0 | 0 | 131,072 states | 8.2 | 13.1 | 16.4 | 32.8 | 65.5 | 131.1 | 262.1 |
| | | 1 | 1,024 states | 0.06 | 0.10 | 0.13 | 0.26 | 0.51 | 1.02 | 2.05 |
| | 1 | 0 | 128 states | 0.00 | 0.01 | 0.02 | 0.03 | 0.06 | 0.13 | 0.26 |
| | | 1 | 16 states | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.03 |

Note: Time unit is ms

### 6.1.2 System Control Register 2 (SYSCR2)

SYSCR2 controls the power-down modes, as well as SYSCR1.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | SMSEL | 0 | R/W | Sleep Mode Selection<br>This bit selects the mode to transit after the execution of a SLEEP instruction, as well as bit SSBY of SYSCR1.<br>For details, see table 6.2. |
| 6 | — | 0 | — | Reserved<br>This bit is always read as 0. |
| 5 | DTON | 0 | R/W | Direct Transfer on Flag<br>This bit selects the mode to transit after the execution of a SLEEP instruction, as well as bit SSBY of SYSCR1.<br>For details, see table 6.2. |
| 4 | MA2 | 0 | R/W | Active Mode Clock Select 2 to 0 |
| 3 | MA1 | 0 | R/W | These bits select the operating clock frequency in active and sleep modes. The operating clock frequency changes to the set frequency after the SLEEP instruction is executed.<br>0XX: $\phi_{osc}$<br>100: $\phi_{osc}$/8<br>101: $\phi_{osc}$/16<br>110: $\phi_{osc}$/32<br>111: $\phi_{osc}$/64 |
| 2 | MA0 | 0 | R/W | |
| 1 | — | 0 | — | Reserved |
| 0 | — | 0 | — | These bits are always read as 0. |

Legend: X: Don't care.

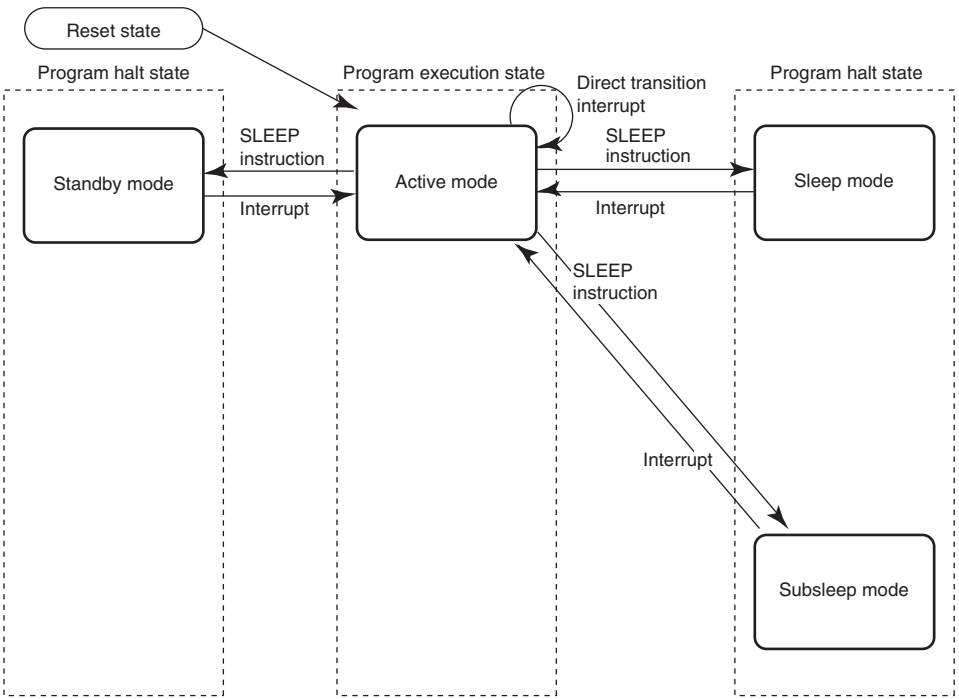
### 6.1.3 Module Standby Control Register 1 (MSTCR1)

MSTCR1 allows the on-chip peripheral modules to enter a standby state in module units.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved |
| 6 | — | 0 | — | These bits are always read as 0. |
| 5 | MSTS3 | 0 | R/W | SCI3 Module Standby<br>SCI3 enters standby mode when this bit is set to 1 |
| 4 | MSTAD | 0 | R/W | A/D Converter Module Standby<br>A/D converter enters standby mode when this bit is set to 1 |
| 3 | MSTWD | 0 | R/W | Watchdog Timer Module Standby<br>Watchdog timer enters standby mode when this bit is set to 1.When the internal oscillator is selected for the watchdog timer clock, the watchdog timer operates regardless of the setting of this bit |
| 2 | MSTTW | 0 | R/W | Timer W Module Standby<br>Timer W enters standby mode when this bit is set to 1 |
| 1 | MSTTV | 0 | R/W | Timer V Module Standby<br>Timer V enters standby mode when this bit is set to 1 |
| 0 | — | 0 | — | Reserved<br>This bit is always read as 0. |

## 6.2 Mode Transitions and States of LSI

Figure 6.1 shows the possible transitions among these operating modes. A transition is made from the program execution state to the program halt state of the program by executing a SLEEP instruction. Interrupts allow for returning from the program halt state to the program execution state of the program. A direct transition from active mode to active mode changes the operating frequency. $\overline{\text{RES}}$ input enables transitions from a mode to the reset state. Table 6.2 shows the transition conditions of each mode after the SLEEP instruction is executed and a mode to return by an interrupt. Table 6.3 shows the internal states of the LSI in each mode.




Notes: 1. To make a transition to another mode by an interrupt, make sure interrupt handling is after the interrupt is accepted.
2. Details on the mode transition conditions are given in table 6.2.

**Figure 6.1 Mode Transition Diagram**

**Table 6.2 Transition Mode after SLEEP Instruction Execution and Interrupt Handling**

| DTON | SSBY | SMSEL | Transition Mode after SLEEP Instruction Execution | Transition Mode due to Interrupt |
|---|---|---|---|---|
| 0 | 0 | 0 | Sleep mode | Active mode |
| | 0 | 1 | Subsleep mode | Active mode |
| | 1 | X | Standby mode | Active mode |
| 1 | X | 0* | Active mode (direct transition) | — |

Legend: X: Don't care.

* When a state transition is performed while SMSEL is 1, timer V, SCI3, and the A/D converter are reset, and all registers are set to their initial values. To use these functions after entering active mode, reset the registers.

**Table 6.3 Internal State in Each Operating Mode**

| Function | | Active Mode | Sleep Mode | Subsleep Mode | Standby Mode |
|---|---|---|---|---|---|
| System clock oscillator | | Functioning | Functioning | Halted | Halted |
| CPU operations | Instructions | Functioning | Halted | Halted | Halted |
| | Registers | Functioning | Retained | Retained | Retained |
| RAM | | Functioning | Retained | Retained | Retained |
| IO ports | | Functioning | Retained | Retained | Register contents are retained, but output is the high-impedance state. |
| External interrupts | IRQ3, IRQ0 | Functioning | Functioning | Functioning | Functioning |
| | WKP5 to WKP0 | Functioning | Functioning | Functioning | Functioning |
| Peripheral functions | Timer V | Functioning | Functioning | Reset | Reset |
| | Timer W | Functioning | Functioning | Retained | Retained (if internal clock $\phi$ is selected as a count clock, the counter is incremented by a subclock) |
| | Watchdog timer | Functioning | Functioning | Retained | Retained (functioning if the internal oscillator is selected as a count clock) |
| | SCI3 | Functioning | Functioning | Reset | Reset |
| | A/D converter | Functioning | Functioning | Reset | Reset |

### 6.2.1 Sleep Mode

In sleep mode, CPU operation is halted but the on-chip peripheral modules function at the clock frequency set by the MA2 to MA0 bits in SYSCR2. CPU register contents are retained. When an interrupt is requested, sleep mode is cleared and interrupt exception handling starts. Sleep mode is not cleared if the I bit of the condition code register (CCR) is set to 1 or the requested interrupt is disabled in the interrupt enable register. a transition is made to subactive mode when the bit is 1.

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

### 6.2.2 Standby Mode

In standby mode, the clock pulse generator stops, so the CPU and on-chip peripheral modules stop functioning. However, 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 retained as long as the voltage set by the RAM data retention voltage is provided. The I/O ports go to the high-impedance state.

Standby mode is cleared by an interrupt. When an interrupt is requested, the system clock pulse generator starts. After the time set in bits STS2–STS0 in SYSCR1 has elapsed, and interrupt exception handling starts. Standby mode is not cleared if the I bit of CCR is set to 1 or the requested interrupt is disabled in the interrupt enable register.

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

### 6.2.3 Subsleep Mode

In subsleep mode, the system clock oscillator is halted, and operation of the CPU and on-chip peripheral modules is halted. As long as a 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.

Subsleep mode is cleared by an interrupt. When an interrupt is requested, the system clock oscillator starts to oscillate. Subsleep mode is cleared and an interrupt exception handling starts when the time set in bits STS2 to STS0 in SYSCR1 elapses. Subsleep mode is not cleared if the I bit of CCR is 1 or the interrupt is disabled in the interrupt enable bit.

## 6.3 Operating Frequency in Active Mode

Operation in active mode is clocked at the frequency designated by the MA2 to MA0 bits in SYSCR2. The operating frequency changes to the set frequency after SLEEP instruction execution.

## 6.4 Direct Transition

The CPU can execute programs in active mode. The operating frequency can be changed by making a transition directly from active mode to active mode. A direct transition can be made by executing a SLEEP instruction while the DTON bit in SYSCR2 is set to 1. The direct transition also enables operating frequency modification in active mode. After the mode transition, direct transition interrupt exception handling starts.

If the direct transition interrupt is disabled in interrupt enable register 1, a transition is made instead to sleep mode. Note that if a direct transition is attempted while the I bit in CCR is set to 1, sleep mode will be entered, and the resulting mode cannot be cleared by means of an interrupt.

## 6.5 Module Standby Function

The module-standby function can be set to any peripheral module. In module standby mode, the clock supply to modules stops to enter the power-down mode. Module standby mode enables each on-chip peripheral module to enter the standby state by setting a bit that corresponds to each module in MSTCR1 and MSTCR2 to 1 and cancels the mode by clearing the bit to 0.

# Section 7 ROM

The features of the 20-kbyte (4 kbytes of them are the E7 or E8 control program area) flash memory built into HD64F3672 are summarized below.

- Programming/erase methods
  - The 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, 16 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.
- 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.

## 7.1 Block Configuration

Figure 7.1 shows the block configuration of 20-kbyte 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 and 16 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.

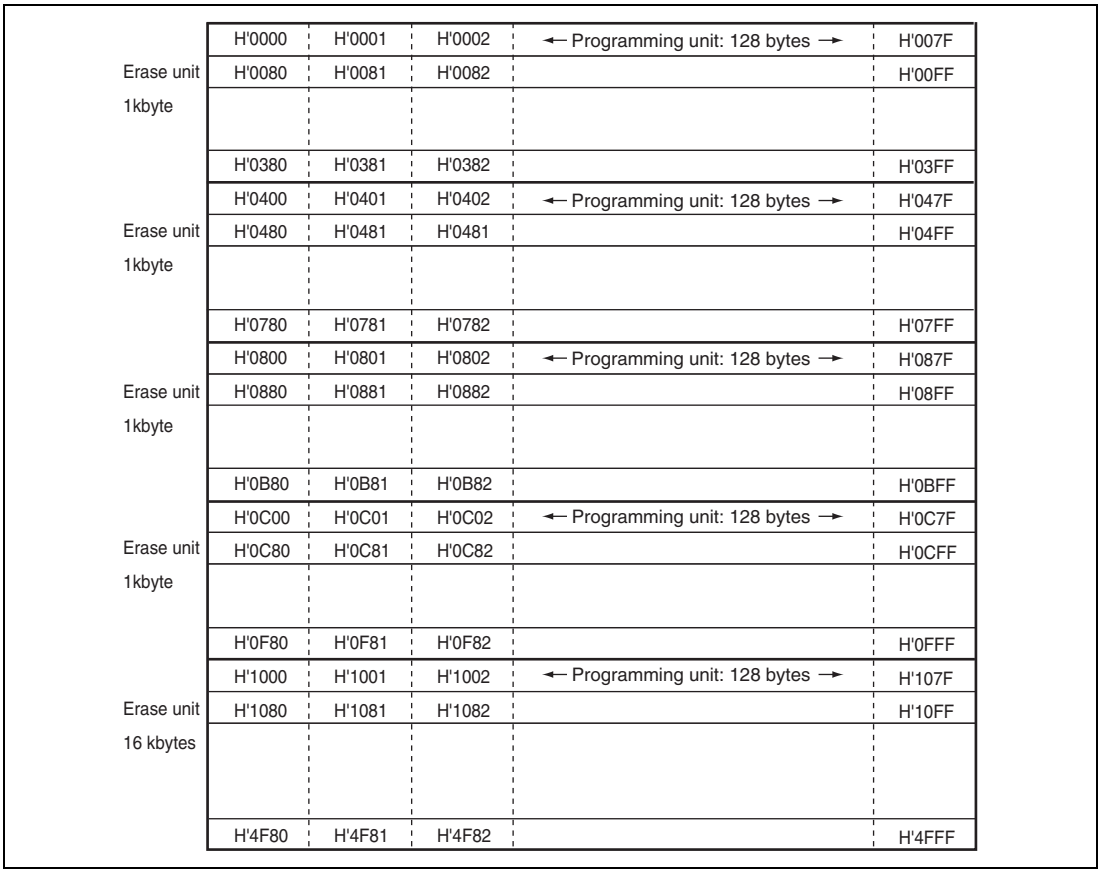



**Figure 7.1 Flash Memory Block Configuration**

## 7.2 Register Descriptions

The flash memory has the following registers.

- Flash memory control register 1 (FLMCR1)
- Flash memory control register 2 (FLMCR2)
- Erase block register 1 (EBR1)
- Flash memory enable register (FENR)

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

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 7.4, Flash Memory Programming/Erasing.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved<br>This bit is always read as 0. |
| 6 | SWE | 0 | R/W | Software Write Enable<br>When this bit is set to 1, flash memory programming/erasing is enabled. When this bit is cleared to 0, other FLMCR1 register bits and all EBR1 bits cannot be set. |
| 5 | ESU | 0 | R/W | Erase Setup<br>When this bit is set to 1, the flash memory changes to the erase setup state. When it is cleared to 0, the erase setup state is cancelled. Set this bit to 1 before setting the E bit to 1 in FLMCR1. |
| 4 | PSU | 0 | R/W | Program Setup<br>When this bit is set to 1, the flash memory changes to the program setup state. When it is cleared to 0, the program setup state is cancelled. Set this bit to 1 before setting the P bit in FLMCR1. |
| 3 | EV | 0 | R/W | Erase-Verify<br>When this bit is set to 1, the flash memory changes to erase-verify mode. When it is cleared to 0, erase-verify mode is cancelled. |
| 2 | PV | 0 | R/W | Program-Verify<br>When this bit is set to 1, the flash memory changes to program-verify mode. When it is cleared to 0, program-verify mode is cancelled. |
| 1 | E | 0 | R/W | Erase<br>When this bit is set to 1, and while the SWE = 1 and ESU = 1 bits are 1, the flash memory changes to erase mode. When it is cleared to 0, erase mode is cancelled. |
| 0 | P | 0 | R/W | Program<br>When this bit is set to 1, and while the SWE = 1 and PSU = 1 bits are 1, the flash memory changes to program mode. When it is cleared to 0, program mode is cancelled. |

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

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 | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | FLER | 0 | R | Flash Memory Error<br>Indicates that an error has occurred during an operation on flash memory (programming or erasing). When FLER is set to 1, flash memory goes to the error-protection state.<br>See 7.5.3, Error Protection, for details. |
| 6 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0. |

### 7.2.3 Erase Block Register 1 (EBR1)

EBR1 specifies the flash memory erase area block. EBR1 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 EBR1 to be automatically cleared to 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 5 | — | All 0 | — | Reserved<br>These bits are always read as 0. |
| 4 | EB4 | 0 | R/W | When this bit is set to 1, 16 kbytes of H'1000 to H'4FFF will be erased. |
| 3 | EB3 | 0 | R/W | When this bit is set to 1, 1 kbyte of H'0C00 to H'0FFF will be erased. |
| 2 | EB2 | 0 | R/W | When this bit is set to 1, 1 kbyte of H'0800 to H'0BFF will be erased. |
| 1 | EB1 | 0 | R/W | When this bit is set to 1, 1 kbyte of H'0400 to H'07FF will be erased. |
| 0 | EB0 | 0 | R/W | When this bit is set to 1, 1 kbyte of H'0000 to H'03FF will be erased. |

### 7.2.4 Flash Memory Enable Register (FENR)

Bit 7 (FLSHE) in FENR enables or disables the CPU access to the flash memory control registers, FLMCR1, FLMCR2, and EBR1.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | FLSHE | 0 | R/W | Flash Memory Control Register Enable<br>Flash memory control registers can be accessed when this bit is set to 1. Flash memory control registers cannot be accessed when this bit is set to 0. |
| 6 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0. |

## 7.3 On-Board Programming Modes

There is a mode for programming/erasing of the flash memory; boot mode, which enables on-board programming/erasing. 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, $\overline{\text{NMI}}$ pin settings, and input level of each port, as shown in table 7.1. 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 SCI3. 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 7.1 Setting Programming Modes**

| TEST | $\overline{\text{NMI}}$ | E10T_0 | PB0 | PB1 | PB2 | LSI State after Reset End |
|---|---|---|---|---|---|---|
| 0 | 1 | X | X | X | X | User Mode |
| 0 | 0 | 1 | X | X | X | Boot Mode |

Legend: X: Don't care.

### 7.3.1 Boot Mode

Table 7.2 shows the boot mode operations between reset end and branching to the programming control program.

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 7.4, 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.
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 7.3.
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 SCR3 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 (PCR22 = 1, P22 = 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 $\overline{\text{NMI}}$ pin. Boot mode is also cleared when a WDT overflow occurs.
8. Do not change the TEST pin and NMI pin input levels in boot mode.

## Table 7.2 Boot Mode Operation

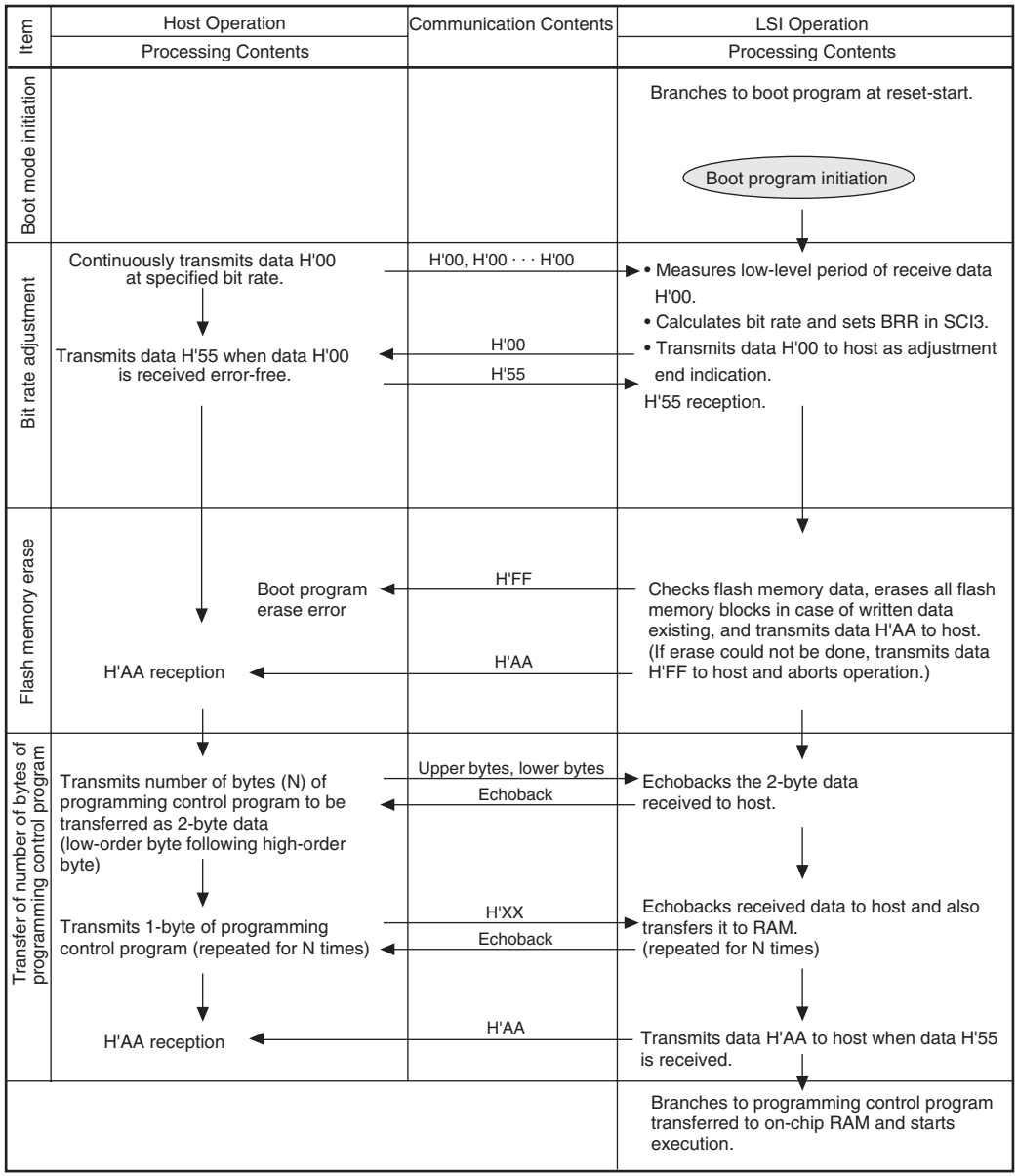


**Table 7.3 System Clock Frequencies for which Automatic Adjustment of LSI Bit Rate is Possible**

| Host Bit Rate | System Clock Frequency Range of LSI |
|---|---|
| 19,200 bps | 16 MHz |
| 9,600 bps | 8 to 16 MHz |
| 4,800 bps | 4 to 16 MHz |
| 2,400 bps | 2 to 16 MHz |

### 7.3.2 Programming/Erasing in User Program Mode

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 7.2 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 7.4, Flash Memory Programming/Erasing.

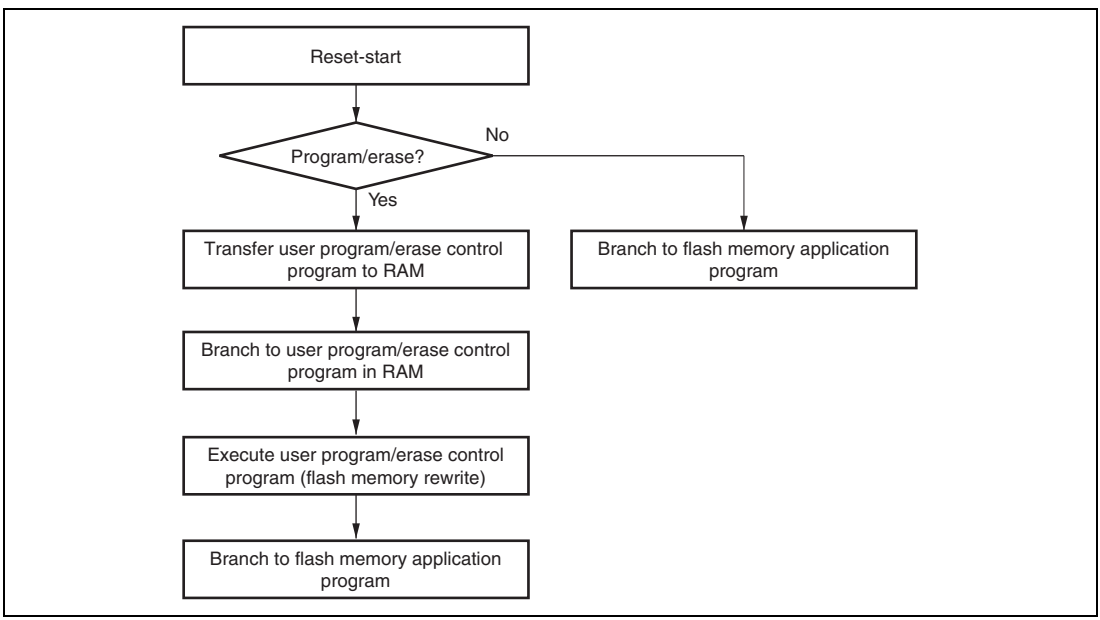



**Figure 7.2 Programming/Erasing Flowchart Example in User Program Mode**

## 7.4 Flash Memory Programming/Erasing

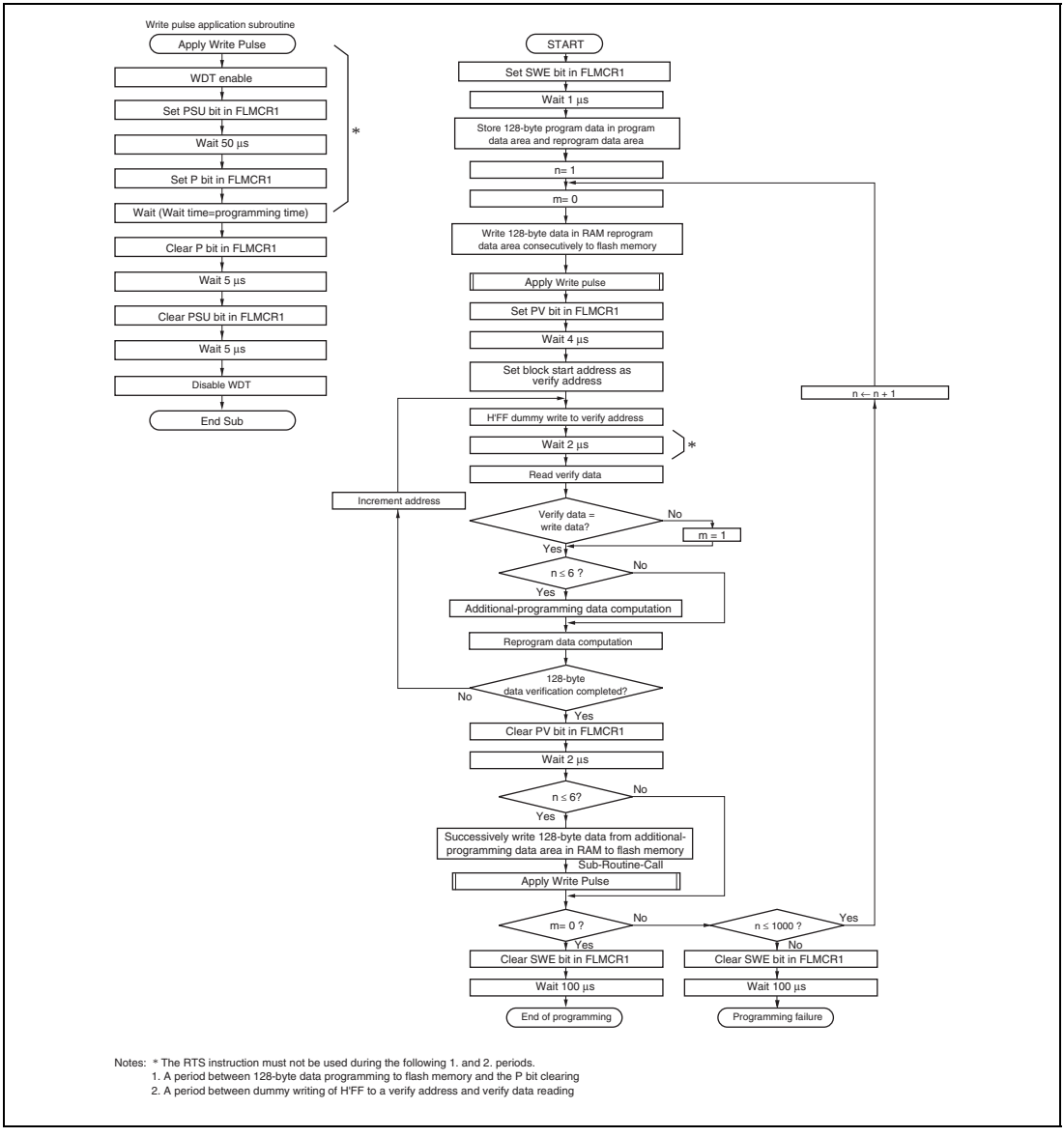
A software method using the CPU is employed to program and erase flash memory in the on-board programming modes. Depending on the FLMCR1 setting, the flash memory operates in one of the following four modes: Program mode, program-verify mode, erase mode, and erase-verify mode. The programming control 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 7.4.1, Program/Program-Verify and section 7.4.2, Erase/Erase-Verify, respectively.

### 7.4.1 Program/Program-Verify

When writing data or programs to the flash memory, the program/program-verify flowchart shown in figure 7.3 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 7.4, and additional programming data computation according to table 7.5.
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. Table 7.6 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 2 bits are B'00. Verify data can be read in words or in longwords 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.




Notes: * The RTS instruction must not be used during the following 1. and 2. periods.
1. A period between 128-byte data programming to flash memory and the P bit clearing
2. A period between dummy writing of H'FF to a verify address and verify data reading

**Figure 7.3 Program/Program-Verify Flowchart**

**Table 7.4 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 7.5 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 7.6 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.

### 7.4.2 Erase/Erase-Verify

When erasing flash memory, the erase/erase-verify flowchart shown in figure 7.4 should be followed.

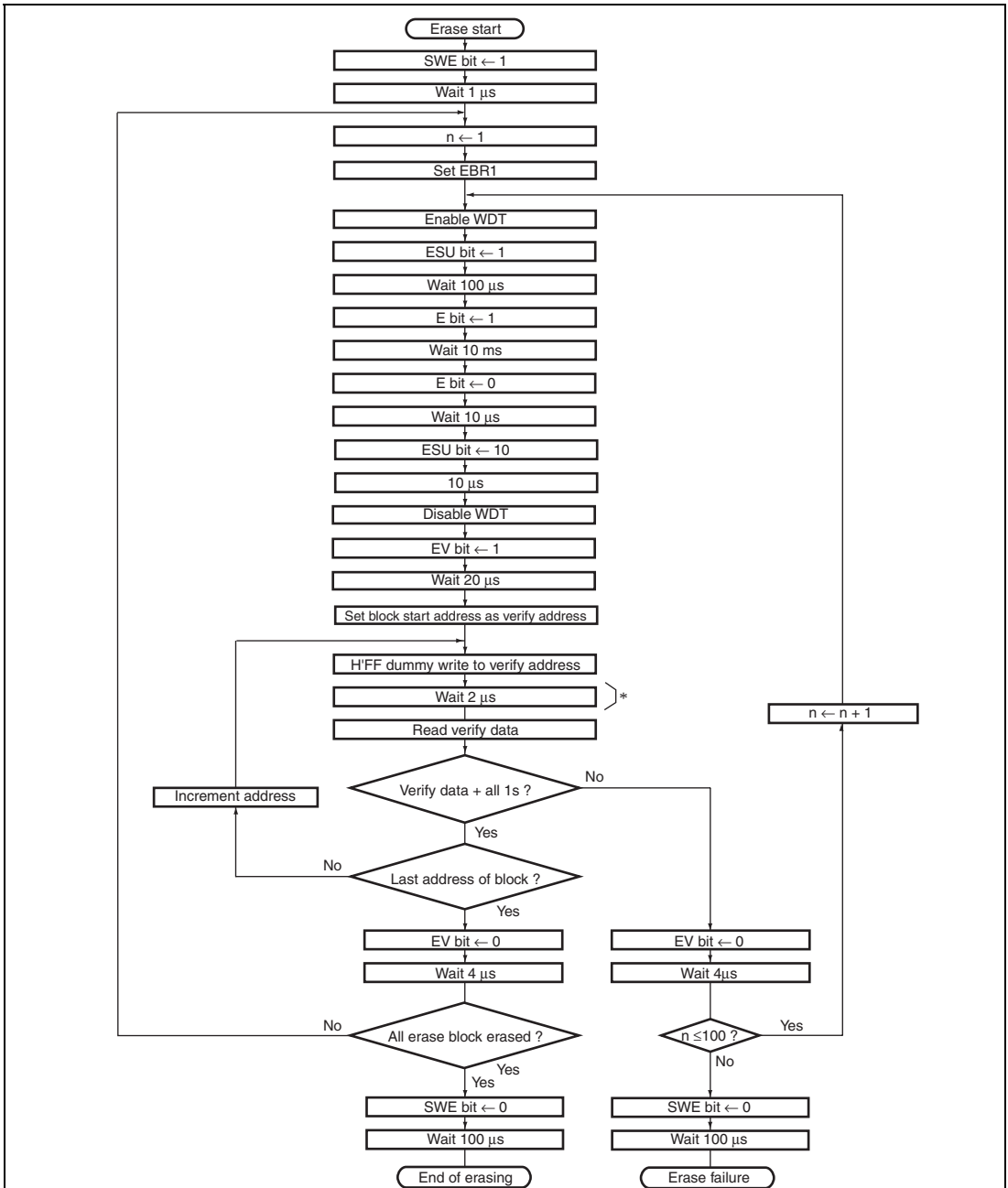
1. Prewriting (setting erase block data to all 0s) is not necessary.
2. Erasing is performed in block units. Make only a single-bit specification in the erase block register (EBR1). To erase multiple blocks, each block must be erased in turn.
3. The time during which the E bit is set to 1 is the flash 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 two bits are B'00. Verify data can be read in longwords 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.

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

All interrupts, including the $\overline{\text{NMI}}$ interrupt, 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.




Note: * The RTS instruction must not be used during a period between dummy writing of H'FF to a verify address and verify data reading.

**Figure 7.4 Erase/Erase-Verify Flowchart**

## 7.5 Program/Erase Protection

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

### 7.5.1 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, subsleep mode or standby mode. Flash memory control register 1 (FLMCR1), flash memory control register 2 (FLMCR2), and erase block register 1 (EBR1) 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.

### 7.5.2 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 1 (EBR1), erase protection can be set for individual blocks. When EBR1 is set to H'00, erase protection is set for all blocks.

### 7.5.3 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 EBR1 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.

# Section 8 RAM

This LSI has 2 kbytes of on-chip high-speed static RAM. The RAM is connected to the CPU by a 16-bit data bus, enabling two-state access by the CPU to both byte data and word data.

Note: The address area H'F780 to H'FB7F must not be accessed while the E7 or E8 is being in use.

# Section 9 I/O Ports

The group of this LSI has twenty-six general I/O ports and four general input-only ports. Port 8 is a large current port, which can drive 20 mA ($@V_{OL}$ = 1.5 V) when a low level signal is output. Any of these ports can become an input port immediately after a reset. They can also be used as I/O pins of the on-chip peripheral modules or external interrupt input pins, and these functions can be switched depending on the register settings. The registers for selecting these functions can be divided into two types: those included in I/O ports and those included in each on-chip peripheral module. General I/O ports are comprised of the port control register for controlling inputs/outputs and the port data register for storing output data and can select inputs/outputs in bit units. For functions in each port, see appendix B.1, I/O Port Block Diagrams. For the execution of bit manipulation instructions to the port control register and port data register, see section 2.8.3, Bit Manipulation Instruction.

## 9.1 Port 1

Port 1 is a general I/O port also functioning as IRQ interrupt input pins and a timer V input pin. Figure 9.1 shows its pin configuration.

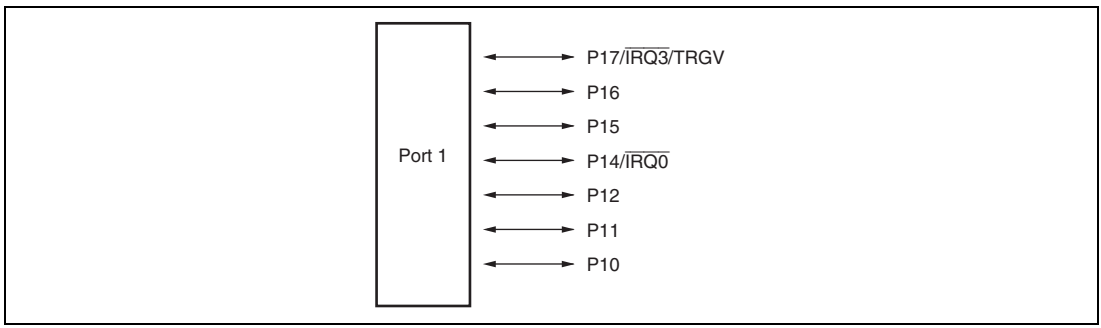



**Figure 9.1 Port 1 Pin Configuration**

Port 1 has the following registers.

- Port mode register 1 (PMR1)
- Port control register 1 (PCR1)
- Port data register 1 (PDR1)
- Port pull-up control register 1 (PUCR1)

### 9.1.1 Port Mode Register 1 (PMR1)

PMR1 switches the functions of pins in port 1 and port 2.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | IRQ3 | 0 | R/W | P17/$\overline{\text{IRQ3}}$/TRGV Pin Function Switch<br>This bit selects whether pin P17/$\overline{\text{IRQ3}}$/TRGV is used as P17 or as $\overline{\text{IRQ3}}$/TRGV.<br>0: General I/O port<br>1: $\overline{\text{IRQ3}}$/TRGV input pin |
| 6 | — | 0 | — | Reserved |
| 5 | — | 0 | — | These bits are always read as 0. |
| 4 | IRQ0 | 0 | R/W | P14/$\overline{\text{IRQ0}}$ Pin Function Switch<br>This bit selects whether pin P14/$\overline{\text{IRQ0}}$ is used as P14 or as $\overline{\text{IRQ0}}$.<br>0: General I/O port<br>1: $\overline{\text{IRQ0}}$ input pin |
| 3 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 2 | — | 0 | R/W | Reserved<br>This bit must always be cleared to 0 (setting to 1 is disabled). |
| 1 | TXD | 0 | R/W | P22/TXD Pin Function Switch<br>This bit selects whether pin P22/TXD is used as P22 or as TXD.<br>0: General I/O port<br>1: TXD output pin |
| 0 | — | 0 | — | Reserved<br>These bits are always read as 0. |

### 9.1.2 Port Control Register 1 (PCR1)

PCR1 selects inputs/outputs in bit units for pins to be used as general I/O ports of port 1.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PCR17 | 0 | W | When the corresponding pin is designated in PMR1 as a general I/O pin, setting a PCR1 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port. |
| 6 | PCR16 | 0 | W | |
| 5 | PCR15 | 0 | W | |
| 4 | PCR14 | 0 | W | Bit 3 is a reserved bit. |
| 3 | — | — | — | |
| 2 | PCR12 | 0 | W | |
| 1 | PCR11 | 0 | W | |
| 0 | PCR10 | 0 | W | |

### 9.1.3 Port Data Register 1 (PDR1)

PDR1 is a general I/O port data register of port 1.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P17 | 0 | R/W | PDR1 stores output data for port 1 pins. |
| 6 | P16 | 0 | R/W | If PDR1 is read while PCR1 bits are set to 1, the value stored in PDR1 are read. If PDR1 is read while PCR1 bits are cleared to 0, the pin states are read regardless of the value stored in PDR1. |
| 5 | P15 | 0 | R/W | |
| 4 | P14 | 0 | R/W | |
| 3 | — | 1 | — | Bit 3 is a reserved bit. This bit is always read as 1. |
| 2 | P12 | 0 | R/W | |
| 1 | P11 | 0 | R/W | |
| 0 | P10 | 0 | R/W | |

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

PUCR1 controls the pull-up MOS in bit units of the pins set as the input ports.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PUCR17 | 0 | R/W | Only bits for which PCR1 is cleared are valid. The pull-up MOS of P17 to P14 and P12 to P10 pins enter the on-state when these bits are set to 1, while they enter the off-state when these bits are cleared to 0.<br>Bit 3 is a reserved bit. This bit is always read as 1. |
| 6 | PUCR16 | 0 | R/W | |
| 5 | PUCR15 | 0 | R/W | |
| 4 | PUCR14 | 0 | R/W | |
| 3 | — | 1 | — | |
| 2 | PUCR12 | 0 | R/W | |
| 1 | PUCR11 | 0 | R/W | |
| 0 | PUCR10 | 0 | R/W | |

### 9.1.5 Pin Functions

The correspondence between the register specification and the port functions is shown below.

**P17/$\overline{\text{IRQ3}}$/TRGV pin**

| Register | PMR1 | PCR1 | |
|---|---|---|---|
| **Bit Name** | **IRQ3** | **PCR17** | **Pin Function** |
| Setting value | 0 | 0 | P17 input pin |
| | | 1 | P17 output pin |
| | 1 | X | $\overline{\text{IRQ3}}$ input/TRGV input pin |

Legend: X: Don't care.

**P16 pin**

| Register | PCR1 | |
|---|---|---|
| **Bit Name** | **PCR16** | **Pin Function** |
| Setting value | 0 | P16 input pin |
| | 1 | P16 output pin |

**P15 pin**

| Register | PCR1 | |
|---|---|---|
| **Bit Name** | **PCR15** | **Pin Function** |
| Setting value | 0 | P15 input pin |
| | 1 | P15 output pin |

**P14/$\overline{\text{IRQ0}}$ pin**

| Register | PMR1 | PCR1 | |
|---|---|---|---|
| **Bit Name** | **IRQ0** | **PCR14** | **Pin Function** |
| Setting value | 0 | 0 | P14 input pin |
| | | 1 | P14 output pin |
| | 1 | X | $\overline{\text{IRQ0}}$ input pin |

Legend: X: Don't care.

**P12 pin**

| Register | PCR1 | |
|---|---|---|
| **Bit Name** | **PCR12** | **Pin Function** |
| **Setting value** | 0 | P12 input pin |
| | 1 | P12 output pin |

**P11 pin**

| Register | PCR1 | |
|---|---|---|
| **Bit Name** | **PCR11** | **Pin Function** |
| **Setting value** | 0 | P11 input pin |
| | 1 | P11 output pin |

**P10 pin**

| Register | PCR1 | |
|---|---|---|
| **Bit Name** | **PCR10** | **Pin Function** |
| Setting value | 0 | P10 input pin |
| | 1 | P10 output pin |

## 9.2 Port 2

Port 2 is a general I/O port also functioning as a SCI3 I/O pin. Each pin of the port 2 is shown in figure 9.2. The register settings of PMR1 and SCI3 have priority for functions of the pins for both uses.

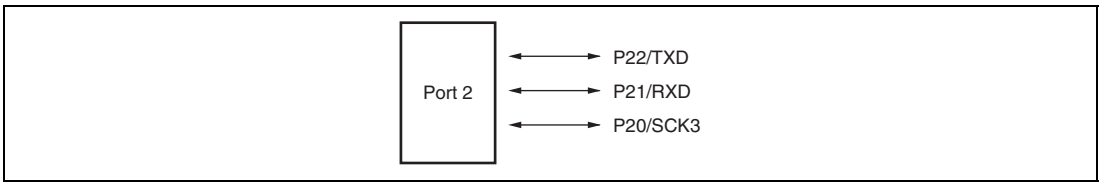



**Figure 9.2 Port 2 Pin Configuration**

Port 2 has the following registers.

- Port control register 2 (PCR2)
- Port data register 2 (PDR2)

### 9.2.1 Port Control Register 2 (PCR2)

PCR2 selects inputs/outputs in bit units for pins to be used as general I/O ports of port 2.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | — | — | Reserved |
| 6 | — | — | — | |
| 5 | — | — | — | |
| 4 | — | — | — | |
| 3 | — | — | — | |
| 2 | PCR22 | 0 | W | When each of the port 2 pins P22 to P20 functions as an general I/O port, setting a PCR2 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port. |
| 1 | PCR21 | 0 | W | |
| 0 | PCR20 | 0 | W | |

### 9.2.2 Port Data Register 2 (PDR2)

PDR2 is a general I/O port data register of port 2.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved |
| 6 | — | 1 | — | These bits are always read as 1. |
| 5 | — | 1 | — | |
| 4 | — | 1 | — | |
| 3 | — | 1 | — | |
| 2 | P22 | 0 | R/W | PDR2 stores output data for port 2 pins. |
| 1 | P21 | 0 | R/W | If PDR2 is read while PCR2 bits are set to 1, the value stored in PDR2 is read. If PDR2 is read while PCR2 bits are cleared to 0, the pin states are read regardless of the value stored in PDR2. |
| 0 | P20 | 0 | R/W | |

### 9.2.3 Pin Functions

The correspondence between the register specification and the port functions is shown below.

**P22/TXD pin**

| Register | PMR1 | PCR2 | |
|---|---|---|---|
| Bit Name | TXD | PCR22 | Pin Function |
| Setting Value | 0 | 0 | P22 input pin |
| | | 1 | P22 output pin |
| | 1 | X | TXD output pin |

Legend: X: Don't care.

**P21/RXD pin**

| Register | SCR3 | PCR2 | |
|---|---|---|---|
| Bit Name | RE | PCR21 | Pin Function |
| Setting Value | 0 | 0 | P21 input pin |
| | | 1 | P21 output pin |
| | 1 | X | RXD input pin |

Legend: X: Don't care.

**P20/SCK3 pin**

| Register | SCR3 | | SMR | PCR2 | |
|---|---|---|---|---|---|
| **Bit Name** | **CKE1** | **CKE0** | **COM** | **PCR20** | **Pin Function** |
| Setting Value | 0 | 0 | 0 | 0 | P20 input pin |
| | | | | 1 | P20 output pin |
| | 0 | 0 | 1 | X | SCK3 output pin |
| | 0 | 1 | X | X | SCK3 output pin |
| | 1 | X | X | X | SCK3 input pin |

Legend: X: Don't care.

## 9.3 Port 5

Port 5 is a general I/O port also functioning as an A/D trigger input pin and wakeup interrupt input pin. Each pin of the port 5 is shown in figure 9.3.

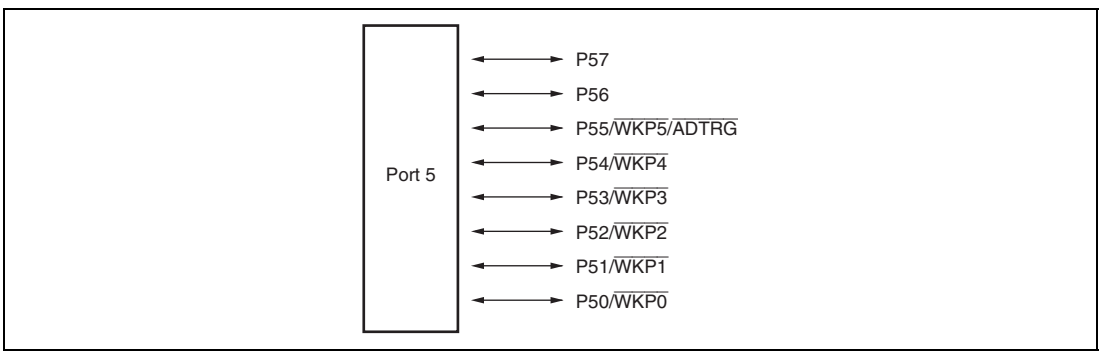



**Figure 9.3 Port 5 Pin Configuration**

Port 5 has the following registers.

- Port mode register 5 (PMR5)
- Port control register 5 (PCR5)
- Port data register 5 (PDR5)
- Port pull-up control register 5 (PUCR5)

### 9.3.1 Port Mode Register 5 (PMR5)

PMR5 switches the functions of pins in port 5.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | POF7 | 0 | R/W | P57 Pin Function Switch<br>0: General I/O port<br>1: NMOS open-drain output |
| 6 | POF6 | 0 | R/W | P56 Pin Function Switch<br>0: General I/O port<br>1: NMOS open-drain output |
| 5 | WKP5 | 0 | R/W | P55/$\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ Pin Function Switch<br>Selects whether pin P55/$\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ is used as P55 or as $\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ input.<br>0: General I/O port<br>1: $\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ input pin |
| 4 | WKP4 | 0 | R/W | P54/$\overline{\text{WKP4}}$ Pin Function Switch<br>Selects whether pin P54/$\overline{\text{WKP4}}$ is used as P54 or as $\overline{\text{WKP4}}$.<br>0: General I/O port<br>1: $\overline{\text{WKP4}}$ input pin |
| 3 | WKP3 | 0 | R/W | P53/$\overline{\text{WKP3}}$ Pin Function Switch<br>Selects whether pin P53/$\overline{\text{WKP3}}$ is used as P53 or as $\overline{\text{WKP3}}$.<br>0: General I/O port<br>1: $\overline{\text{WKP3}}$ input pin |
| 2 | WKP2 | 0 | R/W | P52/$\overline{\text{WKP2}}$ Pin Function Switch<br>Selects whether pin P52/$\overline{\text{WKP2}}$ is used as P52 or as $\overline{\text{WKP2}}$.<br>0: General I/O port<br>1: $\overline{\text{WKP2}}$ input pin |
| 1 | WKP1 | 0 | R/W | P51/$\overline{\text{WKP1}}$ Pin Function Switch<br>Selects whether pin P51/$\overline{\text{WKP1}}$ is used as P51 or as $\overline{\text{WKP1}}$.<br>0: General I/O port<br>1: $\overline{\text{WKP1}}$ input pin |
| 0 | WKP0 | 0 | R/W | P50/$\overline{\text{WKP0}}$ Pin Function Switch<br>Selects whether pin P50/$\overline{\text{WKP0}}$ is used as P50 or as $\overline{\text{WKP0}}$.<br>0: General I/O port<br>1: $\overline{\text{WKP0}}$ input pin |

### 9.3.2 Port Control Register 5 (PCR5)

PCR5 selects inputs/outputs in bit units for pins to be used as general I/O ports of port 5.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PCR57 | 0 | W | When each of the port 5 pins P57 to P50 functions as an general I/O port, setting a PCR5 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port. |
| 6 | PCR56 | 0 | W | |
| 5 | PCR55 | 0 | W | |
| 4 | PCR54 | 0 | W | |
| 3 | PCR53 | 0 | W | |
| 2 | PCR52 | 0 | W | |
| 1 | PCR51 | 0 | W | |
| 0 | PCR50 | 0 | W | |

### 9.3.3 Port Data Register 5 (PDR5)

PDR5 is a general I/O port data register of port 5.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P57 | 0 | R/W | Stores output data for port 5 pins. |
| 6 | P56 | 0 | R/W | If PDR5 is read while PCR5 bits are set to 1, the value stored in PDR5 are read. If PDR5 is read while PCR5 bits are cleared to 0, the pin states are read regardless of the value stored in PDR5. |
| 5 | P55 | 0 | R/W | |
| 4 | P54 | 0 | R/W | |
| 3 | P53 | 0 | R/W | |
| 2 | P52 | 0 | R/W | |
| 1 | P51 | 0 | R/W | |
| 0 | P50 | 0 | R/W | |

### 9.3.4 Port Pull-Up Control Register 5 (PUCR5)

PUCR5 controls the pull-up MOS in bit units of the pins set as the input ports.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved |
| 6 | — | 0 | — | These bits are always read as 0. |
| 5 | PUCR55 | 0 | R/W | Only bits for which PCR5 is cleared are valid. The pull-up MOS of the corresponding pins enter the on-state when these bits are set to 1, while they enter the off-state when these bits are cleared to 0. |
| 4 | PUCR54 | 0 | R/W | |
| 3 | PUCR53 | 0 | R/W | |
| 2 | PUCR52 | 0 | R/W | |
| 1 | PUCR51 | 0 | R/W | |
| 0 | PUCR50 | 0 | R/W | |

### 9.3.5 Pin Functions

The correspondence between the register specification and the port functions is shown below.

**P57 pin**

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **POF7** | **PCR57** | **Pin Function** |
| Setting Value | X | 0 | P57 input pin |
| | 0 | 1 | CMOS output |
| | 1 | 1 | NMOS open-drain output |

Legend: X: Don't care.

**P56 pin**

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **POF6** | **PCR56** | **Pin Function** |
| Setting Value | X | 0 | P56 input pin |
| | 0 | 1 | CMOS output |
| | 1 | 1 | NMOS open-drain output |

Legend: X: Don't care.

### P55/$\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ pin

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP5** | **PCR55** | **Pin Function** |
| Setting Value | 0 | 0 | P55 input pin |
| | | 1 | P55 output pin |
| | 1 | X | $\overline{\text{WKP5}}$/$\overline{\text{ADTRG}}$ input pin |

Legend: X: Don't care.

### P54/$\overline{\text{WKP4}}$ pin

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP4** | **PCR54** | **Pin Function** |
| Setting Value | 0 | 0 | P54 input pin |
| | | 1 | P54 output pin |
| | 1 | X | $\overline{\text{WKP4}}$ input pin |

Legend: X: Don't care.

### P53/$\overline{\text{WKP3}}$ pin

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP3** | **PCR53** | **Pin Function** |
| Setting Value | 0 | 0 | P53 input pin |
| | | 1 | P53 output pin |
| | 1 | X | $\overline{\text{WKP3}}$ input pin |

Legend: X: Don't care.

### P52/$\overline{\text{WKP2}}$ pin

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP2** | **PCR52** | **Pin Function** |
| Setting Value | 0 | 0 | P52 input pin |
| | | 1 | P52 output pin |
| | 1 | X | $\overline{\text{WKP2}}$ input pin |

Legend: X: Don't care.

**P51/$\overline{\text{WKP1}}$ pin**

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP1** | **PCR51** | **Pin Function** |
| Setting Value | 0 | 0 | P51 input pin |
| | | 1 | P51 output pin |
| | 1 | X | $\overline{\text{WKP1}}$ input pin |

Legend: X: Don't care.

**P50/$\overline{\text{WKP0}}$ pin**

| Register | PMR5 | PCR5 | |
|---|---|---|---|
| **Bit Name** | **WKP0** | **PCR50** | **Pin Function** |
| Setting Value | 0 | 0 | P50 input pin |
| | | 1 | P50 output pin |
| | 1 | X | $\overline{\text{WKP0}}$ input pin |

Legend: X: Don't care.

## 9.4 Port 7

Port 7 is a general I/O port also functioning as a timer V I/O pin. Each pin of the port 7 is shown in figure 9.4. The register setting of TCSRV in timer V has priority for functions of pin P76/TMOV. The pins, P75/TMCIV and P74/TMRIV, are also functioning as timer V input ports that are connected to the timer V regardless of the register setting of port 7.

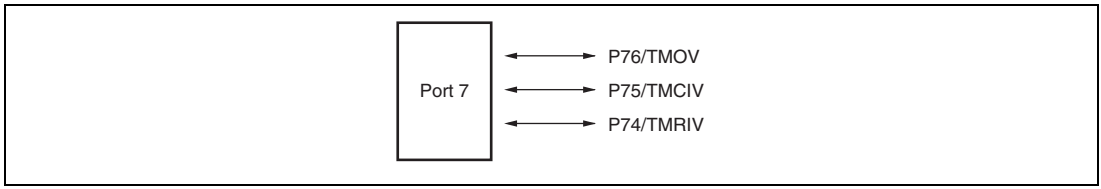



**Figure 9.4 Port 7 Pin Configuration**

Port 7 has the following registers.

- Port control register 7 (PCR7)
- Port data register 7 (PDR7)

### 9.4.1 Port Control Register 7 (PCR7)

PCR7 selects inputs/outputs in bit units for pins to be used as general I/O ports of port 7.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | — | — | Reserved |
| 6 | PCR76 | 0 | W | Setting a PCR7 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port. Note that the TCSRV setting of the timer V has priority for deciding input/output direction of the P76/TMOV pin. |
| 5 | PCR75 | 0 | W | |
| 4 | PCR74 | 0 | W | |
| 3 | — | — | — | Reserved |
| 2 | — | — | — | |
| 1 | — | — | — | |
| 0 | — | — | — | |

### 9.4.2 Port Data Register 7 (PDR7)

PDR7 is a general I/O port data register of port 7.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 6 | P76 | 0 | R/W | PDR7 stores output data for port 7 pins. |
| 5 | P75 | 0 | R/W | If PDR7 is read while PCR7 bits are set to 1, the value stored in PDR7 is read. If PDR7 is read while PCR7 bits are cleared to 0, the pin states are read regardless of the value stored in PDR7. |
| 4 | P74 | 0 | R/W | |
| 3 | — | 1 | — | Reserved |
| 2 | — | 1 | — | These bits are always read as 1. |
| 1 | — | 1 | — | |
| 0 | — | 1 | — | |

### 9.4.3 Pin Functions

The correspondence between the register specification and the port functions is shown below.

**P76/TMOV pin**

| Register | TCSRV | PCR7 | |
|---|---|---|---|
| **Bit Name** | **OS3 to OS0** | **PCR76** | **Pin Function** |
| Setting Value | 0000 | 0 | P76 input pin |
| | | 1 | P76 output pin |
| | Other than the above values | X | TMOV output pin |

Legend: X: Don't care.

**P75/TMCIV pin**

| Register | PCR7 | |
|---|---|---|
| **Bit Name** | **PCR75** | **Pin Function** |
| Setting Value | 0 | P75 input/TMCIV input pin |
| | 1 | P75 output/TMCIV input pin |

**P74/TMRIV pin**

| Register | PCR7 | |
|---|---|---|
| **Bit Name** | **PCR74** | **Pin Function** |
| Setting Value | 0 | P74 input/TMRIV input pin |
| | 1 | P74 output/TMRIV input pin |

## 9.5 Port 8

Port 8 is a general I/O port also functioning as a timer W I/O pin. Each pin of the port 8 is shown in figure 9.5. The register setting of the timer W has priority for functions of the pins P84/FTIOD, P83/FTIOC, P82/FTIOB, and P81/FTIOA. The P80/FTCI pin also functions as a timer W input port that is connected to the timer W regardless of the register setting of port 8.

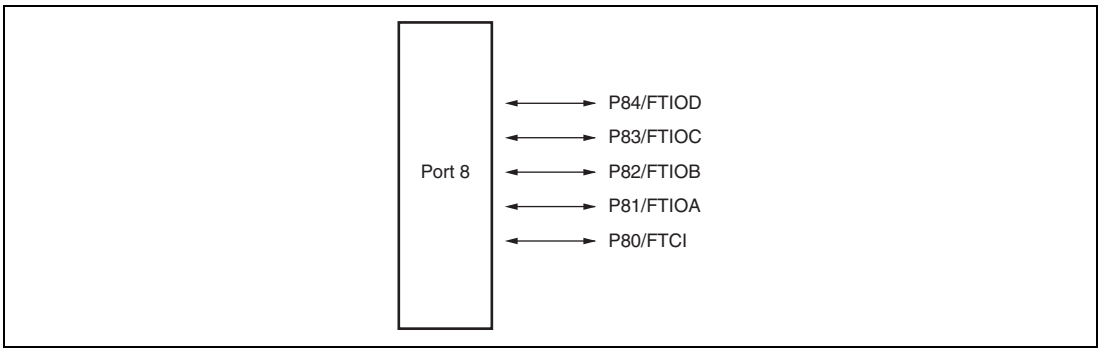



**Figure 9.5 Port 8 Pin Configuration**

Port 8 has the following registers.

- Port control register 8 (PCR8)
- Port data register 8 (PDR8)

### 9.5.1 Port Control Register 8 (PCR8)

PCR8 selects inputs/outputs in bit units for pins to be used as general I/O ports of port 8.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | — | — | Reserved |
| 6 | — | — | — | |
| 5 | — | — | — | |
| 4 | PCR84 | 0 | W | When each of the port 8 pins P84 to P80 functions as an general I/O port, setting a PCR8 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port. |
| 3 | PCR83 | 0 | W | |
| 2 | PCR82 | 0 | W | |
| 1 | PCR81 | 0 | W | |
| 0 | PCR80 | 0 | W | |

### 9.5.2 Port Data Register 8 (PDR8)

PDR8 is a general I/O port data register of port 8.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved |
| 6 | — | 0 | — | |
| 5 | — | 0 | — | |
| 4 | P84 | 0 | R/W | PDR8 stores output data for port 8 pins. |
| 3 | P83 | 0 | R/W | If PDR8 is read while PCR8 bits are set to 1, the value stored in PDR8 is read. If PDR8 is read while PCR8 bits are cleared to 0, the pin states are read regardless of the value stored in PDR8. |
| 2 | P82 | 0 | R/W | |
| 1 | P81 | 0 | R/W | |
| 0 | P80 | 0 | R/W | |

### 9.5.3 Pin Functions

The correspondence between the register specification and the port functions is shown below.

**P84/FTIOD pin**

| Register | TIOR1 | | | PCR8 | |
|---|---|---|---|---|---|
| **Bit Name** | **IOD2** | **IOD1** | **IOD0** | **PCR84** | **Pin Function** |
| Setting Value | 0 | 0 | 0 | 0 | P84 input/FTIOD input pin |
| | | | | 1 | P84 output/FTIOD input pin |
| | 0 | 0 | 1 | X | FTIOD output pin |
| | 0 | 1 | X | X | FTIOD output pin |
| | 1 | X | X | 0 | P84 input/FTIOD input pin |
| | | | | 1 | P84 output/FTIOD input pin |

Legend: X: Don't care.

### P83/FTIOC pin

| Register | TIOR1 | | | PCR8 | |
|---|---|---|---|---|---|
| **Bit Name** | **IOC2** | **IOC1** | **IOC0** | **PCR83** | **Pin Function** |
| Setting Value | 0 | 0 | 0 | 0 | P83 input/FTIOC input pin |
| | | | | 1 | P83 output/FTIOC input pin |
| | 0 | 0 | 1 | X | FTIOC output pin |
| | 0 | 1 | X | X | FTIOC output pin |
| | 1 | X | X | 0 | P83 input/FTIOC input pin |
| | | | | 1 | P83 output/FTIOC input pin |

Legend: X: Don't care.

### P82/FTIOB pin

| Register | TIOR0 | | | PCR8 | |
|---|---|---|---|---|---|
| **Bit Name** | **IOB2** | **IOB1** | **IOB0** | **PCR82** | **Pin Function** |
| Setting Value | 0 | 0 | 0 | 0 | P82 input/FTIOB input pin |
| | | | | 1 | P82 output/FTIOB input pin |
| | 0 | 0 | 1 | X | FTIOB output pin |
| | 0 | 1 | X | X | FTIOB output pin |
| | 1 | X | X | 0 | P82 input/FTIOB input pin |
| | | | | 1 | P82 output/FTIOB input pin |

Legend: X: Don't care.

### P81/FTIOA pin

| Register | TIOR0 | | | PCR8 | |
|---|---|---|---|---|---|
| **Bit Name** | **IOA2** | **IOA1** | **IOA0** | **PCR81** | **Pin Function** |
| Setting Value | 0 | 0 | 0 | 0 | P81 input/FTIOA input pin |
| | | | | 1 | P81 output/FTIOA input pin |
| | 0 | 0 | 1 | X | FTIOA output pin |
| | 0 | 1 | X | X | FTIOA output pin |
| | 1 | X | X | 0 | P81 input/FTIOA input pin |
| | | | | 1 | P81 output/FTIOA input pin |

Legend: X: Don't care.

**P80/FTCI pin**

| Register | PCR8 | |
|---|---|---|
| **Bit Name** | **PCR80** | **Pin Function** |
| Setting Value | 0 | P80 input/FTCI input pin |
| | 1 | P80 output/FTCI input pin |

## 9.6 Port B

Port B is an input port also functioning as an A/D converter analog input pin. Each pin of the port B is shown in figure 9.6.

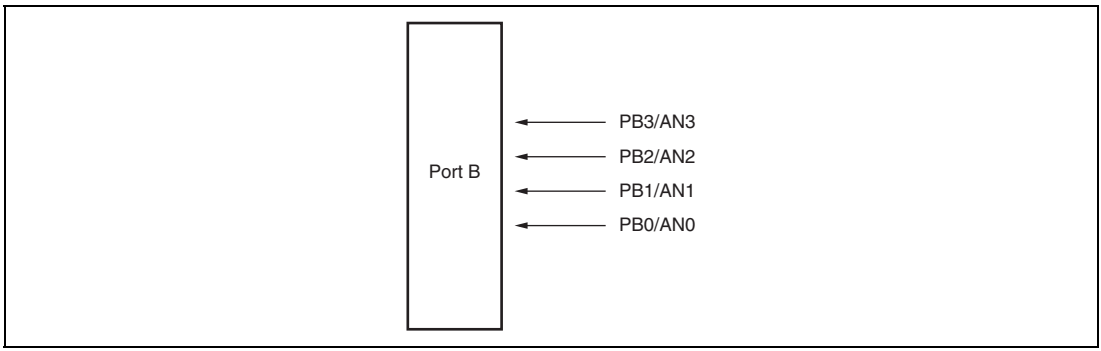



**Figure 9.6 Port B Pin Configuration**

Port B has the following register.

- Port data register B (PDRB)

### 9.6.1 Port Data Register B (PDRB)

PDRB is a general input-only port data register of port B.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | — | — | Reserved |
| 6 | — | — | — | |
| 5 | — | — | — | |
| 4 | — | — | — | |
| 3 | PB3 | — | R | The input value of each pin is read by reading this register. |
| 2 | PB2 | — | R | However, if a port B pin is designated as an analog input channel by ADCSR in A/D converter, 0 is read. |
| 1 | PB1 | — | R | |
| 0 | PB0 | — | R | |

# Section 10 Timer V

Timer V is an 8-bit timer based on an 8-bit counter. Timer V counts external events. Compare-match signals with two registers can also be used to reset the counter, request an interrupt, or output a pulse signal with an arbitrary duty cycle. Counting can be initiated by a trigger input at the TRGV pin, enabling pulse output control to be synchronized to the trigger, with an arbitrary delay from the trigger input. Figure 10.1 shows a block diagram of timer V.

## 10.1 Features

- Choice of seven clock signals is available.
  Choice of six internal clock sources (ø/128, ø/64, ø/32, ø/16, ø/8, ø/4) or an external clock.
- Counter can be cleared by compare match A or B, or by an external reset signal. If the count stop function is selected, the counter can be halted when cleared.
- Timer output is controlled by two independent compare match signals, enabling pulse output with an arbitrary duty cycle, PWM output, and other applications.
- Three interrupt sources: compare match A, compare match B, timer overflow
- Counting can be initiated by trigger input at the TRGV pin. The rising edge, falling edge, or both edges of the TRGV input can be selected.

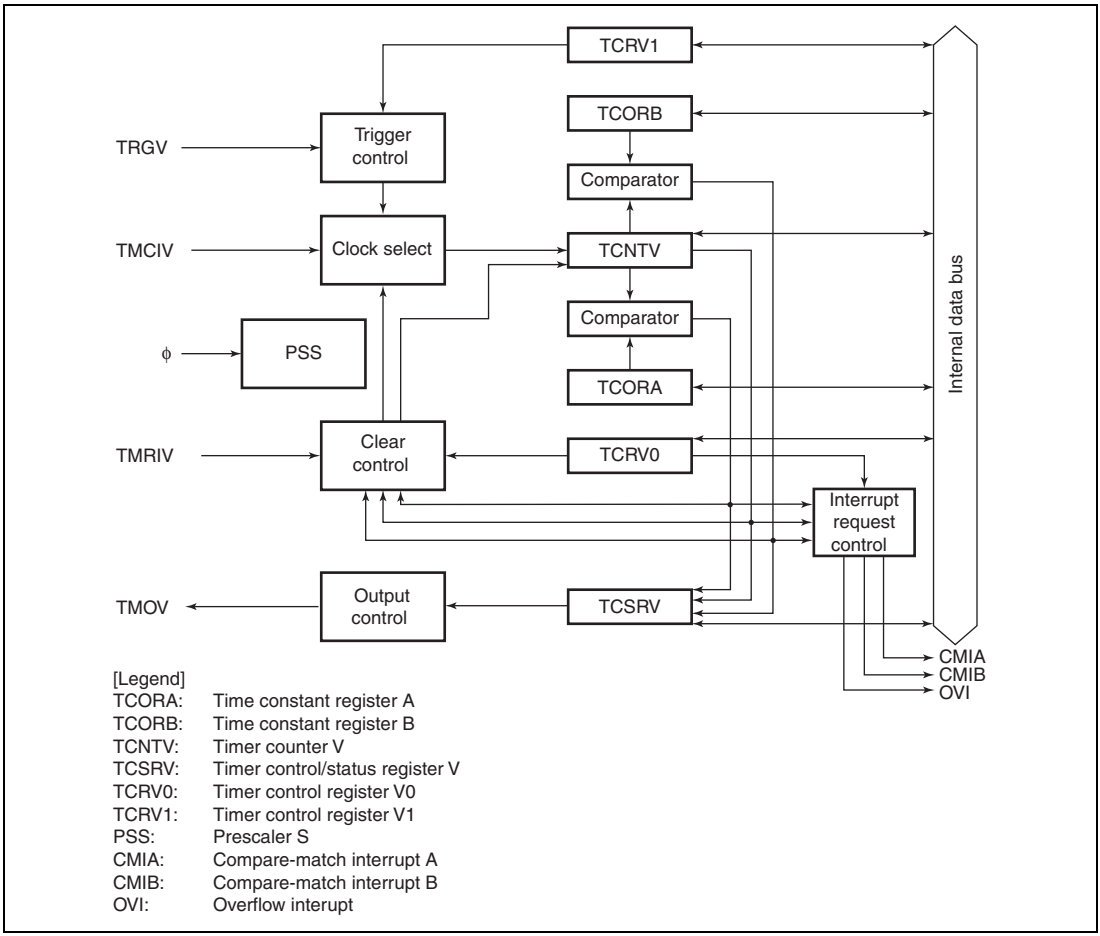



**Figure 10.1 Block Diagram of Timer V**

## 10.2 Input/Output Pins

Table 10.1 shows the timer V pin configuration.

**Table 10.1 Pin Configuration**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Timer V output | TMOV | Output | Timer V waveform output |
| Timer V clock input | TMCIV | Input | Clock input to TCNTV |
| Timer V reset input | TMRIV | Input | External input to reset TCNTV |
| Trigger input | TRGV | Input | Trigger input to initiate counting |

## 10.3 Register Descriptions

Time V has the following registers.

- Timer counter V (TCNTV)
- Timer constant register A (TCORA)
- Timer constant register B (TCORB)
- Timer control register V0 (TCRV0)
- Timer control/status register V (TCSRV)
- Timer control register V1 (TCRV1)

### 10.3.1 Timer Counter V (TCNTV)

TCNTV is an 8-bit up-counter. The clock source is selected by bits CKS2 to CKS0 in timer control register V0 (TCRV0). The TCNTV value can be read and written by the CPU at any time. TCNTV can be cleared by an external reset input signal, or by compare match A or B. The clearing signal is selected by bits CCLR1 and CCLR0 in TCRV0.

When TCNTV overflows, OVF is set to 1 in timer control/status register V (TCSRV).

TCNTV is initialized to H'00.

### 10.3.2 Time Constant Registers A and B (TCORA, TCORB)

TCORA and TCORB have the same function.

TCORA and TCORB are 8-bit read/write registers.

TCORA and TCNTV are compared at all times. When the TCORA and TCNTV contents match, CMFA is set to 1 in TCSRV. If CMIEA is also set to 1 in TCRV0, a CPU interrupt is requested. Note that they must not be compared during the T3 state of a TCORA write cycle.

Timer output from the TMOV pin can be controlled by the identifying signal (compare match A) and the settings of bits OS3 to OS0 in TCSRV.

TCORA and TCORB are initialized to H'FF.

### 10.3.3 Timer Control Register V0 (TCRV0)

TCRV0 selects the input clock signals of TCNTV, specifies the clearing conditions of TCNTV, and controls each interrupt request.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMIEB | 0 | R/W | Compare Match Interrupt Enable B<br>When this bit is set to 1, interrupt request from the CMFB bit in TCSRV is enabled. |
| 6 | CMIEA | 0 | R/W | Compare Match Interrupt Enable A<br>When this bit is set to 1, interrupt request from the CMFA bit in TCSRV is enabled. |
| 5 | OVIE | 0 | R/W | Timer Overflow Interrupt Enable<br>When this bit is set to 1, interrupt request from the OVF bit in TCSRV is enabled. |
| 4 | CCLR1 | 0 | R/W | Counter Clear 1 and 0 |
| 3 | CCLR0 | 0 | R/W | These bits specify the clearing conditions of TCNTV.<br>00: Clearing is disabled<br>01: Cleared by compare match A<br>10: Cleared by compare match B<br>11: Cleared on the rising edge of the TMRIV pin. The operation of TCNTV after clearing depends on TRGE in TCRV1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | CKS2 | 0 | R/W | Clock Select 2 to 0 |
| 1 | CKS1 | 0 | R/W | These bits select clock signals to input to TCNTV and the counting condition in combination with ICKS0 in TCRV1. |
| 0 | CKS0 | 0 | R/W | Refer to table 10.2. |

**Table 10.2 Clock Signals to Input to TCNTV and Counting Conditions**

| TCRV0 | | | TCRV1 | |
|---|---|---|---|---|
| Bit 2 | Bit 1 | Bit 0 | Bit 0 | |
| CKS2 | CKS1 | CKS0 | ICKS0 | Description |
| 0 | 0 | 0 | — | Clock input prohibited |
| | | 1 | 0 | Internal clock: counts on $\phi$/4, falling edge |
| | | | 1 | Internal clock: counts on $\phi$/8, falling edge |
| | 1 | 0 | 0 | Internal clock: counts on $\phi$/16, falling edge |
| | | | 1 | Internal clock: counts on $\phi$/32, falling edge |
| | | 1 | 0 | Internal clock: counts on $\phi$/64, falling edge |
| | | | 1 | Internal clock: counts on $\phi$/128, falling edge |
| 1 | 0 | 0 | — | Clock input prohibited |
| | | 1 | — | External clock: counts on rising edge |
| | 1 | 0 | — | External clock: counts on falling edge |
| | | 1 | — | External clock: counts on rising and falling edge |

### 10.3.4 Timer Control/Status Register V (TCSRV)

TCSRV indicates the status flag and controls outputs by using a compare match.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMFB | 0 | R/W | Compare Match Flag B<br>Setting condition:<br>When the TCNTV value matches the TCORB value<br>Clearing condition:<br>After reading CMFB = 1, cleared by writing 0 to CMFB |
| 6 | CMFA | 0 | R/W | Compare Match Flag A<br>Setting condition:<br>When the TCNTV value matches the TCORA value<br>Clearing condition:<br>After reading CMFA = 1, cleared by writing 0 to CMFA |
| 5 | OVF | 0 | R/W | Timer Overflow Flag<br>Setting condition:<br>When TCNTV overflows from H'FF to H'00<br>Clearing condition:<br>After reading OVF = 1, cleared by writing 0 to OVF |
| 4 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 3 | OS3 | 0 | R/W | Output Select 3 and 2 |
| 2 | OS2 | 0 | R/W | These bits select an output method for the TMOV pin by the compare match of TCORB and TCNTV.<br>00: No change<br>01: 0 output<br>10: 1 output<br>11: Output toggles |
| 1 | OS1 | 0 | R/W | Output Select 1 and 0 |
| 0 | OS0 | 0 | R/W | These bits select an output method for the TOMV pin by the compare match of TCORA and TCNTV.<br>00: No change<br>01: 0 output<br>10: 1 output<br>11: Output toggles |

OS3 and OS2 select the output level for compare match B. OS1 and OS0 select the output level for compare match A. The two output levels can be controlled independently. After a reset, the timer output is 0 until the first compare match.

### 10.3.5 Timer Control Register V1 (TCRV1)

TCRV1 selects the edge at the TRGV pin, enables TRGV input, and selects the clock input to TCNTV.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 5 | — | All 1 | — | Reserved<br>These bits are always read as 1. |
| 4 | TVEG1 | 0 | R/W | TRGV Input Edge Select |
| 3 | TVEG0 | 0 | R/W | These bits select the TRGV input edge.<br>00: TRGV trigger input is prohibited<br>01: Rising edge is selected<br>10: Falling edge is selected<br>11: Rising and falling edges are both selected |
| 2 | TRGE | 0 | R/W | TCNTV starts counting up by the input of the edge which is selected by TVEG1 and TVEG0.<br>0: Disables starting counting-up TCNTV by the input of the TRGV pin and halting counting-up TCNTV when TCNTV is cleared by a compare match.<br>1: Enables starting counting-up TCNTV by the input of the TRGV pin and halting counting-up TCNTV when TCNTV is cleared by a compare match. |
| 1 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 0 | ICKS0 | 0 | R/W | Internal Clock Select 0<br>This bit selects clock signals to input to TCNTV in combination with CKS2 to CKS0 in TCRV0.<br>Refer to table 10.2. |

## 10.4 Operation

### 10.4.1 Timer V Operation

1. According to table 10.2, six internal/external clock signals output by prescaler S can be selected as the timer V operating clock signals. When the operating clock signal is selected, TCNTV starts counting-up. Figure 10.2 shows the count timing with an internal clock signal selected, and figure 10.3 shows the count timing with both edges of an external clock signal selected.
2. When TCNTV overflows (changes from H'FF to H'00), the overflow flag (OVF) in TCRV0 will be set. The timing at this time is shown in figure 10.4. An interrupt request is sent to the CPU when OVIE in TCRV0 is 1.
3. TCNTV is constantly compared with TCORA and TCORB. Compare match flag A or B (CMFA or CMFB) is set to 1 when TCNTV matches TCORA or TCORB, respectively. The compare-match signal is generated in the last state in which the values match. Figure 10.5 shows the timing. An interrupt request is generated for the CPU when CMIEA or CMIEB in TCRV0 is 1.
4. When a compare match A or B is generated, the TMOV responds with the output value selected by bits OS3 to OS0 in TCSRV. Figure 10.6 shows the timing when the output is toggled by compare match A.
5. When CCLR1 or CCLR0 in TCRV0 is 01 or 10, TCNTV can be cleared by the corresponding compare match. Figure 10.7 shows the timing.
6. When CCLR1 or CCLR0 in TCRV0 is 11, TCNTV can be cleared by the rising edge of the input of TMRIV pin. A TMRIV input pulse-width of at least 1.5 system clocks is necessary. Figure 10.8 shows the timing.
7. When a counter-clearing source is generated with TRGE in TCRV1 set to 1, the counting-up is halted as soon as TCNTV is cleared. TCNTV resumes counting-up when the edge selected by TVEG1 or TVEG0 in TCRV1 is input from the TGRV pin.

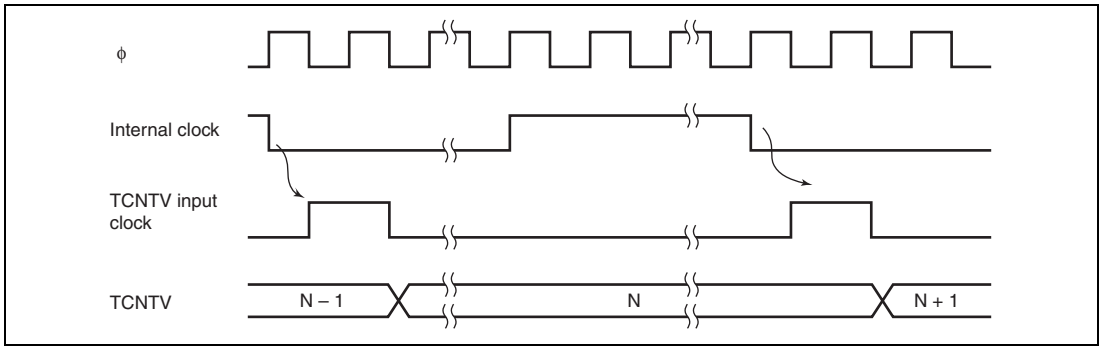



**Figure 10.2 Increment Timing with Internal Clock**

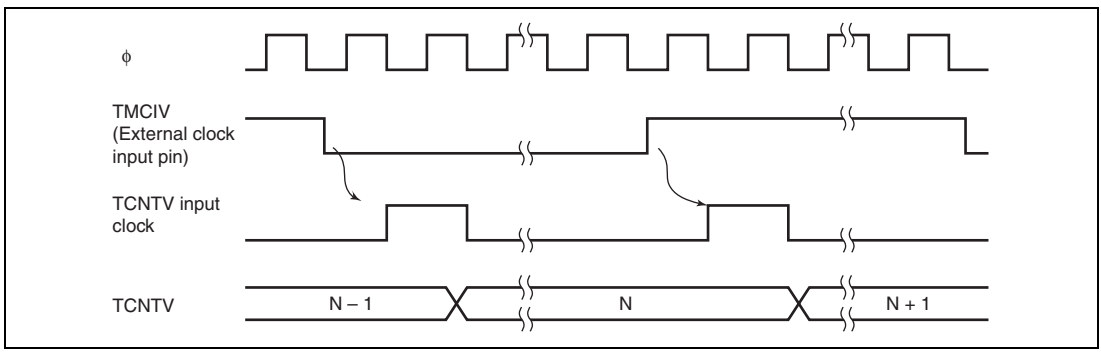



**Figure 10.3 Increment Timing with External Clock**

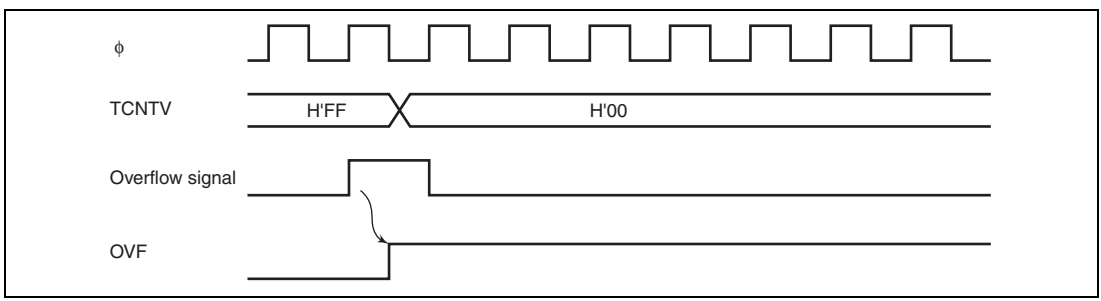



**Figure 10.4 OVF Set Timing**

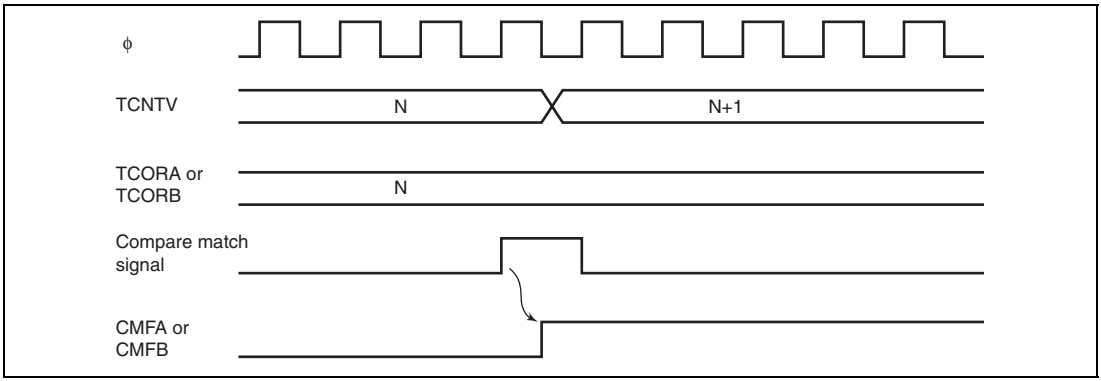



**Figure 10.5 CMFA and CMFB Set Timing**

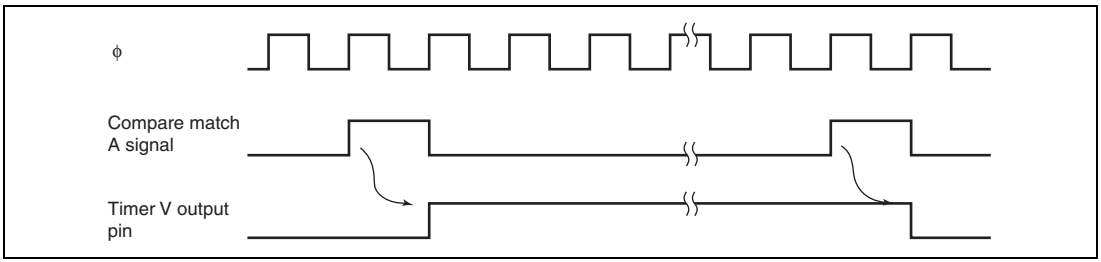



**Figure 10.6 TMOV Output Timing**

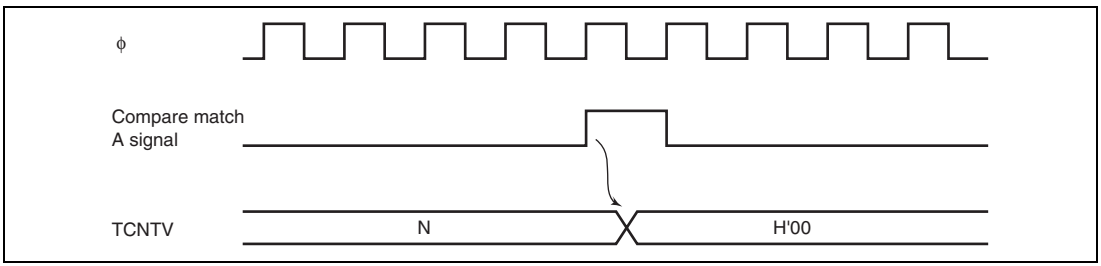



**Figure 10.7 Clear Timing by Compare Match**

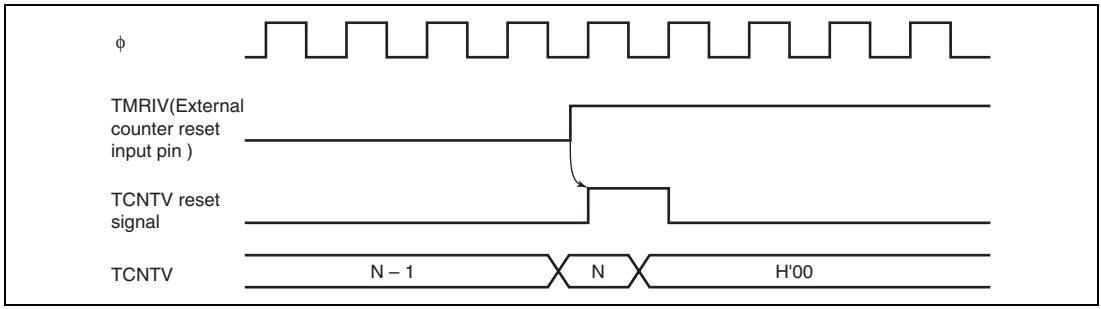



**Figure 10.8 Clear Timing by TMRIV Input**

## 10.5 Timer V Application Examples

### 10.5.1 Pulse Output with Arbitrary Duty Cycle

Figure 10.9 shows an example of output of pulses with an arbitrary duty cycle.

1. Set bits CCLR1 and CCLR0 in TCRV0 so that TCNTV will be cleared by compare match with TCORA.
2. Set bits OS3 to OS0 in TCSRV so that the output will go to 1 at compare match with TCORA and to 0 at compare match with TCORB.
3. Set bits CKS2 to CKS0 in TCRV0 and bit ICKS0 in TCRV1 to select the desired clock source.
4. With these settings, a waveform is output without further software intervention, with a period determined by TCORA and a pulse width determined by TCORB.

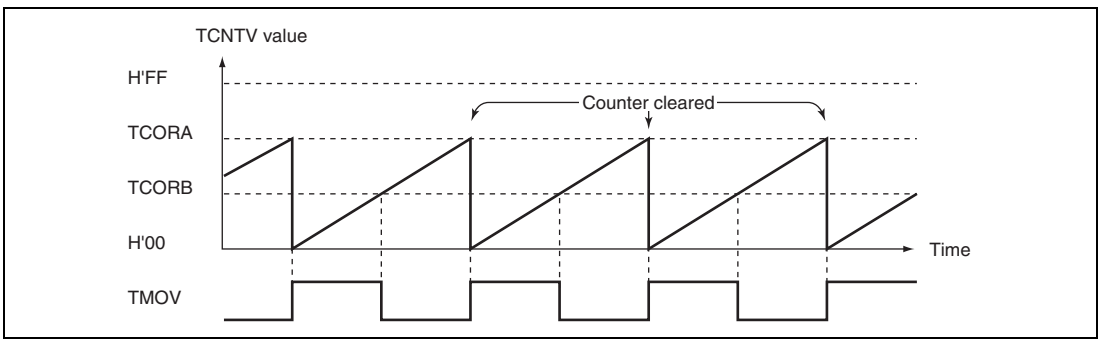



**Figure 10.9 Pulse Output Example**

### 10.5.2 Pulse Output with Arbitrary Pulse Width and Delay from TRGV Input

The trigger function can be used to output a pulse with an arbitrary pulse width at an arbitrary delay from the TRGV input, as shown in figure 10.10. To set up this output:

1. Set bits CCLR1 and CCLR0 in TCRV0 so that TCNTV will be cleared by compare match with TCORB.
2. Set bits OS3 to OS0 in TCSRV so that the output will go to 1 at compare match with TCORA and to 0 at compare match with TCORB.
3. Set bits TVEG1 and TVEG0 in TCRV1 and set TRGE to select the falling edge of the TRGV input.
4. Set bits CKS2 to CKS0 in TCRV0 and bit ICKS0 in TCRV1 to select the desired clock source.
5. After these settings, a pulse waveform will be output without further software intervention, with a delay determined by TCORA from the TRGV input, and a pulse width determined by (TCORB – TCORA).

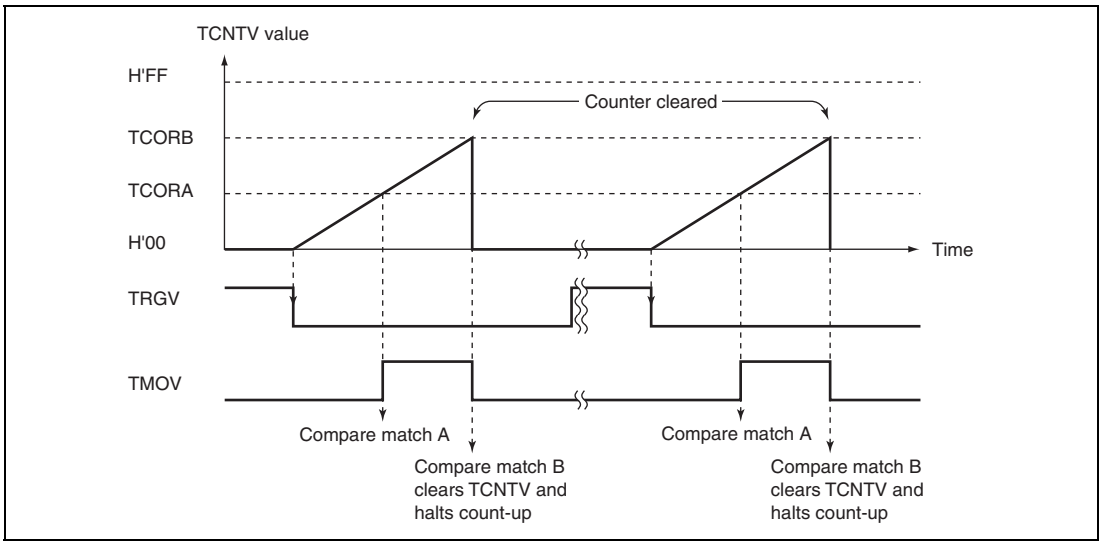



**Figure 10.10 Example of Pulse Output Synchronized to TRGV Input**

## 10.6 Usage Notes

The following types of contention or operation can occur in timer V operation.

1. Writing to registers is performed in the T3 state of a TCNTV write cycle. If a TCNTV clear signal is generated in the T3 state of a TCNTV write cycle, as shown in figure 10.11, clearing takes precedence and the write to the counter is not carried out. If counting-up is generated in the T3 state of a TCNTV write cycle, writing takes precedence.
2. If a compare match is generated in the T3 state of a TCORA or TCORB write cycle, the write to TCORA or TCORB takes precedence and the compare match signal is inhibited. Figure 10.12 shows the timing.
3. If compare matches A and B occur simultaneously, any conflict between the output selections for compare match A and compare match B is resolved by the following priority: toggle output > output 1 > output 0.
4. Depending on the timing, TCNTV may be incremented by a switch between different internal clock sources. When TCNTV is internally clocked, an increment pulse is generated from the falling edge of an internal clock signal, that is divided system clock (ϕ). Therefore, as shown in figure 10.3 the switch is from a high clock signal to a low clock signal, the switchover is seen as a falling edge, causing TCNTV to increment. TCNTV can also be incremented by a switch between internal and external clocks.

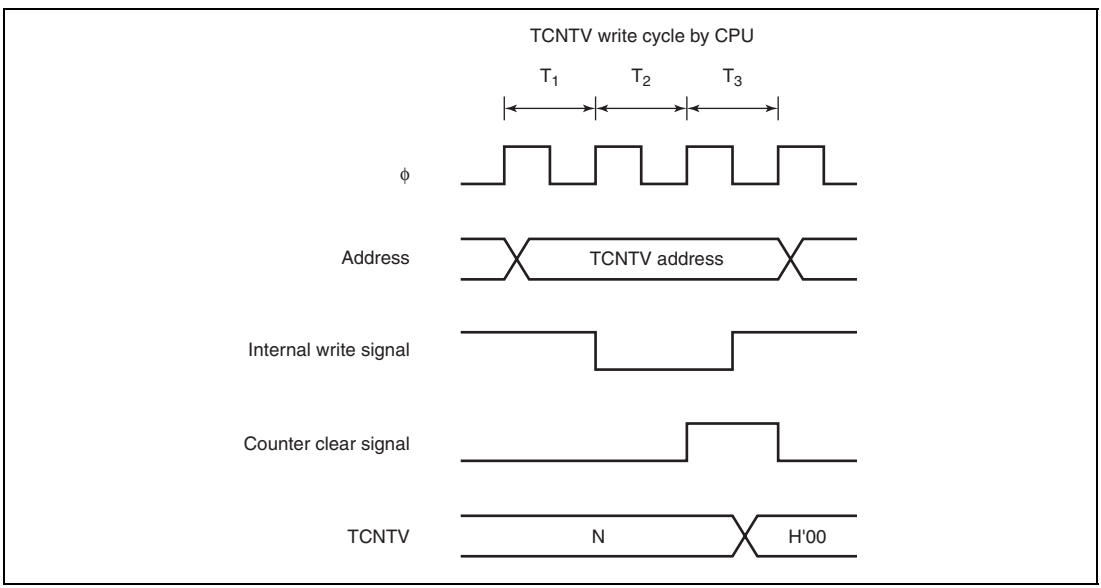



**Figure 10.11 Contention between TCNTV Write and Clear**

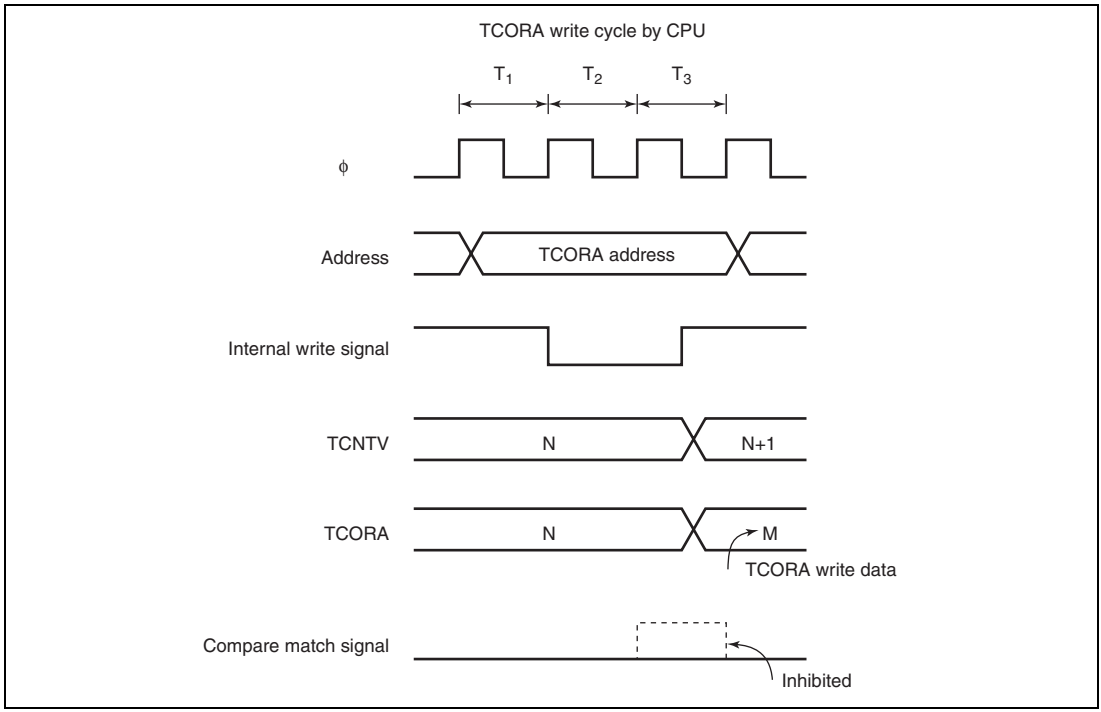



**Figure 10.12 Contention between TCORA Write and Compare Match**

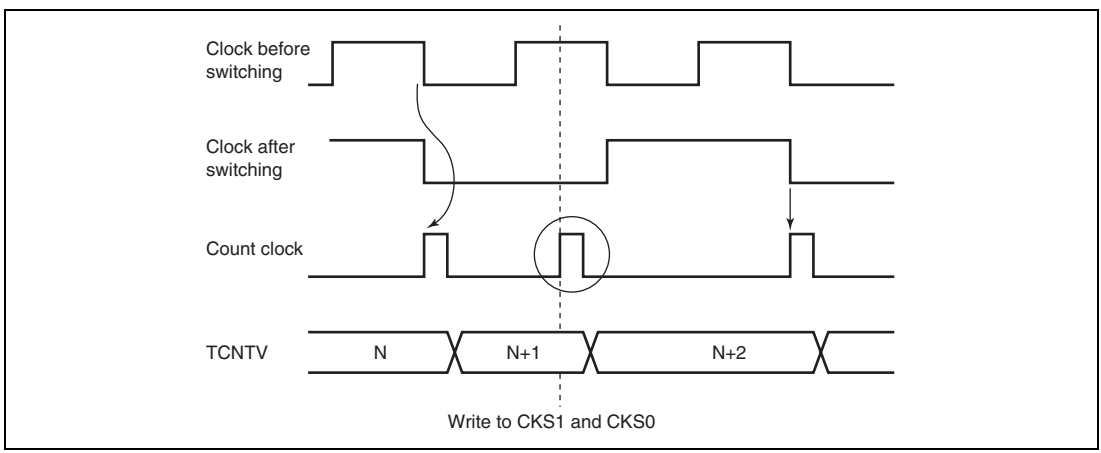



**Figure 10.13 Internal Clock Switching and TCNTV Operation**

# Section 11 Timer W

The timer W has a 16-bit timer having output compare and input capture functions. The timer W can count external events and output pulses with an arbitrary duty cycle by compare match between the timer counter and four general registers. Thus, it can be applied to various systems.

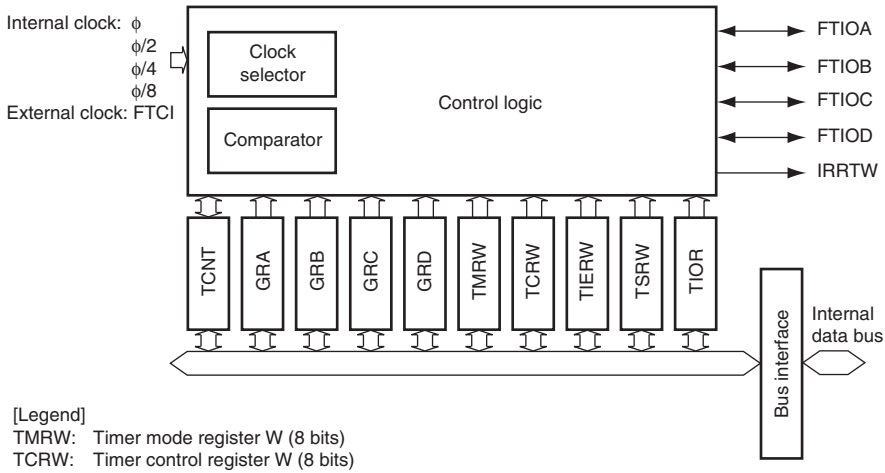
## 11.1 Features

- Selection of five counter clock sources: four internal clocks (ϕ,ϕ/2, ϕ/4, and ϕ/8) and an external clock (external events can be counted)
- Capability to process up to four pulse outputs or four pulse inputs
- Four general registers:
  - — Independently assignable output compare or input capture functions
  - — Usable as two pairs of registers; one register of each pair operates as a buffer for the output compare or input capture register
- Four selectable operating modes:
  - — Waveform output by compare match
    Selection of 0 output, 1 output, or toggle output
  - — Input capture function
    Rising edge, falling edge, or both edges
  - — Counter clearing function
    Counters can be cleared by compare match
  - — PWM mode
    Up to three-phase PWM output can be provided with desired duty ratio.
- Any initial timer output value can be set
- Five interrupt sources
  Four compare match/input capture interrupts and an overflow interrupt.

Table 11.1 summarizes the timer W functions, and figure 11.1 shows a block diagram of the timer W.

**Table 11.1 Timer W Functions**

| Item | | Counter | Input/Output Pins FTIOA | FTIOB | FTIOC | FTIOD |
|---|---|---|---|---|---|---|
| Count clock | | Internal clocks: φ, φ/2, φ/4, φ/8<br>External clock: FTCI | | | | |
| General registers (output compare/input capture registers) | | Period specified in GRA | GRA | GRB | GRC (buffer register for GRA in buffer mode) | GRD (buffer register for GRB in buffer mode) |
| Counter clearing function | | GRA compare match | GRA compare match | — | — | — |
| Initial output value setting function | | — | Yes | Yes | Yes | Yes |
| Buffer function | | — | Yes | Yes | — | — |
| Compare match output | 0 | — | Yes | Yes | Yes | Yes |
| | 1 | — | Yes | Yes | Yes | Yes |
| | Toggle | — | Yes | Yes | Yes | Yes |
| Input capture function | | — | Yes | Yes | Yes | Yes |
| PWM mode | | — | — | Yes | Yes | Yes |
| Interrupt sources | | Overflow | Compare match/input capture | Compare match/input capture | Compare match/input capture | Compare match/input capture |




[Legend]
TMRW: Timer mode register W (8 bits)
TCRW: Timer control register W (8 bits)
TIERW: Timer interrupt enable register W (8 bits)
TSRW: Timer status register W (8 bits)
TIOR: Timer I/O control register (8 bits)
TCNT: Timer counter (16 bits)
GRA: General register A (input capture/output compare register: 16 bits)
GRB: General register B (input capture/output compare register: 16 bits)
GRC: General register C (input capture/output compare register: 16 bits)
GRD: General register D (input capture/output compare register: 16 bits)
IRRTW: Timer W interrupt request

**Figure 11.1 Timer W Block Diagram**

## 11.2 Input/Output Pins

Table 11.2 summarizes the timer W pins.

**Table 11.2 Pin Configuration**

| Name | Abbreviation | Input/Output | Function |
|---|---|---|---|
| External clock input | FTCI | Input | External clock input pin |
| Input capture/output compare A | FTIOA | Input/output | Output pin for GRA output compare or input pin for GRA input capture |
| Input capture/output compare B | FTIOB | Input/output | Output pin for GRB output compare, input pin for GRB input capture, or PWM output pin in PWM mode |
| Input capture/output compare C | FTIOC | Input/output | Output pin for GRC output compare, input pin for GRC input capture, or PWM output pin in PWM mode |
| Input capture/output compare D | FTIOD | Input/output | Output pin for GRD output compare, input pin for GRD input capture, or PWM output pin in PWM mode |

## 11.3 Register Descriptions

The timer W has the following registers.

- Timer mode register W (TMRW)
- Timer control register W (TCRW)
- Timer interrupt enable register W (TIERW)
- Timer status register W (TSRW)
- Timer I/O control register 0 (TIOR0)
- Timer I/O control register 1 (TIOR1)
- Timer counter (TCNT)
- General register A (GRA)
- General register B (GRB)
- General register C (GRC)
- General register D (GRD)

### 11.3.1 Timer Mode Register W (TMRW)

TMRW selects the general register functions and the timer output mode.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CTS | 0 | R/W | Counter Start<br>The counter operation is halted when this bit is 0, while it can be performed when this bit is 1. |
| 6 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 5 | BUFEB | 0 | R/W | Buffer Operation B<br>Selects the GRD function.<br>0: GRD operates as an input capture/output compare register<br>1: GRD operates as the buffer register for GRB |
| 4 | BUFEA | 0 | R/W | Buffer Operation A<br>Selects the GRC function.<br>0: GRC operates as an input capture/output compare register<br>1: GRC operates as the buffer register for GRA |
| 3 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 2 | PWMD | 0 | R/W | PWM Mode D<br>Selects the output mode of the FTIOD pin.<br>0: FTIOD operates normally (output compare output)<br>1: PWM output |
| 1 | PWMC | 0 | R/W | PWM Mode C<br>Selects the output mode of the FTIOC pin.<br>0: FTIOC operates normally (output compare output)<br>1: PWM output |
| 0 | PWMB | 0 | R/W | PWM Mode B<br>Selects the output mode of the FTIOB pin.<br>0: FTIOB operates normally (output compare output)<br>1: PWM output |

### 11.3.2 Timer Control Register W (TCRW)

TCRW selects the timer counter clock source, selects a clearing condition, and specifies the timer output levels.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CCLR | 0 | R/W | Counter Clear<br>The TCNT value is cleared by compare match A when this bit is 1. When it is 0, TCNT operates as a free-running counter. |
| 6 | CKS2 | 0 | R/W | Clock Select 2 to 0<br>Select the TCNT clock source.<br>000: Internal clock: counts on $\phi$<br>001: Internal clock: counts on $\phi/2$<br>010: Internal clock: counts on $\phi/4$<br>011: Internal clock: counts on $\phi/8$<br>1XX: Counts on rising edges of the external event (FTCI)<br>When the internal clock source ($\phi$) is selected, subclock sources are counted in subactive and subsleep modes. |
| 5 | CKS1 | 0 | R/W | |
| 4 | CKS0 | 0 | R/W | |
| 3 | TOD | 0 | R/W | Timer Output Level Setting D<br>0: Output value is 0*<br>1: Output value is 1* |
| 2 | TOC | 0 | R/W | Timer Output Level Setting C<br>0: Output value is 0*<br>1: Output value is 1* |
| 1 | TOB | 0 | R/W | Timer Output Level Setting B<br>0: Output value is 0*<br>1: Output value is 1* |
| 0 | TOA | 0 | R/W | Timer Output Level Setting A<br>0: Output value is 0*<br>1: Output value is 1* |

Legend: X: Don't care.

Note: * The change of the setting is immediately reflected in the output value.

### 11.3.3 Timer Interrupt Enable Register W (TIERW)

TIERW controls the timer W interrupt request.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | OVIE | 0 | R/W | Timer Overflow Interrupt Enable<br>When this bit is set to 1, FOVI interrupt requested by OVF flag in TSRW is enabled. |
| 6 | — | 1 | — | Reserved |
| 5 | — | 1 | — | These bits are always read as 1. |
| 4 | — | 1 | — | |
| 3 | IMIED | 0 | R/W | Input Capture/Compare Match Interrupt Enable D<br>When this bit is set to 1, IMID interrupt requested by IMFD flag in TSRW is enabled. |
| 2 | IMIEC | 0 | R/W | Input Capture/Compare Match Interrupt Enable C<br>When this bit is set to 1, IMIC interrupt requested by IMFC flag in TSRW is enabled. |
| 1 | IMIEB | 0 | R/W | Input Capture/Compare Match Interrupt Enable B<br>When this bit is set to 1, IMIB interrupt requested by IMFB flag in TSRW is enabled. |
| 0 | IMIEA | 0 | R/W | Input Capture/Compare Match Interrupt Enable A<br>When this bit is set to 1, IMIA interrupt requested by IMFA flag in TSRW is enabled. |

### 11.3.4 Timer Status Register W (TSRW)

TSRW shows the status of interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | OVF | 0 | R/W | Timer Overflow Flag<br>[Setting condition]<br>When TCNT overflows from H'FFFF to H'0000<br>[Clearing condition]<br>Read OVF when OVF = 1, then write 0 in OVF |
| 6 | — | 1 | — | Reserved |
| 5 | — | 1 | — | These bits are always read as 1. |
| 4 | — | 1 | — | |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | IMFD | 0 | R/W | Input Capture/Compare Match Flag D<br>[Setting conditions]<br>• TCNT = GRD when GRD functions as an output compare register<br>• The TCNT value is transferred to GRD by an input capture signal when GRD functions as an input capture register<br>[Clearing condition]<br>Read IMFD when IMFD = 1, then write 0 in IMFD |
| 2 | IMFC | 0 | R/W | Input Capture/Compare Match Flag C<br>[Setting conditions]<br>• TCNT = GRC when GRC functions as an output compare register<br>• The TCNT value is transferred to GRC by an input capture signal when GRC functions as an input capture register<br>[Clearing condition]<br>Read IMFC when IMFC = 1, then write 0 in IMFC |
| 1 | IMFB | 0 | R/W | Input Capture/Compare Match Flag B<br>[Setting conditions]<br>• TCNT = GRB when GRB functions as an output compare register<br>• The TCNT value is transferred to GRB by an input capture signal when GRB functions as an input capture register<br>[Clearing condition]<br>Read IMFB when IMFB = 1, then write 0 in IMFB |
| 0 | IMFA | 0 | R/W | Input Capture/Compare Match Flag A<br>[Setting conditions]<br>• TCNT = GRA when GRA functions as an output compare register<br>• The TCNT value is transferred to GRA by an input capture signal when GRA functions as an input capture register<br>[Clearing condition]<br>Read IMFA when IMFA = 1, then write 0 in IMFA |

### 11.3.5 Timer I/O Control Register 0 (TIOR0)

TIOR0 selects the functions of GRA and GRB, and specifies the functions of the FTIOA and FTIOB pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 6 | IOB2 | 0 | R/W | I/O Control B2<br>Selects the GRB function.<br>0: GRB functions as an output compare register<br>1: GRB functions as an input capture register |
| 5 | IOB1 | 0 | R/W | I/O Control B1 and B0 |
| 4 | IOB0 | 0 | R/W | When IOB2 = 0,<br>00: No output at compare match<br>01: 0 output to the FTIOB pin at GRB compare match<br>10: 1 output to the FTIOB pin at GRB compare match<br>11: Output toggles to the FTIOB pin at GRB compare match<br>When IOB2 = 1,<br>00: Input capture at rising edge at the FTIOB pin<br>01: Input capture at falling edge at the FTIOB pin<br>1X: Input capture at rising and falling edges of the FTIOB pin |
| 3 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 2 | IOA2 | 0 | R/W | I/O Control A2<br>Selects the GRA function.<br>0: GRA functions as an output compare register<br>1: GRA functions as an input capture register |
| 1 | IOA1 | 0 | R/W | I/O Control A1 and A0 |
| 0 | IOA0 | 0 | R/W | When IOA2 = 0,<br>00: No output at compare match<br>01: 0 output to the FTIOA pin at GRA compare match<br>10: 1 output to the FTIOA pin at GRA compare match<br>11: Output toggles to the FTIOA pin at GRA compare match<br>When IOA2 = 1,<br>00: Input capture at rising edge of the FTIOA pin<br>01: Input capture at falling edge of the FTIOA pin<br>1X: Input capture at rising and falling edges of the FTIOA pin |

Legend: X: Don't care.

### 11.3.6 Timer I/O Control Register 1 (TIOR1)

TIOR1 selects the functions of GRC and GRD, and specifies the functions of the FTIOC and FTIOD pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 6 | IOD2 | 0 | R/W | I/O Control D2<br>Selects the GRD function.<br>0: GRD functions as an output compare register<br>1: GRD functions as an input capture register |
| 5 | IOD1 | 0 | R/W | I/O Control D1 and D0 |
| 4 | IOD0 | 0 | R/W | When IOD2 = 0,<br>00: No output at compare match<br>01: 0 output to the FTIOD pin at GRD compare match<br>10: 1 output to the FTIOD pin at GRD compare match<br>11: Output toggles to the FTIOD pin at GRD compare match<br>When IOD2 = 1,<br>00: Input capture at rising edge at the FTIOD pin<br>01: Input capture at falling edge at the FTIOD pin<br>1X: Input capture at rising and falling edges at the FTIOD pin |
| 3 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 2 | IOC2 | 0 | R/W | I/O Control C2<br>Selects the GRC function.<br>0: GRC functions as an output compare register<br>1: GRC functions as an input capture register |
| 1 | IOC1 | 0 | R/W | I/O Control C1 and C0 |
| 0 | IOC0 | 0 | R/W | When IOC2 = 0,<br>00: No output at compare match<br>01: 0 output to the FTIOC pin at GRC compare match<br>10: 1 output to the FTIOC pin at GRC compare match<br>11: Output toggles to the FTIOC pin at GRC compare match<br>When IOC2 = 1,<br>00: Input capture to GRC at rising edge of the FTIOC pin<br>01: Input capture to GRC at falling edge of the FTIOC pin<br>1X: Input capture to GRC at rising and falling edges of the FTIOC pin |

Legend: X: Don't care.

### 11.3.7 Timer Counter (TCNT)

TCNT is a 16-bit readable/writable up-counter. The clock source is selected by bits CKS2 to CKS0 in TCRW. TCNT can be cleared to H'0000 through a compare match with GRA by setting the CCLR in TCRW to 1. When TCNT overflows (changes from H'FFFF to H'0000), the OVF flag in TSRW is set to 1. If OVIE in TIERW is set to 1 at this time, an interrupt request is generated. TCNT must always be read or written in 16-bit units; 8-bit access is not allowed. TCNT is initialized to H'0000 by a reset.

### 11.3.8 General Registers A to D (GRA to GRD)

Each general register is a 16-bit readable/writable register that can function as either an output-compare register or an input-capture register. The function is selected by settings in TIOR0 and TIOR1.

When a general register is used as an input-compare register, its value is constantly compared with the TCNT value. When the two values match (a compare match), the corresponding flag (IMFA, IMFB, IMFC, or IMFD) in TSRW is set to 1. An interrupt request is generated at this time, when IMIEA, IMIEB, IMIEC, or IMIED is set to 1. Compare match output can be selected in TIOR.

When a general register is used as an input-capture register, an external input-capture signal is detected and the current TCNT value is stored in the general register. The corresponding flag (IMFA, IMFB, IMFC, or IMFD) in TSRW is set to 1. If the corresponding interrupt-enable bit (IMIEA, IMIEB, IMIEC, or IMIED) in TSRW is set to 1 at this time, an interrupt request is generated. The edge of the input-capture signal is selected in TIOR.

GRC and GRD can be used as buffer registers of GRA and GRB, respectively, by setting BUFEA and BUFEB in TMRW.

For example, when GRA is set as an output-compare register and GRC is set as the buffer register for GRA, the value in the buffer register GRC is sent to GRA whenever compare match A is generated.

When GRA is set as an input-capture register and GRC is set as the buffer register for GRA, the value in TCNT is transferred to GRA and the value in the buffer register GRC is transferred to GRA whenever an input capture is generated.

GRA to GRD must be written or read in 16-bit units; 8-bit access is not allowed. GRA to GRD are initialized to H'FFFF by a reset.

## 11.4 Operation

The timer W has the following operating modes.

- Normal Operation
- PWM Operation

### 11.4.1 Normal Operation

TCNT performs free-running or periodic counting operations. After a reset, TCNT is set as a free-running counter. When the CTS bit in TMRW is set to 1, TCNT starts incrementing the count. When the count overflows from H'FFFF to H'0000, the OVF flag in TSRW is set to 1. If the OVIE in TIERW is set to 1, an interrupt request is generated. Figure 11.2 shows free-running counting.

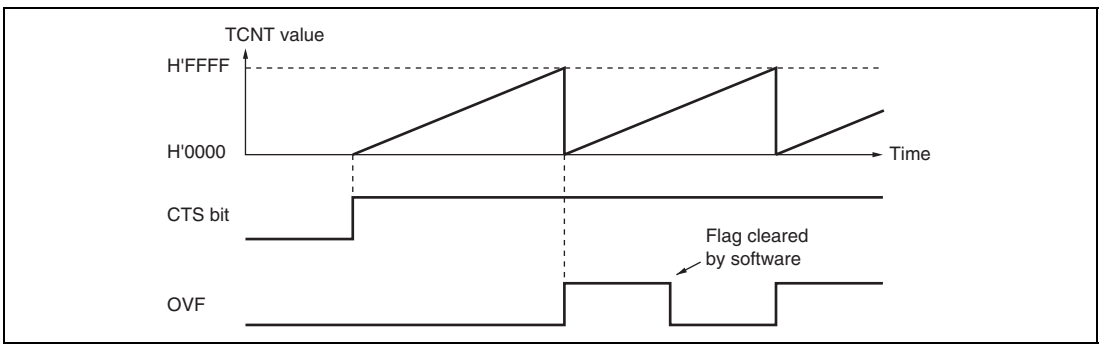



**Figure 11.2 Free-Running Counter Operation**

Periodic counting operation can be performed when GRA is set as an output compare register and bit CCLR in TCRW is set to 1. When the count matches GRA, TCNT is cleared to H'0000, the IMFA flag in TSRW is set to 1. If the corresponding IMIEA bit in TIERW is set to 1, an interrupt request is generated. TCNT continues counting from H'0000. Figure 11.3 shows periodic counting.

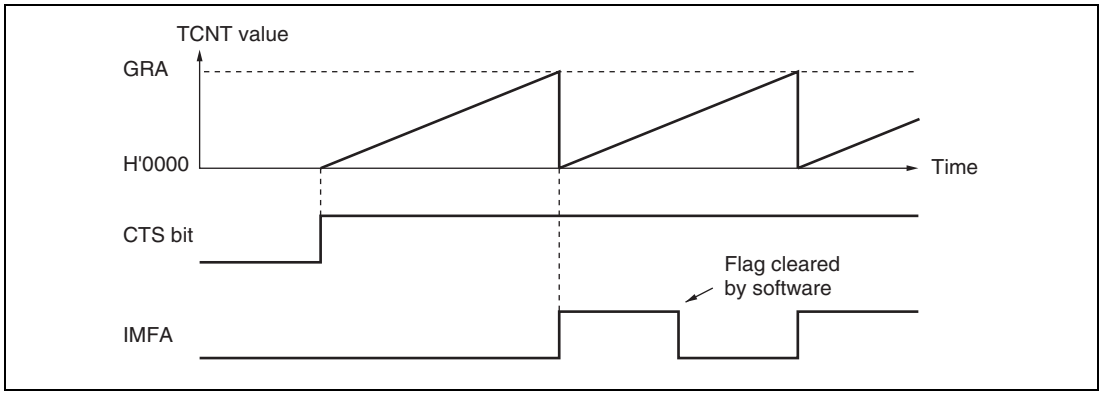



**Figure 11.3 Periodic Counter Operation**

By setting a general register as an output compare register, compare match A, B, C, or D can cause the output at the FTIOA, FTIOB, FTIOC, or FTIOD pin to output 0, output 1, or toggle. Figure 11.4 shows an example of 0 and 1 output when TCNT operates as a free-running counter, 1 output is selected for compare match A, and 0 output is selected for compare match B. When signal is already at the selected output level, the signal level does not change at compare match.

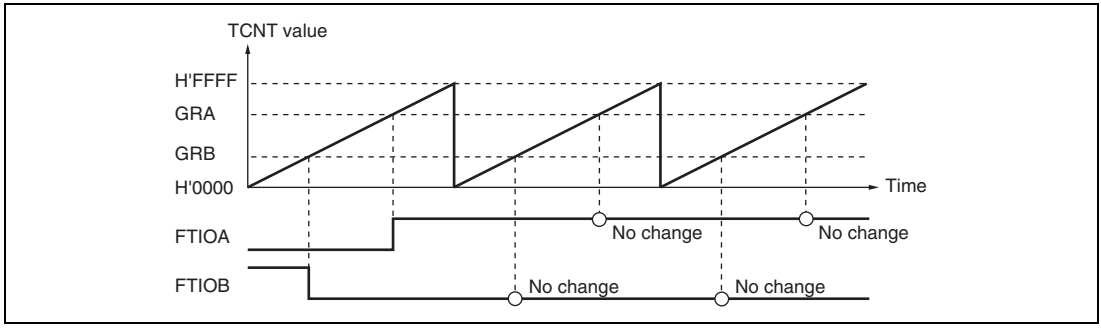



**Figure 11.4 0 and 1 Output Example (TOA = 0, TOB = 1)**

Figure 11.5 shows an example of toggle output when TCNT operates as a free-running counter, and toggle output is selected for both compare match A and B.

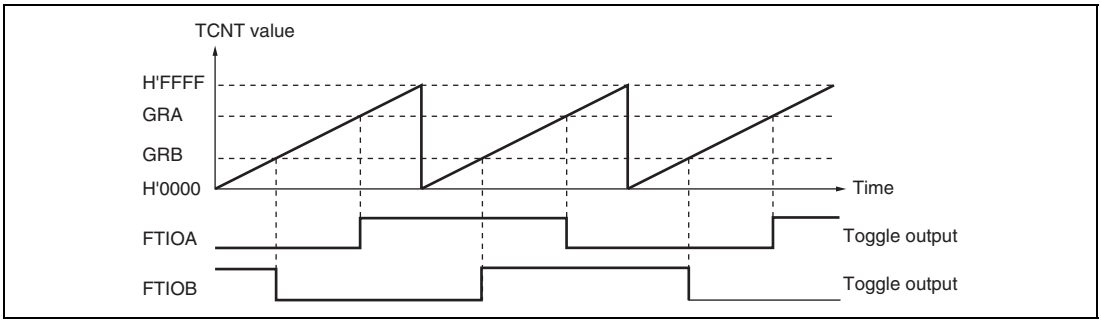



**Figure 11.5 Toggle Output Example (TOA = 0, TOB = 1)**

Figure 11.6 shows another example of toggle output when TCNT operates as a periodic counter, cleared by compare match A. Toggle output is selected for both compare match A and B.

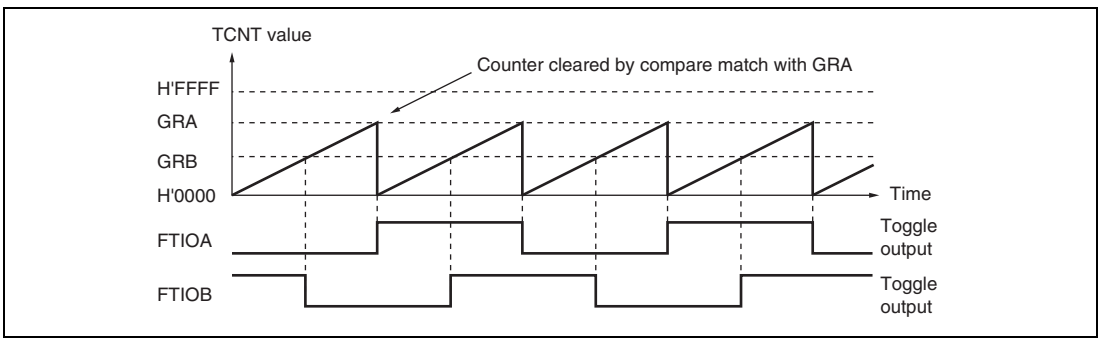



**Figure 11.6 Toggle Output Example (TOA = 0, TOB = 1)**

The TCNT value can be captured into a general register (GRA, GRB, GRC, or GRD) when a signal level changes at an input-capture pin (FTIOA, FTIOB, FTIOC, or FTIOD). Capture can take place on the rising edge, falling edge, or both edges. By using the input-capture function, the pulse width and periods can be measured. Figure 11.7 shows an example of input capture when both edges of FTIOA and the falling edge of FTIOB are selected as capture edges. TCNT operates as a free-running counter.

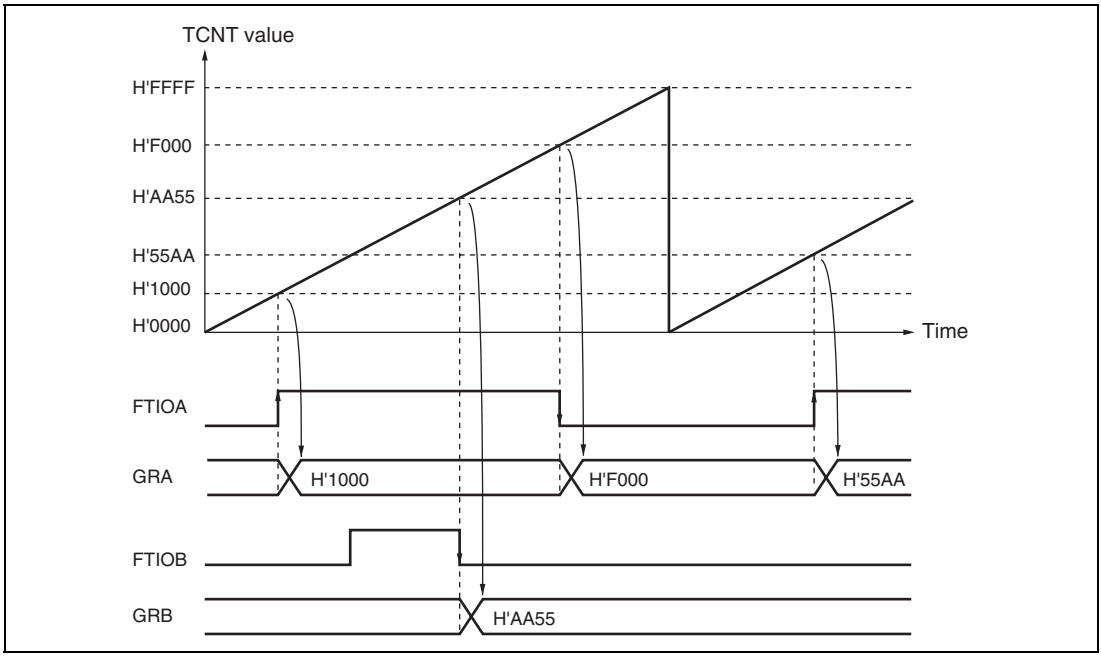



**Figure 11.7 Input Capture Operating Example**

Figure 11.8 shows an example of buffer operation when the GRA is set as an input-capture register and GRC is set as the buffer register for GRA. TCNT operates as a free-running counter, and FTIOA captures both rising and falling edge of the input signal. Due to the buffer operation, the GRA value is transferred to GRC by input-capture A and the TCNT value is stored in GRA.

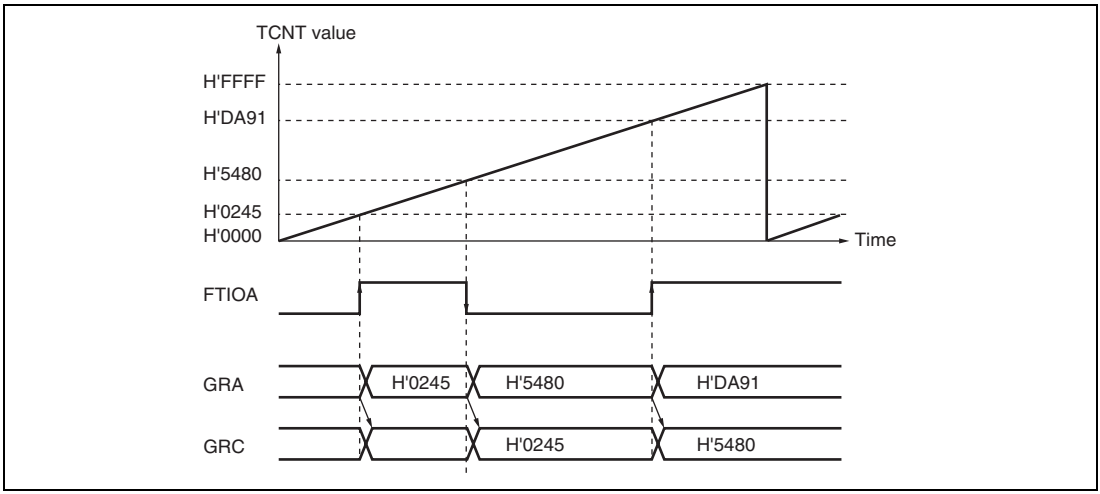



**Figure 11.8 Buffer Operation Example (Input Capture)**

### 11.4.2 PWM Operation

In PWM mode, PWM waveforms are generated by using GRA as the period register and GRB, GRC, and GRD as duty registers. PWM waveforms are output from the FTIOB, FTIOC, and FTIOD pins. Up to three-phase PWM waveforms can be output. In PWM mode, a general register functions as an output compare register automatically. The output level of each pin depends on the corresponding timer output level set bit (TOB, TOC, and TOD) in TCRW. When TOB is 1, the FTIOB output goes to 1 at compare match A and to 0 at compare match B. When TOB is 0, the FTIOB output goes to 0 at compare match A and to 1 at compare match B. Thus the compare match output level settings in TIOR0 and TIOR1 are ignored for the output pin set to PWM mode. If the same value is set in the cycle register and the duty register, the output does not change when a compare match occurs.

Figure 11.9 shows an example of operation in PWM mode. The output signals go to 1 and TCNT is cleared at compare match A, and the output signals go to 0 at compare match B, C, and D (TOB, TOC, and TOD = 1: initial output values are set to 1).

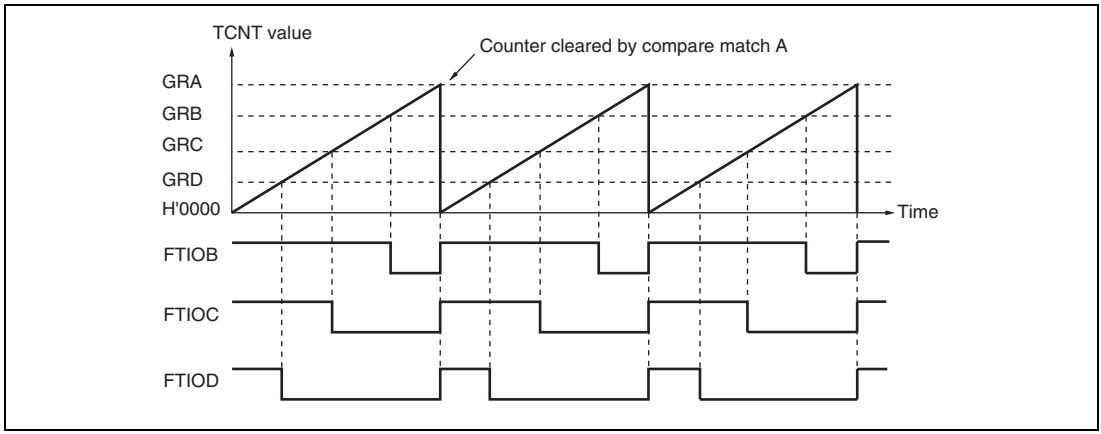



**Figure 11.9 PWM Mode Example (1)**

Figure 11.10 shows another example of operation in PWM mode. The output signals go to 0 and TCNT is cleared at compare match A, and the output signals go to 1 at compare match B, C, and D (TOB, TOC, and TOD = 0: initial output values are set to 1).

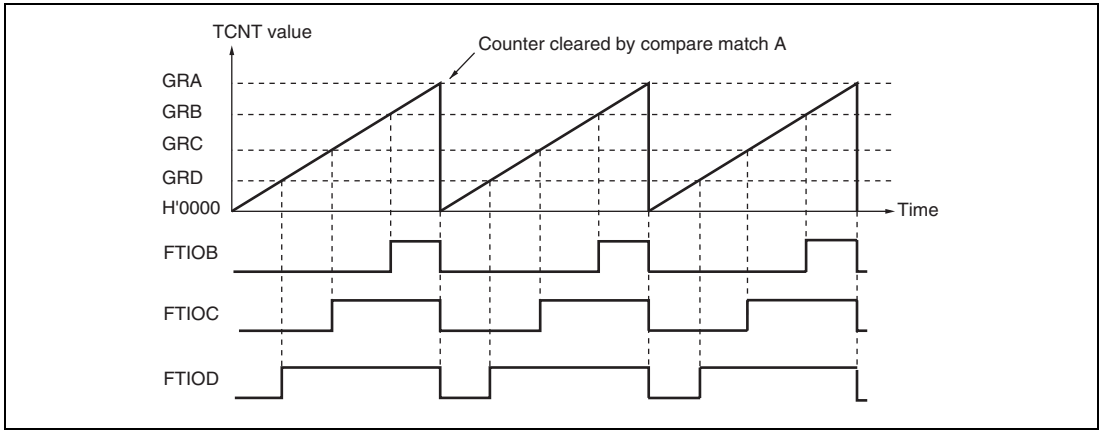



**Figure 11.10 PWM Mode Example (2)**

Figure 11.11 shows an example of buffer operation when the FTIOB pin is set to PWM mode and GRD is set as the buffer register for GRB. TCNT is cleared by compare match A, and FTIOB outputs 1 at compare match B and 0 at compare match A.

Due to the buffer operation, the FTIOB output level changes and the value of buffer register GRD is transferred to GRB whenever compare match B occurs. This procedure is repeated every time compare match B occurs.

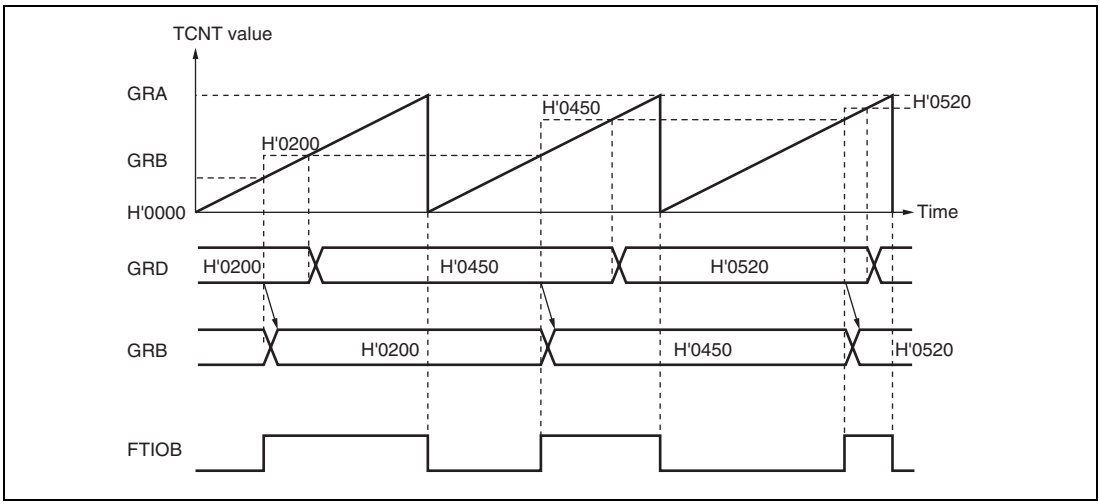



**Figure 11.11 Buffer Operation Example (Output Compare)**

Figures 11.12 and 11.13 show examples of the output of PWM waveforms with duty cycles of 0% and 100%.

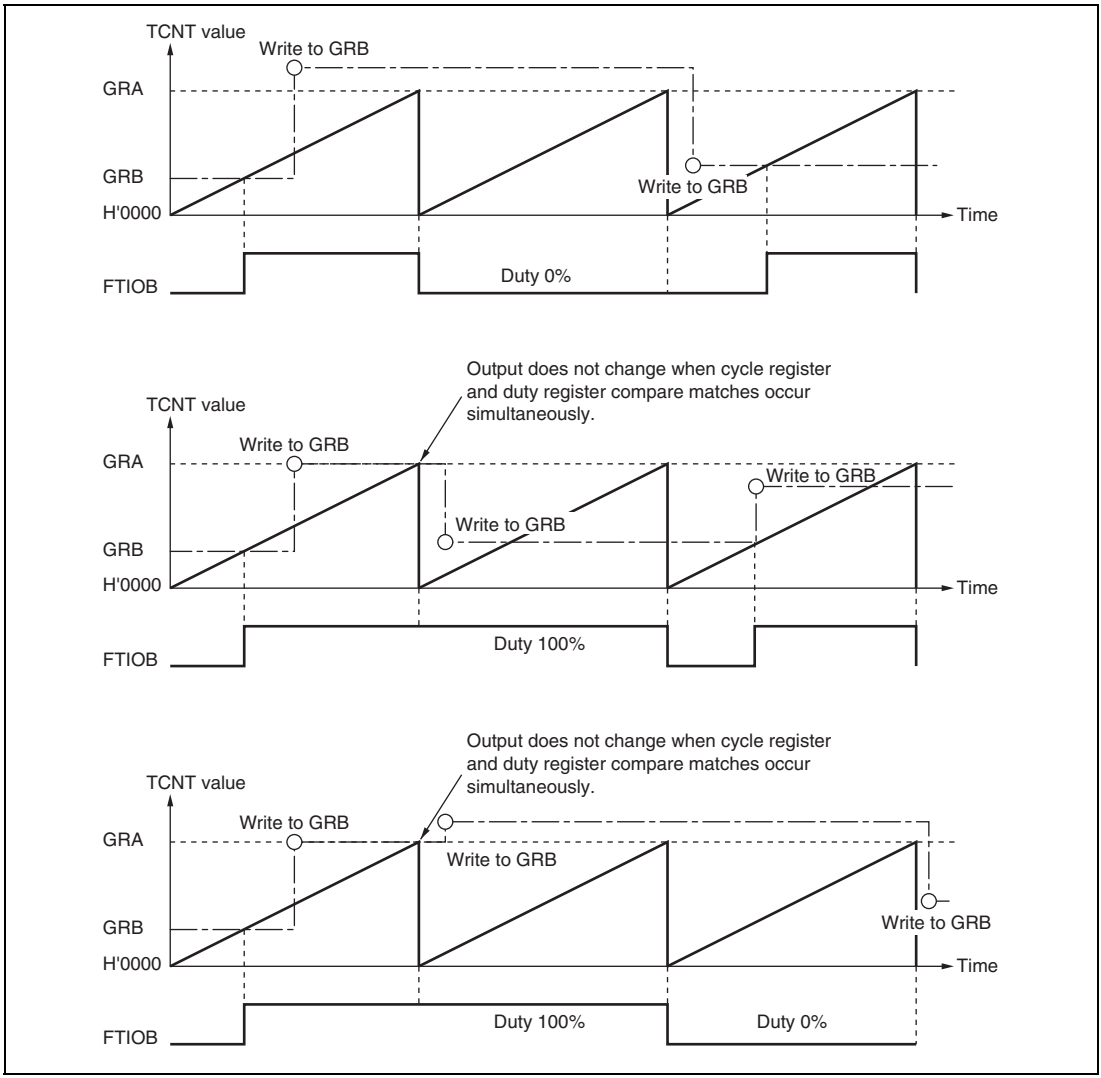



**Figure 11.12 PWM Mode Example**
**(TOB, TOC, and TOD = 0: initial output values are set to 0)**

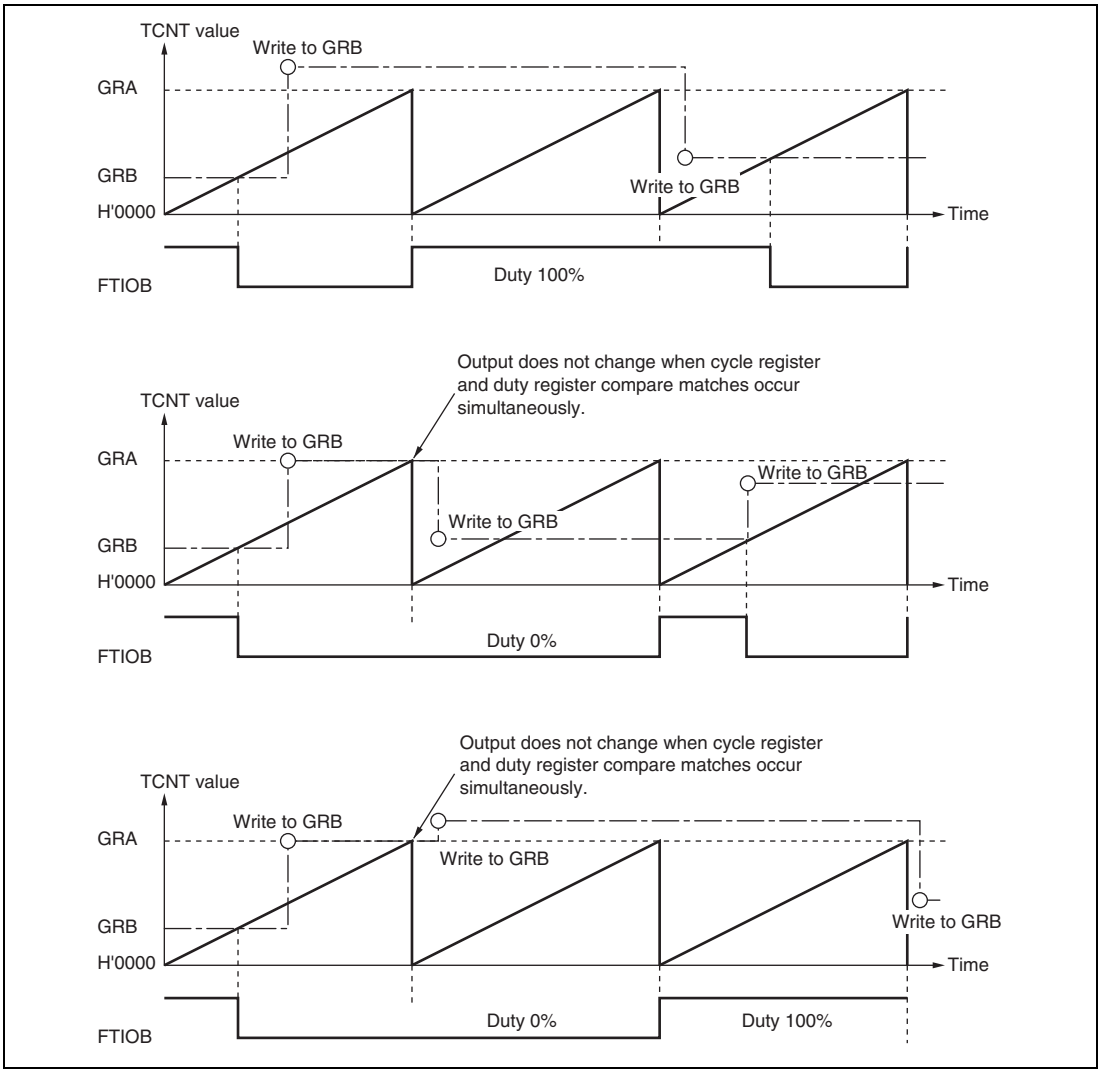



**Figure 11.13 PWM Mode Example**
**(TOB, TOC, and TOD = 1: initial output values are set to 1)**

## 11.5 Operation Timing

### 11.5.1 TCNT Count Timing

Figure 11.14 shows the TCNT count timing when the internal clock source is selected. Figure 11.15 shows the timing when the external clock source is selected. The pulse width of the external clock signal must be at least two system clock (ϕ) cycles; shorter pulses will not be counted correctly.

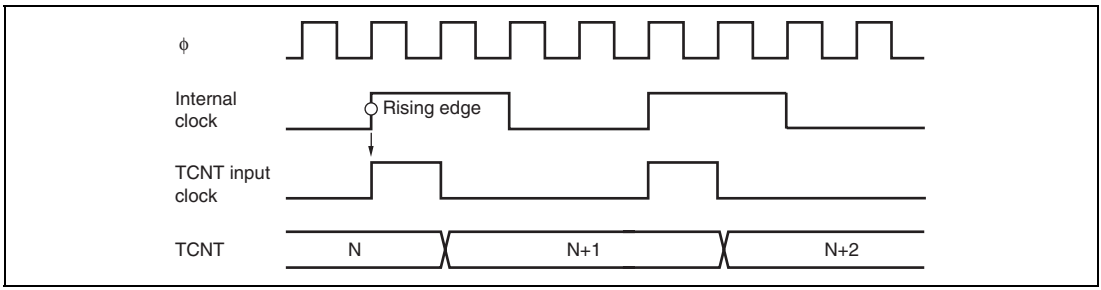



**Figure 11.14 Count Timing for Internal Clock Source**

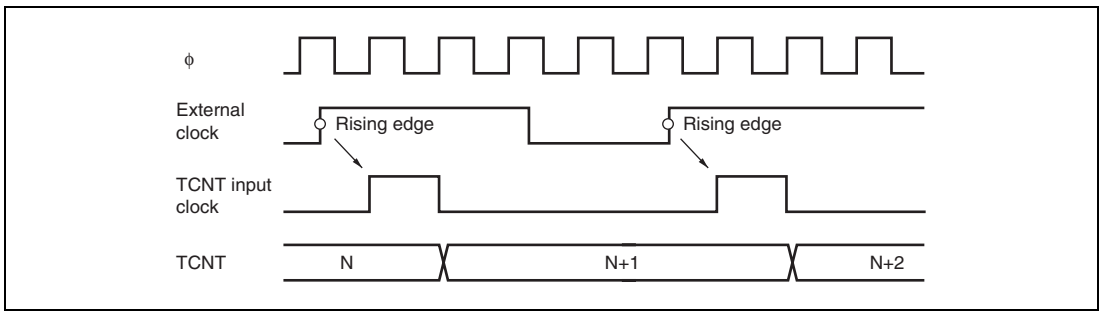



**Figure 11.15 Count Timing for External Clock Source**

### 11.5.2 Output Compare Output Timing

The compare match signal is generated in the last state in which TCNT and GR match (when TCNT changes from the matching value to the next value). When the compare match signal is generated, the output value selected in TIOR is output at the compare match output pin (FTIOA, FTIOB, FTIOC, or FTIOD).

When TCNT matches GR, the compare match signal is generated only after the next counter clock pulse is input.

Figure 11.16 shows the output compare timing.

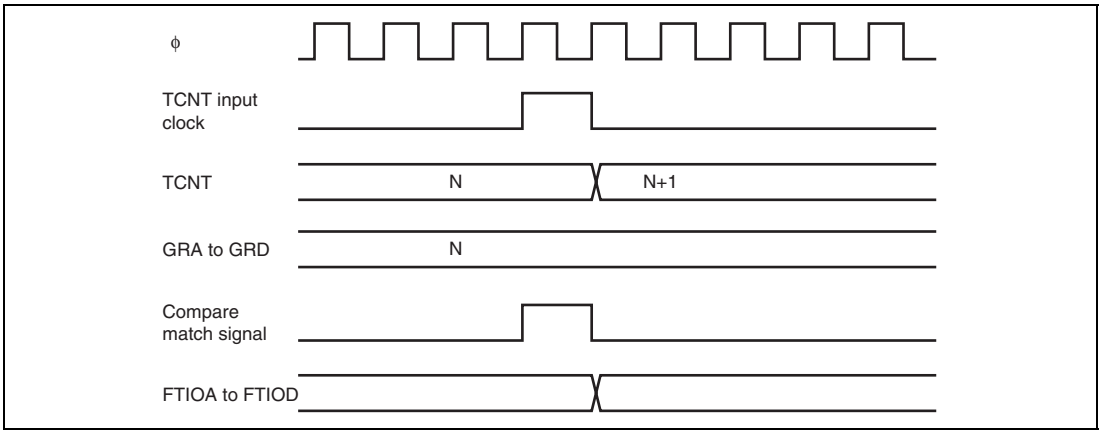



**Figure 11.16 Output Compare Output Timing**

### 11.5.3 Input Capture Timing

Input capture on the rising edge, falling edge, or both edges can be selected through settings in TIOR0 and TIOR1. Figure 11.17 shows the timing when the falling edge is selected. The pulse width of the input capture signal must be at least two system clock (ϕ) cycles; shorter pulses will not be detected correctly.

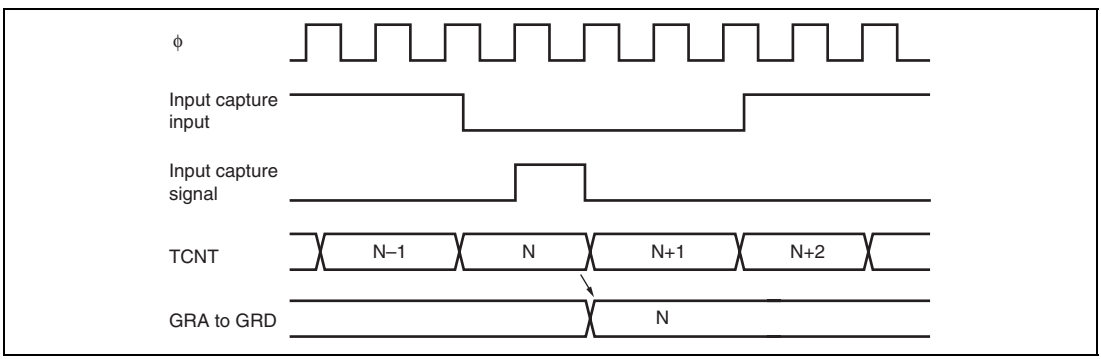



**Figure 11.17 Input Capture Input Signal Timing**

### 11.5.4 Timing of Counter Clearing by Compare Match

Figure 11.18 shows the timing when the counter is cleared by compare match A. When the GRA value is N, the counter counts from 0 to N, and its cycle is N + 1.

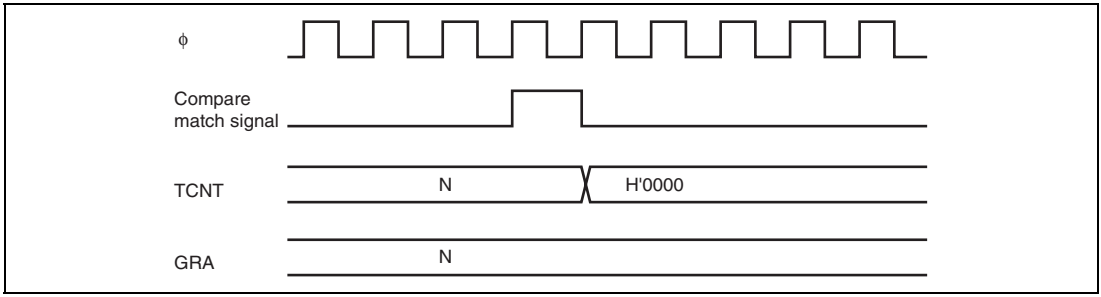



**Figure 11.18 Timing of Counter Clearing by Compare Match**

### 11.5.5 Buffer Operation Timing

Figures 11.19 and 11.20 show the buffer operation timing.

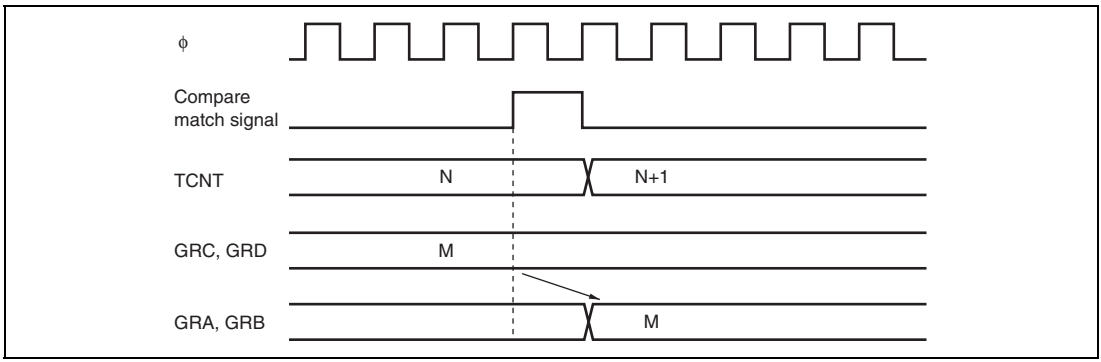



**Figure 11.19 Buffer Operation Timing (Compare Match)**

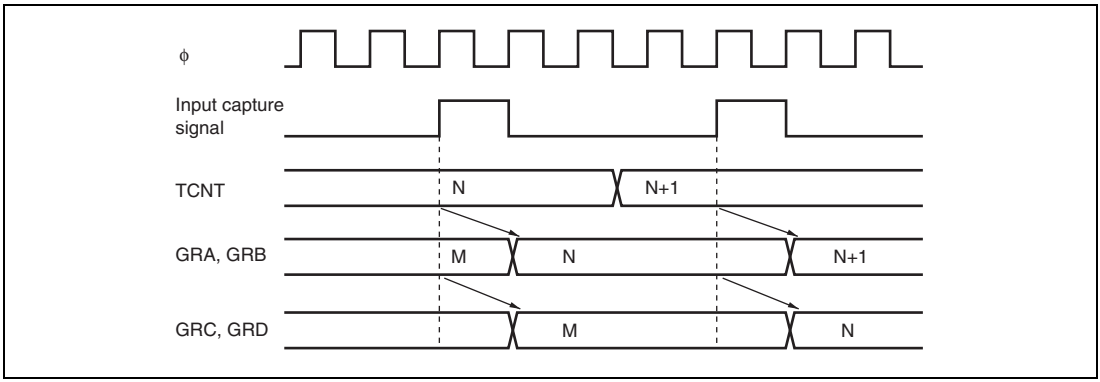



**Figure 11.20 Buffer Operation Timing (Input Capture)**

### 11.5.6 Timing of IMFA to IMFD Flag Setting at Compare Match

If a general register (GRA, GRB, GRC, or GRD) is used as an output compare register, the corresponding IMFA, IMFB, IMFC, or IMFD flag is set to 1 when TCNT matches the general register.

The compare match signal is generated in the last state in which the values match (when TCNT is updated from the matching count to the next count). Therefore, when TCNT matches a general register, the compare match signal is generated only after the next TCNT clock pulse is input.

Figure 11.21 shows the timing of the IMFA to IMFD flag setting at compare match.

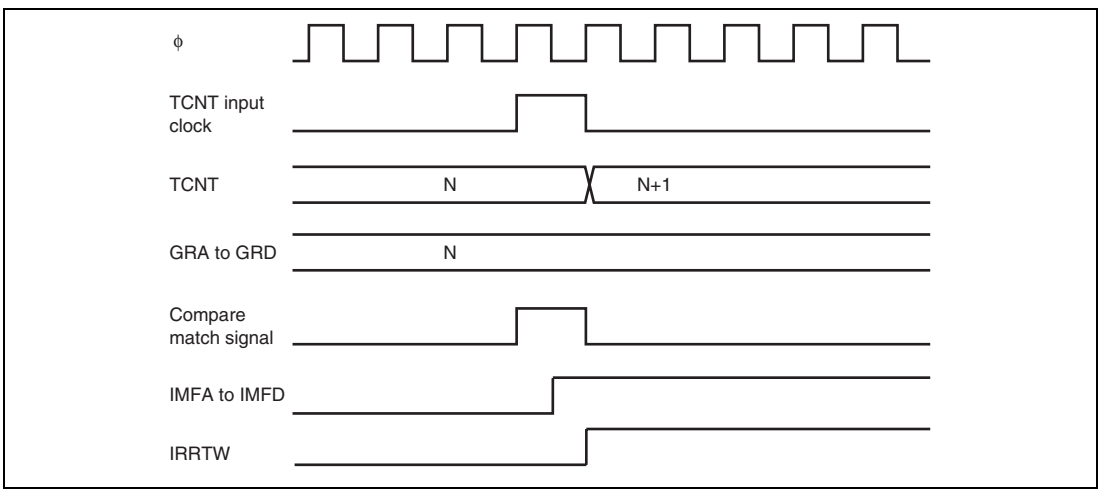



**Figure 11.21 Timing of IMFA to IMFD Flag Setting at Compare Match**

### 11.5.7 Timing of IMFA to IMFD Setting at Input Capture

If a general register (GRA, GRB, GRC, or GRD) is used as an input capture register, the corresponding IMFA, IMFB, IMFC, or IMFD flag is set to 1 when an input capture occurs. Figure 11.22 shows the timing of the IMFA to IMFD flag setting at input capture.

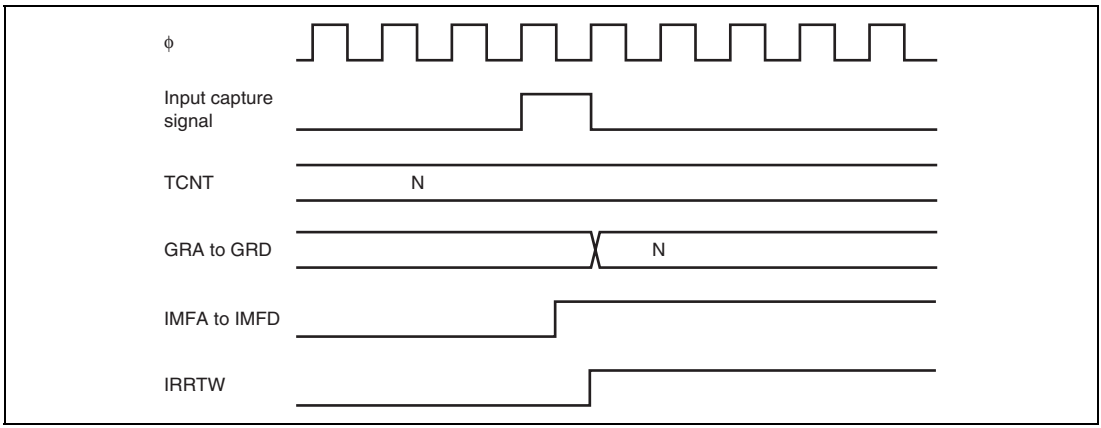



**Figure 11.22 Timing of IMFA to IMFD Flag Setting at Input Capture**

### 11.5.8 Timing of Status Flag Clearing

When the CPU reads a status flag while it is set to 1, then writes 0 in the status flag, the status flag is cleared. Figure 11.23 shows the status flag clearing timing.

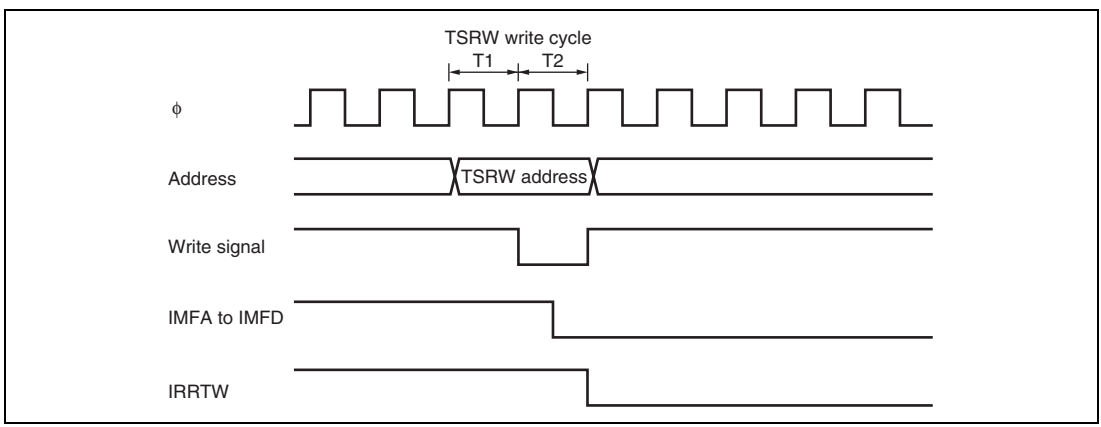



**Figure 11.23 Timing of Status Flag Clearing by CPU**

## 11.6 Usage Notes

The following types of contention or operation can occur in timer W operation.

1. The pulse width of the input clock signal and the input capture signal must be at least two system clock (ϕ) cycles; shorter pulses will not be detected correctly.
2. Writing to registers is performed in the T2 state of a TCNT write cycle.

   If counter clear signal occurs in the T2 state of a TCNT write cycle, clearing of the counter takes priority and the write is not performed, as shown in figure 11.24. If counting-up is generated in the TCNT write cycle to contend with the TCNT counting-up, writing takes precedence.
3. Depending on the timing, TCNT may be incremented by a switch between different internal clock sources. When TCNT is internally clocked, an increment pulse is generated from the rising edge of an internal clock signal, that is divided system clock (ϕ). Therefore, as shown in figure 11.25 the switch is from a low clock signal to a high clock signal, the switchover is seen as a rising edge, causing TCNT to increment.
4. If timer W enters module standby mode while an interrupt request is generated, the interrupt request cannot be cleared. Before entering module standby mode, disable interrupt requests.

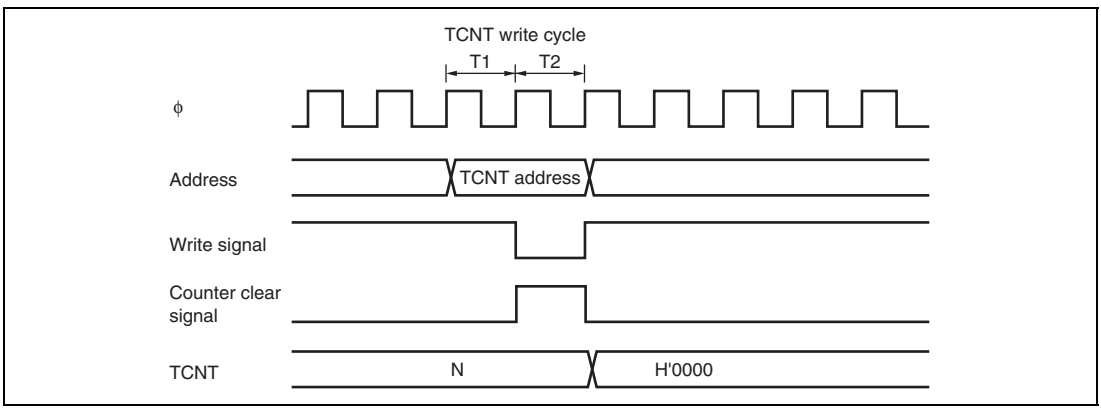



**Figure 11.24 Contention between TCNT Write and Clear**

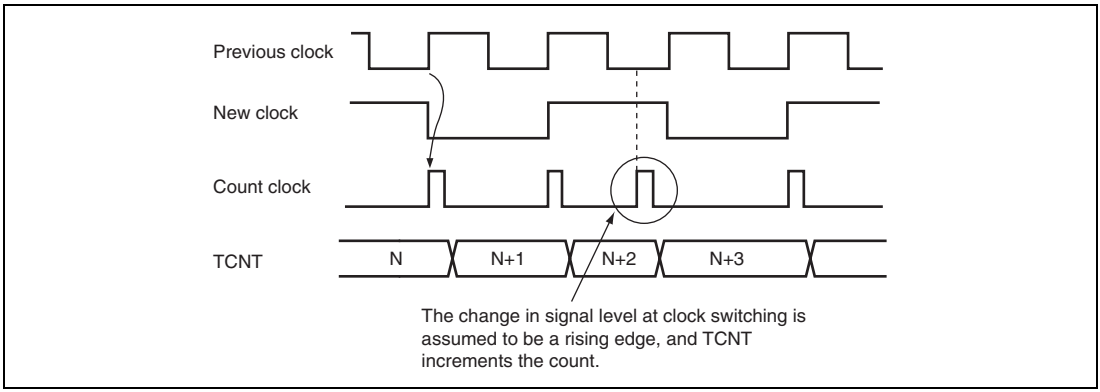



**Figure 11.25 Internal Clock Switching and TCNT Operation**

5. The TOA to TOD bits in TCRW decide the value of the FTIO pin, which is output until the first compare match occurs. Once a compare match occurs and this compare match changes the values of FTIOA to FTIOD output, the values of the FTIOA to FTIOD pin output and the values read from the TOA to TOD bits may differ. Moreover, when the writing to TCRW and the generation of the compare match A to D occur at the same timing, the writing to TCRW has the priority. Thus, output change due to the compare match is not reflected to the FTIOA to FTIOD pins. Therefore, when bit manipulation instruction is used to write to TCRW, the values of the FTIOA to FTIOD pin output may result in an unexpected result. When TCRW is to be written to while compare match is operating, stop the counter once before accessing to TCRW, read the port 8 state to reflect the values of FTIOA to FTIOD output, to TOA to TOD, and then restart the counter. Figure 11.26 shows an example when the compare match and the bit manipulation instruction to TCRW occur at the same timing.

TCRW has been set to H'06. Compare match B and compare match C are used. The FTIOB pin is in the 1 output state, and is set to the toggle output or the 0 output by compare match B.
When BCLR#2, @TCRW is executed to clear the TOC bit (the FTIOC signal is low) and compare match B occurs at the same timing as shown below, the H'02 writing to TCRW has priority and compare match B does not drive the FTIOB signal low; the FTIOB signal remains high.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TCRW | CCLR | CKS2 | CKS1 | CKS0 | TOD | TOC | TOB | TOA |
| Set value | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

BCLR#2, @TCRW
(1) TCRW read operation: Read H'06
(2) Modify operation: Modify H'06 to H'02
(3) Write operation to TCRW: Write H'02

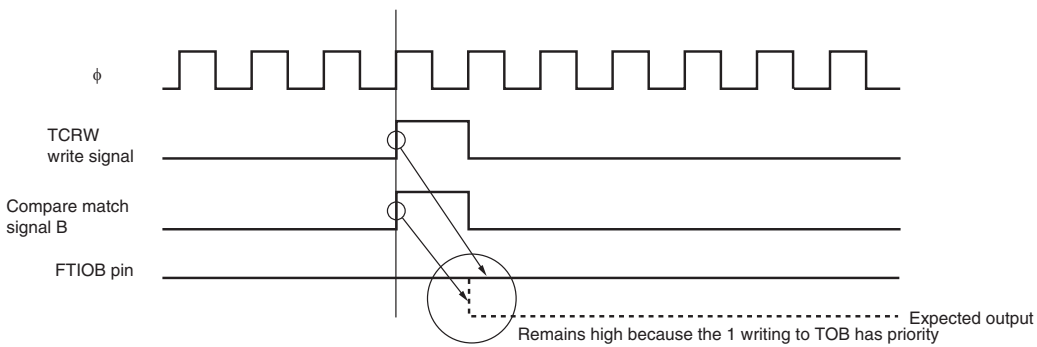



**Figure 11.26 When Compare Match and Bit Manipulation Instruction to TCRW Occur at the Same Timing**

# Section 12 Watchdog Timer

The watchdog timer is an 8-bit timer that can generate an internal reset signal for this LSI if a system crash prevents the CPU from writing to the timer counter, thus allowing it to overflow.

The block diagram of the watchdog timer is shown in figure 12.1.

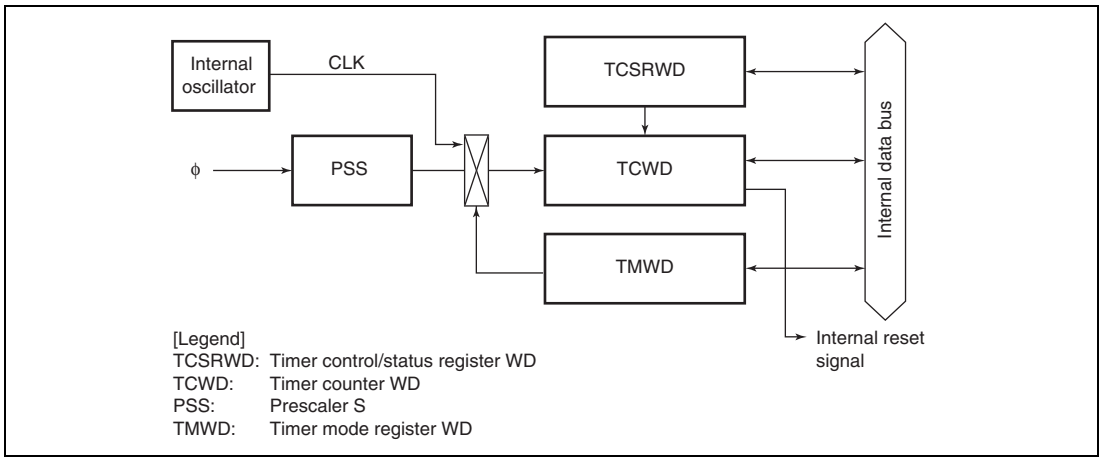



**Figure 12.1 Block Diagram of Watchdog Timer**

## 12.1 Features

- Selectable from nine counter input clocks.
  Eight clock sources (ϕ/64, ϕ/128, ϕ/256, ϕ/512, ϕ/1024, ϕ/2048, ϕ/4096, and ϕ/8192) or the internal oscillator can be selected as the timer-counter clock. When the internal oscillator is selected, it can operate as the watchdog timer in any operating mode.
- Reset signal generated on counter overflow
  An overflow period of 1 to 256 times the selected clock can be set.

## 12.2 Register Descriptions

The watchdog timer has the following registers.

- Timer control/status register WD (TCSRWD)
- Timer counter WD (TCWD)
- Timer mode register WD (TMWD)

### 12.2.1 Timer Control/Status Register WD (TCSRWD)

TCSRWD performs the TCSRWD and TCWD write control. TCSRWD also controls the watchdog timer operation and indicates the operating state. TCSRWD must be rewritten by using the MOV instruction. The bit manipulation instruction cannot be used to change the setting value.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | B6WI | 1 | R/W | Bit 6 Write Inhibit<br>The TCWE bit can be written only when the write value of the B6WI bit is 0.<br>This bit is always read as 1. |
| 6 | TCWE | 0 | R/W | Timer Counter WD Write Enable<br>TCWD can be written when the TCWE bit is set to 1.<br>When writing data to this bit, the value for bit 7 must be 0. |
| 5 | B4WI | 1 | R/W | Bit 4 Write Inhibit<br>The TCSRWE bit can be written only when the write value of the B4WI bit is 0. This bit is always read as 1. |
| 4 | TCSRWE | 0 | R/W | Timer Control/Status Register WD Write Enable<br>The WDON and WRST bits can be written when the TCSRWE bit is set to 1.<br>When writing data to this bit, the value for bit 5 must be 0. |
| 3 | B2WI | 1 | R/W | Bit 2 Write Inhibit<br>This bit can be written to the WDON bit only when the write value of the B2WI bit is 0.<br>This bit is always read as 1. |
| 2 | WDON | 0 | R/W | Watchdog Timer On<br>TCWD starts counting up when WDON is set to 1 and halts when WDON is cleared to 0.<br>[Setting condition]<br>When 1 is written to the WDON bit while writing 0 to the B2WI bit when the TCSRWE bit = 1<br>[Clearing conditions]<br>• Reset by RES pin<br>• When 0 is written to the WDON bit while writing 0 to the B2WI when the TCSRWE bit=1 |
| 1 | B0WI | 1 | R/W | Bit 0 Write Inhibit<br>This bit can be written to the WRST bit only when the write value of the B0WI bit is 0. This bit is always read as 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | WRST | 0 | R/W | Watchdog Timer Reset<br>[Setting condition]<br>When TCWD overflows and an internal reset signal is generated<br>[Clearing conditions]<br>• Reset by RES pin<br>• When 0 is written to the WRST bit while writing 0 to the B0WI bit when the TCSRWE bit = 1 |

### 12.2.2 Timer Counter WD (TCWD)

TCWD is an 8-bit readable/writable up-counter. When TCWD overflows from H'FF to H'00, the internal reset signal is generated and the WRST bit in TCSRWD is set to 1. TCWD is initialized to H'00.

### 12.2.3 Timer Mode Register WD (TMWD)

TMWD selects the input clock.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 1 | — | Reserved<br>These bits are always read as 1. |
| 3 | CKS3 | 1 | R/W | Clock Select 3 to 0 |
| 2 | CKS2 | 1 | R/W | Select the clock to be input to TCWD. |
| 1 | CKS1 | 1 | R/W | 1000: Internal clock: counts on $\phi$/64 |
| 0 | CKS0 | 1 | R/W | 1001: Internal clock: counts on $\phi$/128<br>1010: Internal clock: counts on $\phi$/256<br>1011: Internal clock: counts on $\phi$/512<br>1100: Internal clock: counts on $\phi$/1024<br>1101: Internal clock: counts on $\phi$/2048<br>1110: Internal clock: counts on $\phi$/4096<br>1111: Internal clock: counts on $\phi$8192<br>0XXX: Internal oscillator<br>For the internal oscillator overflow periods, see section 17, Electrical Characteristics. |

Legend: X: Don't care.

## 12.3 Operation

The watchdog timer is provided with an 8-bit counter. If 1 is written to WDON while writing 0 to B2WI when the TCSRWE bit in TCSRWD is set to 1, TCWD begins counting up. (To operate the watchdog timer, two write accesses to TCSRWD are required.) When a clock pulse is input after the TCWD count value has reached H'FF, the watchdog timer overflows and an internal reset signal is generated. The internal reset signal is output for a period of 256 $\phi_{osc}$ clock cycles. TCWD is a writable counter, and when a value is set in TCWD, the count-up starts from that value. An overflow period in the range of 1 to 256 input clock cycles can therefore be set, according to the TCWD set value.

Figure 12.2 shows an example of watchdog timer operation.

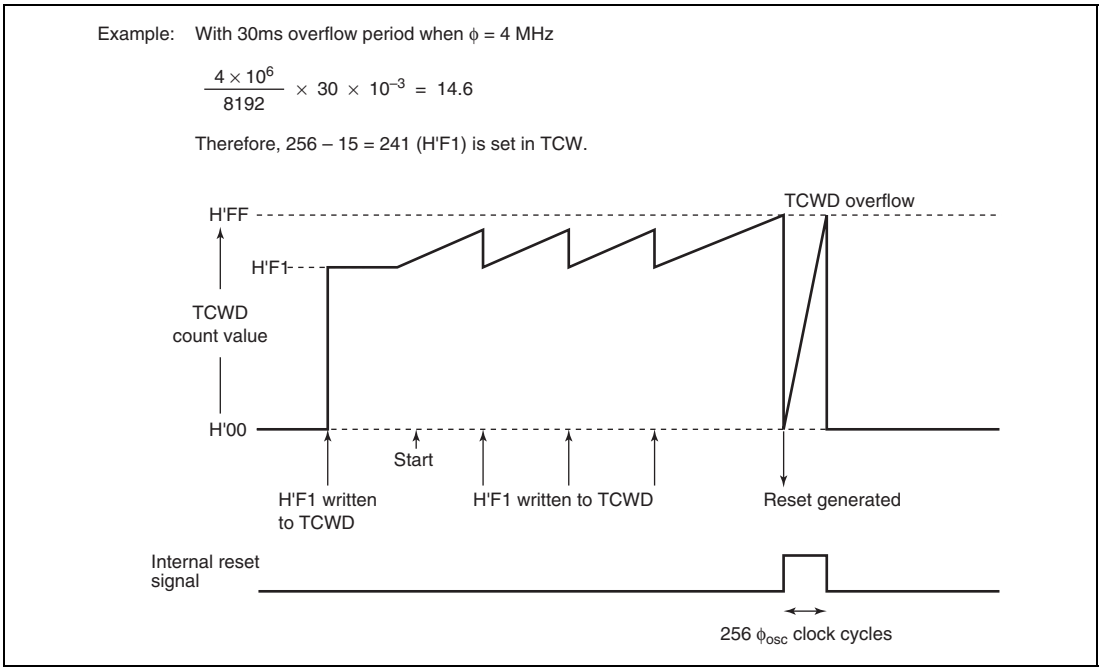



**Figure 12.2 Watchdog Timer Operation Example**

# Section 13 Serial Communication Interface 3 (SCI3)

Serial Communication Interface 3 (SCI3) can handle both asynchronous and clocked synchronous serial communication. In the asynchronous method, serial data communication can be carried out using standard asynchronous communication chips such as a Universal Asynchronous Receiver/Transmitter (UART) or an Asynchronous Communication Interface Adapter (ACIA). A function is also provided for serial communication between processors (multiprocessor communication function).

Figure 13.1 shows a block diagram of the SCI3.

## 13.1 Features

- Choice of asynchronous or clocked synchronous serial communication mode
- Full-duplex communication capability

  The transmitter and receiver are mutually independent, enabling transmission and reception to be executed simultaneously.

  Double-buffering is used in both the transmitter and the receiver, enabling continuous transmission and continuous reception of serial data.
- On-chip baud rate generator allows any bit rate to be selected
- External clock or on-chip baud rate generator can be selected as a transfer clock source.
- Six interrupt sources

  Transmit-end, transmit-data-empty, receive-data-full, overrun error, framing error, and parity error.

Asynchronous mode

- Data length: 7 or 8 bits
- Stop bit length: 1 or 2 bits
- Parity: Even, odd, or none
- Receive error detection: Parity, overrun, and framing errors
- Break detection: Break can be detected by reading the RxD pin level directly in the case of a framing error

Clocked synchronous mode

- Data length: 8 bits
- Receive error detection: Overrun errors detected

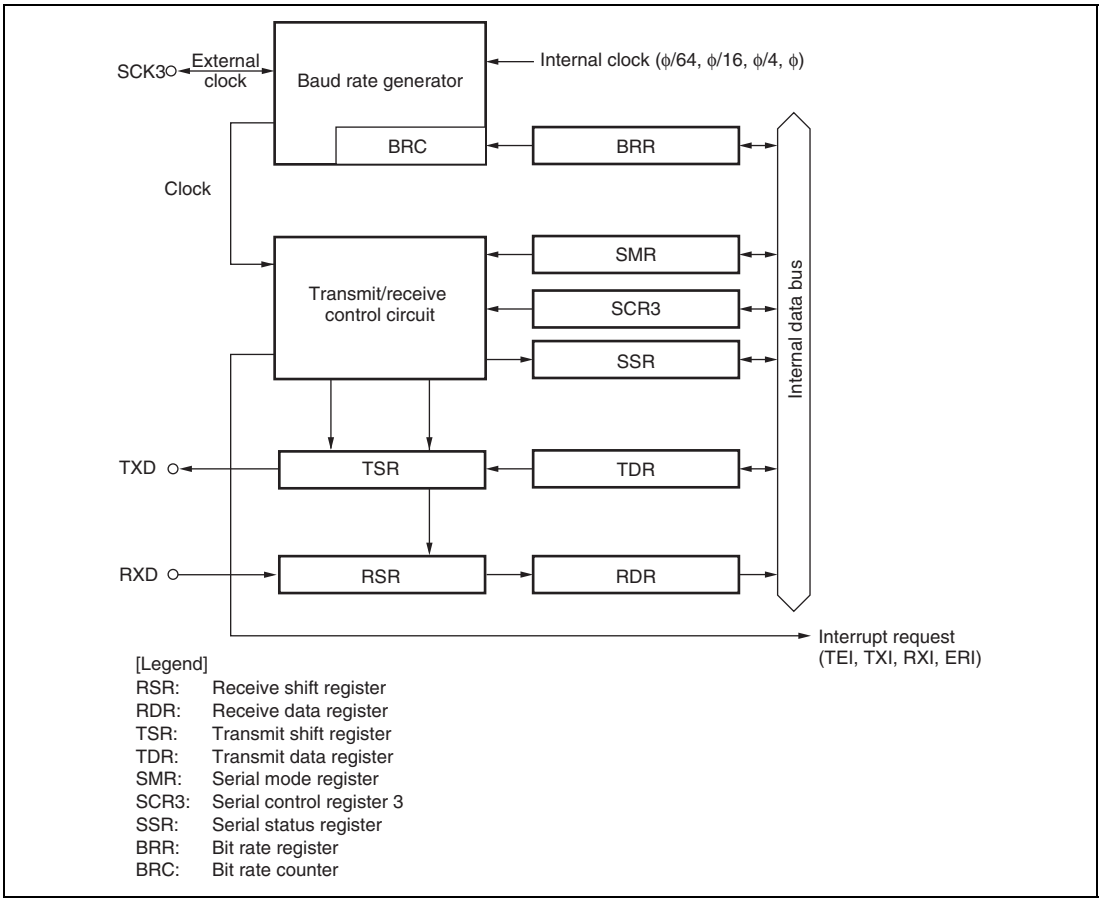



**Figure 13.1 Block Diagram of SCI3**

## 13.2 Input/Output Pins

Table 13.1 shows the SCI3 pin configuration.

**Table 13.1 Pin Configuration**

| Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|
| SCI3 clock | SCK3 | I/O | SCI3 clock input/output |
| SCI3 receive data input | RXD | Input | SCI3 receive data input |
| SCI3 transmit data output | TXD | Output | SCI3 transmit data output |

## 13.3 Register Descriptions

The SCI3 has the following registers.

- Receive shift register (RSR)
- Receive data register (RDR)
- Transmit shift register (TSR)
- Transmit data register (TDR)
- Serial mode register (SMR)
- Serial control register 3 (SCR3)
- Serial status register (SSR)
- Bit rate register (BRR)

### 13.3.1 Receive Shift Register (RSR)

RSR is a shift register that is used to receive serial data input from the RxD pin and convert it into parallel data. When one byte of data has been received, it is transferred to RDR automatically. RSR cannot be directly accessed by the CPU.

### 13.3.2 Receive Data Register (RDR)

RDR is an 8-bit register that stores received data. When the SCI3 has received one byte of serial data, it transfers the received serial data from RSR to RDR, where it is stored. After this, RSR is receive-enabled. As RSR and RDR function as a double buffer in this way, continuous receive operations are possible. After confirming that the RDRF bit in SSR is set to 1, read RDR only once. RDR cannot be written to by the CPU. RDR is initialized to H'00.

### 13.3.3 Transmit Shift Register (TSR)

TSR is a shift register that transmits serial data. To perform serial data transmission, the SCI3 first transfers transmit data from TDR to TSR automatically, then sends the data that starts from the LSB to the TXD pin. TSR cannot be directly accessed by the CPU.

### 13.3.4 Transmit Data Register (TDR)

TDR is an 8-bit register that stores data for transmission. When the SCI3 detects that TSR is empty, it transfers the transmit data written in TDR to TSR and starts transmission. The double-buffered structure of TDR and TSR enables continuous serial transmission. If the next transmit data has already been written to TDR during transmission of one-frame data, the SCI3 transfers the written data to TSR to continue transmission. To achieve reliable serial transmission, write transmit data to TDR only once after confirming that the TDRE bit in SSR is set to 1. TDR is initialized to H'FF.

### 13.3.5 Serial Mode Register (SMR)

SMR is used to set the SCI3's serial transfer format and select the on-chip baud rate generator clock source.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | COM | 0 | R/W | Communication Mode<br>0: Asynchronous mode<br>1: Clocked synchronous mode |
| 6 | CHR | 0 | R/W | Character Length (enabled only in asynchronous mode)<br>0: Selects 8 bits as the data length.<br>1: Selects 7 bits as the data length. |
| 5 | PE | 0 | R/W | Parity Enable (enabled only in asynchronous mode)<br>When this bit is set to 1, the parity bit is added to transmit data before transmission, and the parity bit is checked in reception. |
| 4 | PM | 0 | R/W | Parity Mode (enabled only when the PE bit is 1 in asynchronous mode)<br>0: Selects even parity.<br>1: Selects odd parity. |
| 3 | STOP | 0 | R/W | Stop Bit Length (enabled only in asynchronous mode)<br>Selects the stop bit length in transmission.<br>0: 1 stop bit<br>1: 2 stop bits<br>For reception, only the first stop bit is checked, regardless of the value in the bit. If the second stop bit is 0, it is treated as the start bit of the next transmit character. |
| 2 | MP | 0 | R/W | Multiprocessor Mode<br>When this bit is set to 1, the multiprocessor communication function is enabled. The PE bit and PM bit settings are invalid. In clocked synchronous mode, this bit should be cleared to 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | CKS1 | 0 | R/W | Clock Select 0 and 1 |
| 0 | CKS0 | 0 | R/W | These bits select the clock source for the on-chip baud rate generator.<br>00: ø clock (n = 0)<br>01: ø/4 clock (n = 1)<br>10: ø/16 clock (n = 2)<br>11: ø/64 clock (n = 3)<br>For the relationship between the bit rate register setting and the baud rate, see section 13.3.8, Bit Rate Register (BRR). n is the decimal representation of the value of n in BRR (see section 13.3.8, Bit Rate Register (BRR)). |

### 13.3.6 Serial Control Register 3 (SCR3)

SCR3 is a register that enables or disables SCI3 transfer operations and interrupt requests, and is also used to select the transfer clock source. For details on interrupt requests, refer to section 13.7, Interrupts.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TIE | 0 | R/W | Transmit Interrupt Enable<br>When this bit is set to 1, the TXI interrupt request is enabled. |
| 6 | RIE | 0 | R/W | Receive Interrupt Enable<br>When this bit is set to 1, RXI and ERI interrupt requests are enabled. |
| 5 | TE | 0 | R/W | Transmit Enable<br>When this bit is set to 1, transmission is enabled. |
| 4 | RE | 0 | R/W | Receive Enable<br>When this bit is set to 1, reception is enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | MPIE | 0 | R/W | Multiprocessor Interrupt Enable (enabled only when the MP bit in SMR is 1 in asynchronous mode)<br>When this bit is set to 1, receive data in which the multiprocessor bit is 0 is skipped, and setting of the RDRF, FER, and OER status flags in SSR is prohibited. On receiving data in which the multiprocessor bit is 1, this bit is automatically cleared and normal reception is resumed. For details, refer to section 13.6, Multiprocessor Communication Function. |
| 2 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>When this bit is set to 1, the TEI interrupt request is enabled. |
| 1 | CKE1 | 0 | R/W | Clock Enable 0 and 1 |
| 0 | CKE0 | 0 | R/W | Selects the clock source.<br>Asynchronous mode:<br>00: Internal baud rate generator<br>01: Internal baud rate generator<br>Outputs a clock of the same frequency as the bit rate from the SCK3 pin.<br>10: External clock<br>Inputs a clock with a frequency 16 times the bit rate from the SCK3 pin.<br>11: Reserved<br>Clocked synchronous mode:<br>00: Internal clock (SCK3 pin functions as clock output)<br>01: Reserved<br>10: External clock (SCK3 pin functions as clock input)<br>11: Reserved |

### 13.3.7 Serial Status Register (SSR)

SSR is a register containing status flags of the SCI3 and multiprocessor bits for transfer. 1 cannot be written to flags TDRE, RDRF, OER, PER, and FER; they can only be cleared.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TDRE | 1 | R/W | Transmit Data Register Empty<br>Displays whether TDR contains transmit data.<br>[Setting conditions]<br>• When the TE bit in SCR3 is 0<br>• When data is transferred from TDR to TSR<br>[Clearing conditions]<br>• When 0 is written to TDRE after reading TDRE = 1<br>• When the transmit data is written to TDR |
| 6 | RDRF | 0 | R/W | Receive Data Register Full<br>Indicates that the received data is stored in RDR.<br>[Setting condition]<br>• When serial reception ends normally and receive data is transferred from RSR to RDR<br>[Clearing conditions]<br>• When 0 is written to RDRF after reading RDRF = 1<br>• When data is read from RDR |
| 5 | OER | 0 | R/W | Overrun Error<br>[Setting condition]<br>• When an overrun error occurs in reception<br>[Clearing condition]<br>• When 0 is written to OER after reading OER = 1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | FER | 0 | R/W | Framing Error<br>[Setting condition]<br>• When a framing error occurs in reception<br>[Clearing condition]<br>• When 0 is written to FER after reading FER = 1 |
| 3 | PER | 0 | R/W | Parity Error<br>[Setting condition]<br>• When a parity error is generated during reception<br>[Clearing condition]<br>• When 0 is written to PER after reading PER = 1 |
| 2 | TEND | 1 | R | Transmit End<br>[Setting conditions]<br>• When the TE bit in SCR3 is 0<br>• When TDRE = 1 at transmission of the last bit of a 1-byte serial transmit character<br>[Clearing conditions]<br>• When 0 is written to TEND after reading TEND = 1<br>• When the transmit data is written to TDR |
| 1 | MPBR | 0 | R | Multiprocessor Bit Receive<br>MPBR stores the multiprocessor bit in the receive character data. When the RE bit in SCR3 is cleared to 0, its previous state is retained. |
| 0 | MPBT | 0 | R/W | Multiprocessor Bit Transfer<br>MPBT stores the multiprocessor bit to be added to the transmit character data. |

### 13.3.8 Bit Rate Register (BRR)

BRR is an 8-bit register that adjusts the bit rate. The initial value of BRR is H'FF. Table 13.2 shows the relationship between the N setting in BRR and the n setting in bits CKS1 and CKS0 of SMR in asynchronous mode. Table 13.3 shows the maximum bit rate for each frequency in asynchronous mode. The values shown in both tables 13.2 and 13.3 are values in active (high-speed) mode. Table 13.4 shows the relationship between the N setting in BRR and the n setting in bits CKS1 and CKS0 in SMR in clocked synchronous mode. The values shown in table 13.4 are values in active (high-speed) mode. The N setting in BRR and error for other operating frequencies and bit rates can be obtained by the following formulas:

**[Asynchronous Mode]**

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

$$\text{Error (\%)} = \left\{ \frac{\phi \times 10^6}{(N + 1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

**[Clocked Synchronous Mode]**

$$N = \frac{\phi}{8 \times 2^{2n-1} \times B} \times 10^6 - 1$$

Note: B: Bit rate (bit/s)
N: BRR setting for baud rate generator ($0 \leq N \leq 255$)
$\phi$: Operating frequency (MHz)
n: CKS1 and CKS0 setting for SMR ($0 \leq N \leq 3$)

**Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (1)**

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | | | 2.097152 | | | 2.4576 | | | 3 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 1 | 141 | 0.03 | 1 | 148 | –0.04 | 1 | 174 | –0.26 | 1 | 212 | 0.03 |
| 150 | 1 | 103 | 0.16 | 1 | 108 | 0.21 | 1 | 127 | 0.00 | 1 | 155 | 0.16 |
| 300 | 0 | 207 | 0.16 | 0 | 217 | 0.21 | 0 | 255 | 0.00 | 1 | 77 | 0.16 |
| 600 | 0 | 103 | 0.16 | 0 | 108 | 0.21 | 0 | 127 | 0.00 | 0 | 155 | 0.16 |
| 1200 | 0 | 51 | 0.16 | 0 | 54 | –0.70 | 0 | 63 | 0.00 | 0 | 77 | 0.16 |
| 2400 | 0 | 25 | 0.16 | 0 | 26 | 1.14 | 0 | 31 | 0.00 | 0 | 38 | 0.16 |
| 4800 | 0 | 12 | 0.16 | 0 | 13 | –2.48 | 0 | 15 | 0.00 | 0 | 19 | –2.34 |
| 9600 | 0 | 6 | –6.99 | 0 | 6 | –2.48 | 0 | 7 | 0.00 | 0 | 9 | –2.34 |
| 19200 | 0 | 2 | 8.51 | 0 | 2 | 13.78 | 0 | 3 | 0.00 | 0 | 4 | –2.34 |
| 31250 | 0 | 1 | 0.00 | 0 | 1 | 4.86 | 0 | 1 | 22.88 | 0 | 2 | 0.00 |
| 38400 | 0 | 1 | –18.62 | 0 | 1 | –14.67 | 0 | 1 | 0.00 | — | — | — |

Legend:

— : A setting is available but error occurs

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 3.6864 | | | 4 | | | 4.9152 | | | 5 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 64 | 0.70 | 2 | 70 | 0.03 | 2 | 86 | 0.31 | 2 | 88 | –0.25 |
| 150 | 1 | 191 | 0.00 | 1 | 207 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 |
| 300 | 1 | 95 | 0.00 | 1 | 103 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 |
| 600 | 0 | 191 | 0.00 | 0 | 207 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 |
| 1200 | 0 | 95 | 0.00 | 0 | 103 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 |
| 2400 | 0 | 47 | 0.00 | 0 | 51 | 0.16 | 0 | 63 | 0.00 | 0 | 64 | 0.16 |
| 4800 | 0 | 23 | 0.00 | 0 | 25 | 0.16 | 0 | 31 | 0.00 | 0 | 32 | –1.36 |
| 9600 | 0 | 11 | 0.00 | 0 | 12 | 0.16 | 0 | 15 | 0.00 | 0 | 15 | 1.73 |
| 19200 | 0 | 5 | 0.00 | 0 | 6 | –6.99 | 0 | 7 | 0.00 | 0 | 7 | 1.73 |
| 31250 | — | — | — | 0 | 3 | 0.00 | 0 | 4 | –1.70 | 0 | 4 | 0.00 |
| 38400 | 0 | 2 | 0.00 | 0 | 2 | 8.51 | 0 | 3 | 0.00 | 0 | 3 | 1.73 |

**Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (2)**

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | | | 6.144 | | | 7.3728 | | | 8 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 106 | –0.44 | 2 | 108 | 0.08 | 2 | 130 | –0.07 | 2 | 141 | 0.03 |
| 150 | 2 | 77 | 0.16 | 2 | 79 | 0.00 | 2 | 95 | 0.00 | 2 | 103 | 0.16 |
| 300 | 1 | 155 | 0.16 | 1 | 159 | 0.00 | 1 | 191 | 0.00 | 1 | 207 | 0.16 |
| 600 | 1 | 77 | 0.16 | 1 | 79 | 0.00 | 1 | 95 | 0.00 | 1 | 103 | 0.16 |
| 1200 | 0 | 155 | 0.16 | 0 | 159 | 0.00 | 0 | 191 | 0.00 | 0 | 207 | 0.16 |
| 2400 | 0 | 77 | 0.16 | 0 | 79 | 0.00 | 0 | 95 | 0.00 | 0 | 103 | 0.16 |
| 4800 | 0 | 38 | 0.16 | 0 | 39 | 0.00 | 0 | 47 | 0.00 | 0 | 51 | 0.16 |
| 9600 | 0 | 19 | –2.34 | 0 | 19 | 0.00 | 0 | 23 | 0.00 | 0 | 25 | 0.16 |
| 19200 | 0 | 9 | –2.34 | 0 | 9 | 0.00 | 0 | 11 | 0.00 | 0 | 12 | 0.16 |
| 31250 | 0 | 5 | 0.00 | 0 | 5 | 2.40 | 0 | 6 | 5.33 | 0 | 7 | 0.00 |
| 38400 | 0 | 4 | –2.34 | 0 | 4 | 0.00 | 0 | 5 | 0.00 | 0 | 6 | -6.99 |

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 9.8304 | | | 10 | | | 12 | | | 12.888 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 174 | –0.26 | 2 | 177 | –0.25 | 2 | 212 | 0.03 | 2 | 217 | 0.08 |
| 150 | 2 | 127 | 0.00 | 2 | 129 | 0.16 | 2 | 155 | 0.16 | 2 | 159 | 0.00 |
| 300 | 1 | 255 | 0.00 | 2 | 64 | 0.16 | 2 | 77 | 0.16 | 2 | 79 | 0.00 |
| 600 | 1 | 127 | 0.00 | 1 | 129 | 0.16 | 1 | 155 | 0.16 | 1 | 159 | 0.00 |
| 1200 | 0 | 255 | 0.00 | 1 | 64 | 0.16 | 1 | 77 | 0.16 | 1 | 79 | 0.00 |
| 2400 | 0 | 127 | 0.00 | 0 | 129 | 0.16 | 0 | 155 | 0.16 | 0 | 159 | 0.00 |
| 4800 | 0 | 63 | 0.00 | 0 | 64 | 0.16 | 0 | 77 | 0.16 | 0 | 79 | 0.00 |
| 9600 | 0 | 31 | 0.00 | 0 | 32 | –1.36 | 0 | 38 | 0.16 | 0 | 39 | 0.00 |
| 19200 | 0 | 15 | 0.00 | 0 | 15 | 1.73 | 0 | 19 | –2.34 | 0 | 19 | 0.00 |
| 31250 | 0 | 9 | –1.70 | 0 | 9 | 0.00 | 0 | 11 | 0.00 | 0 | 11 | 2.40 |
| 38400 | 0 | 7 | 0.00 | 0 | 7 | 1.73 | 0 | 9 | –2.34 | 0 | 9 | 0.00 |

**Table 13.2 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (3)**

| | Operating Frequency ϕ (MHz) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 14 | | | 14.7456 | | | 16 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 248 | –0.17 | 3 | 64 | 0.70 | 3 | 70 | 0.03 |
| 150 | 2 | 181 | 0.16 | 2 | 191 | 0.00 | 2 | 207 | 0.16 |
| 300 | 2 | 90 | 0.16 | 2 | 95 | 0.00 | 2 | 103 | 0.16 |
| 600 | 1 | 181 | 0.16 | 1 | 191 | 0.00 | 1 | 207 | 0.16 |
| 1200 | 1 | 90 | 0.16 | 1 | 95 | 0.00 | 1 | 103 | 0.16 |
| 2400 | 0 | 181 | 0.16 | 0 | 191 | 0.00 | 0 | 207 | 0.16 |
| 4800 | 0 | 90 | 0.16 | 0 | 95 | 0.00 | 0 | 103 | 0.16 |
| 9600 | 0 | 45 | –0.93 | 0 | 47 | 0.00 | 0 | 51 | 0.16 |
| 19200 | 0 | 22 | –0.93 | 0 | 23 | 0.00 | 0 | 25 | 0.16 |
| 31250 | 0 | 13 | 0.00 | 0 | 14 | –1.70 | 0 | 15 | 0.00 |
| 38400 | — | — | — | 0 | 11 | 0.00 | 0 | 12 | 0.16 |

Legend:

—: A setting is available but error occurs.

**Table 13.3 Maximum Bit Rate for Each Frequency (Asynchronous Mode)**

| ϕ (MHz) | Maximum Bit Rate (bit/s) | n | N | ϕ (MHz) | Maximum Bit Rate (bit/s) | n | N |
|---|---|---|---|---|---|---|---|
| 2 | 62500 | 0 | 0 | 7.3728 | 230400 | 0 | 0 |
| 2.097152 | 65536 | 0 | 0 | 8 | 250000 | 0 | 0 |
| 2.4576 | 76800 | 0 | 0 | 9.8304 | 307200 | 0 | 0 |
| 3 | 93750 | 0 | 0 | 10 | 312500 | 0 | 0 |
| 3.6864 | 115200 | 0 | 0 | 12 | 375000 | 0 | 0 |
| 4 | 125000 | 0 | 0 | 12.288 | 384000 | 0 | 0 |
| 4.9152 | 153600 | 0 | 0 | 14 | 437500 | 0 | 0 |
| 5 | 156250 | 0 | 0 | 14.7456 | 460800 | 0 | 0 |
| 6 | 187500 | 0 | 0 | 16 | 500000 | 0 | 0 |
| 6.144 | 192000 | 0 | 0 | | | | |

**Table 13.4 BRR Settings for Various Bit Rates (Clocked Synchronous Mode)**

| Bit Rate (bit/s) | Operating Frequency ϕ (MHz) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | | 4 | | 8 | | 10 | | 16 | |
| | n | N | n | N | n | N | n | N | n | N |
| 110 | 3 | 70 | — | — | — | — | — | — | | |
| 250 | 2 | 124 | 2 | 249 | 3 | 124 | — | — | 3 | 249 |
| 500 | 1 | 249 | 2 | 124 | 2 | 249 | — | — | 3 | 124 |
| 1k | 1 | 124 | 1 | 249 | 2 | 124 | — | — | 2 | 249 |
| 2.5k | 0 | 199 | 1 | 99 | 1 | 199 | 1 | 249 | 2 | 99 |
| 5k | 0 | 99 | 0 | 199 | 1 | 99 | 1 | 124 | 1 | 199 |
| 10k | 0 | 49 | 0 | 99 | 0 | 199 | 0 | 249 | 1 | 99 |
| 25k | 0 | 19 | 0 | 39 | 0 | 79 | 0 | 99 | 0 | 159 |
| 50k | 0 | 9 | 0 | 19 | 0 | 39 | 0 | 49 | 0 | 79 |
| 100k | 0 | 4 | 0 | 9 | 0 | 19 | 0 | 24 | 0 | 39 |
| 250k | 0 | 1 | 0 | 3 | 0 | 7 | 0 | 9 | 0 | 15 |
| 500k | 0 | 0* | 0 | 1 | 0 | 3 | 0 | 4 | 0 | 7 |
| 1M | | | 0 | 0* | 0 | 1 | — | — | 0 | 3 |
| 2M | | | | | 0 | 0* | — | — | 0 | 1 |
| 2.5M | | | | | | | 0 | 0* | — | — |
| 4M | | | | | | | | | 0 | 0* |

Legend:

Blank : No setting is available.

— : A setting is available but error occurs.

* : Continuous transfer is not possible.

## 13.4 Operation in Asynchronous Mode

Figure 13.2 shows the general format for asynchronous serial communication. One frame consists of a start bit (low level), followed by data (in LSB-first order), a parity bit (high or low level), and finally stop bits (high level). Inside the SCI3, the transmitter and receiver are independent units, enabling full duplex. Both the transmitter and the receiver also have a double-buffered structure, so data can be read or written during transmission or reception, enabling continuous data transfer.

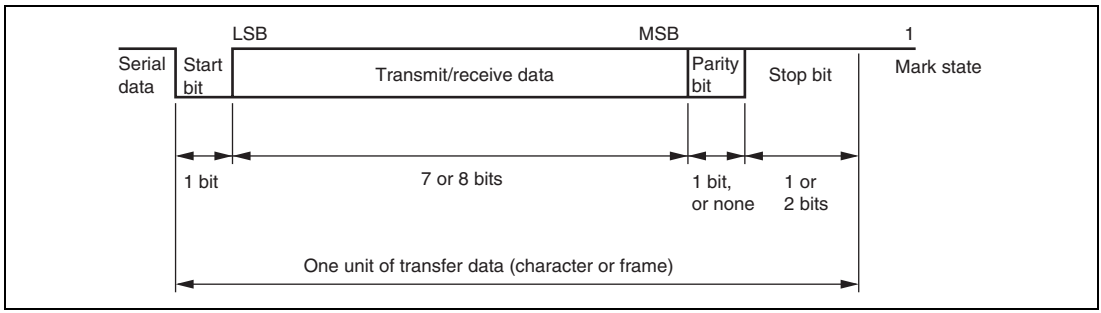



**Figure 13.2 Data Format in Asynchronous Communication**

### 13.4.1 Clock

Either an internal clock generated by the on-chip baud rate generator or an external clock input at the SCK3 pin can be selected as the SCI3's serial clock source, according to the setting of the COM bit in SMR and the CKE0 and CKE1 bits in SCR3. When an external clock is input at the SCK3 pin, the clock frequency should be 16 times the bit rate used.

When the SCI3 is operated on an internal clock, the clock can be output from the SCK3 pin. The frequency of the clock output in this case is equal to the bit rate, and the phase is such that the rising edge of the clock is in the middle of the transmit data, as shown in figure 13.3.

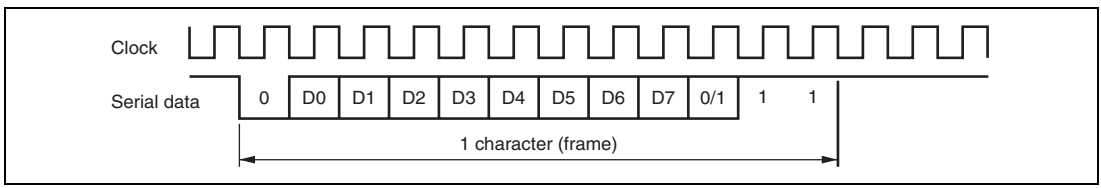
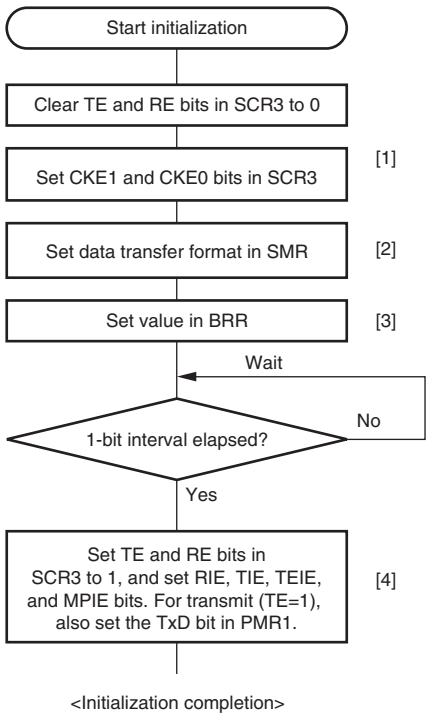



**Figure 13.3 Relationship between Output Clock and Transfer Data Phase (Asynchronous Mode) (Example with 8-Bit Data, Parity, Two Stop Bits)**

### 13.4.2 SCI3 Initialization

Follow the flowchart as shown in figure 13.4 to initialize the SCI3. When the TE bit is cleared to 0, the TDRE flag is set to 1. Note that clearing the RE bit to 0 does not initialize the contents of the RDRF, PER, FER, and OER flags, or the contents of RDR. When the external clock is used in asynchronous mode, the clock must be supplied even during initialization.




[1] Set the clock selection in SCR3. Be sure to clear bits RIE, TIE, TEIE, and MPIE, and bits TE and RE, to 0.

When the clock output is selected in asynchronous mode, clock is output immediately after CKE1 and CKE0 settings are made. When the clock output is selected at reception in clocked synchronous mode, clock is output immediately after CKE1, CKE0, and RE are set to 1.

[2] Set the data transfer format in SMR.

[3] Write a value corresponding to the bit rate to BRR. Not necessary if an external clock is used.

[4] Wait at least one bit interval, then set the TE bit or RE bit in SCR3 to 1. RE settings enable the RXD pin to be used. For transmission, set the TXD bit in PMR1 to 1 to enable the TXD output pin to be used. Also set the RIE, TIE, TEIE, and MPIE bits, depending on whether interrupts are required. In asynchronous mode, the bits are marked at transmission and idled at reception to wait for the start bit.

**Figure 13.4 Sample SCI3 Initialization Flowchart**

### 13.4.3 Data Transmission

Figure 13.5 shows an example of operation for transmission in asynchronous mode. In transmission, the SCI3 operates as described below.

1. The SCI3 monitors the TDRE flag in SSR. If the flag is cleared to 0, the SCI3 recognizes that data has been written to TDR, and transfers the data from TDR to TSR.
2. After transferring data from TDR to TSR, the SCI3 sets the TDRE flag to 1 and starts transmission. If the TIE bit is set to 1 at this time, a TXI interrupt request is generated. Continuous transmission is possible because the TXI interrupt routine writes next transmit data to TDR before transmission of the current transmit data has been completed.
3. The SCI3 checks the TDRE flag at the timing for sending the stop bit.
4. If the TDRE flag is 0, the data is transferred from TDR to TSR, the stop bit is sent, and then serial transmission of the next frame is started.
5. If the TDRE flag is 1, the TEND flag in SSR is set to 1, the stop bit is sent, and then the “mark state” is entered, in which 1 is output. If the TEIE bit in SCR3 is set to 1 at this time, a TEI interrupt request is generated.
6. Figure 13.6 shows a sample flowchart for transmission in asynchronous mode.

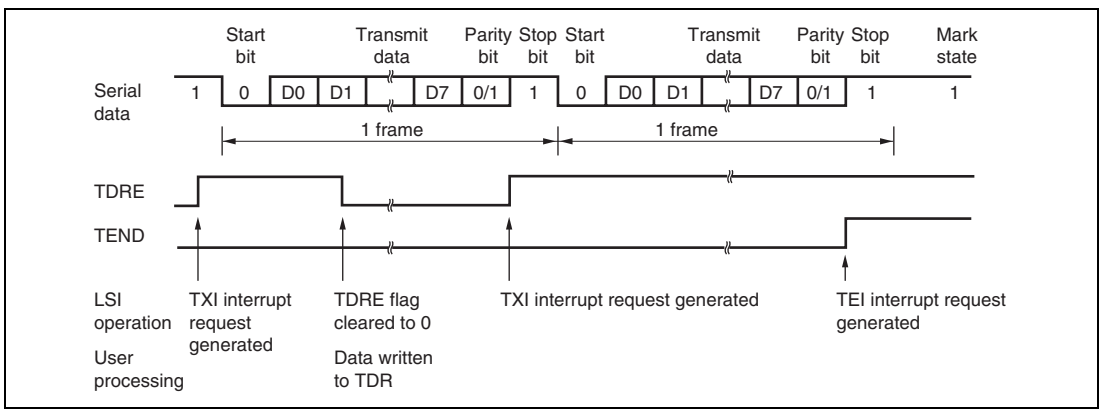
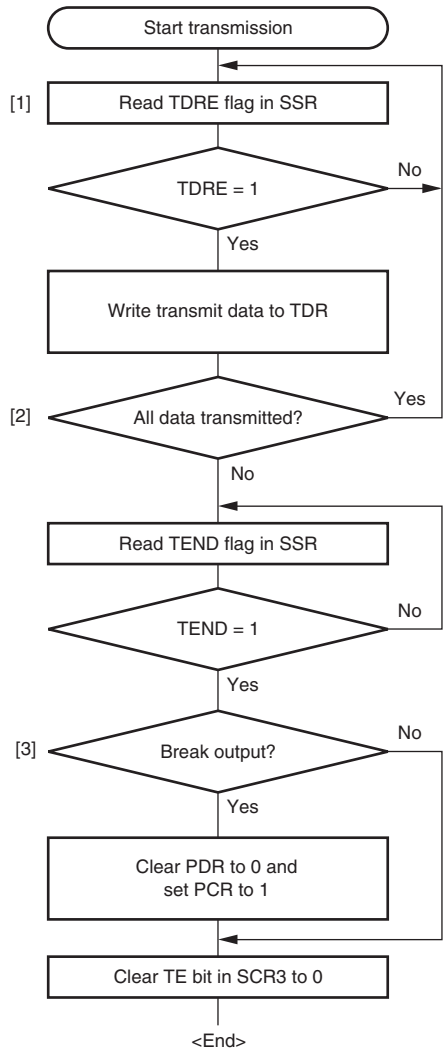



**Figure 13.5 Example SCI3 Operation in Transmission in Asynchronous Mode (8-Bit Data, Parity, One Stop Bit)**




[1] Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR. When data is written to TDR, the TDRE flag is automaticaly cleared to 0.

[2] To continue serial transmission, read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR. When data is written to TDR, the TDRE flag is automaticaly cleared to 0.

[3] To output a break in serial transmission, after setting PCR to 1 and PDR to 0, clear TxD in PMR1 to 0, then clear the TE bit in SCR3 to 0.

**Figure 13.6 Sample Serial Transmission Flowchart (Asynchronous Mode)**

### 13.4.4 Serial Data Reception

Figure 13.7 shows an example of operation for reception in asynchronous mode. In serial reception, the SCI operates as described below.

1. The SCI3 monitors the communication line. If a start bit is detected, the SCI3 performs internal synchronization, receives data in RSR, and checks the parity bit and stop bit.
2. If an overrun error occurs (when reception of the next data is completed while the RDRF flag is still set to 1), the OER bit in SSR is set to 1. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR.
3. If a parity error is detected, the PER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated.
4. If a framing error is detected (when the stop bit is 0), the FER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated.
5. If reception is completed successfully, the RDRF bit in SSR is set to 1, and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an RXI interrupt request is generated. Continuous reception is possible because the RXI interrupt routine reads the receive data transferred to RDR before reception of the next receive data has been completed.

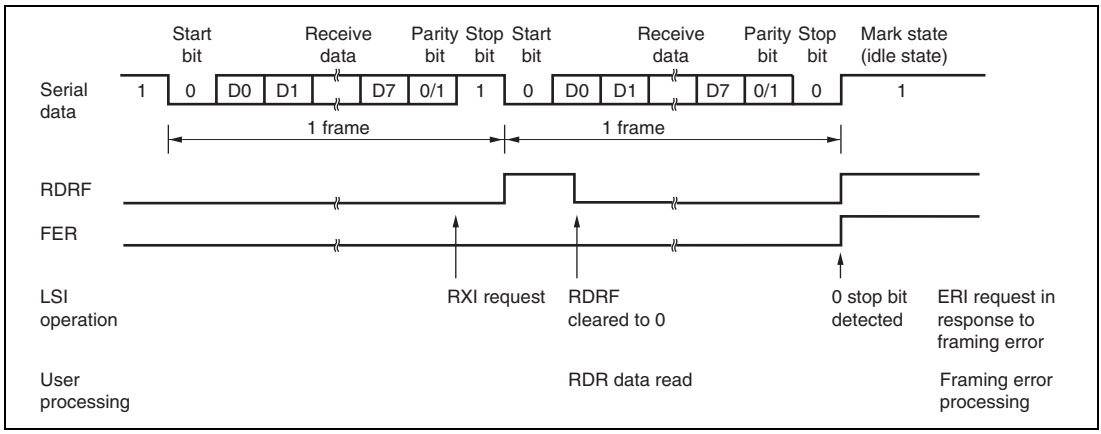


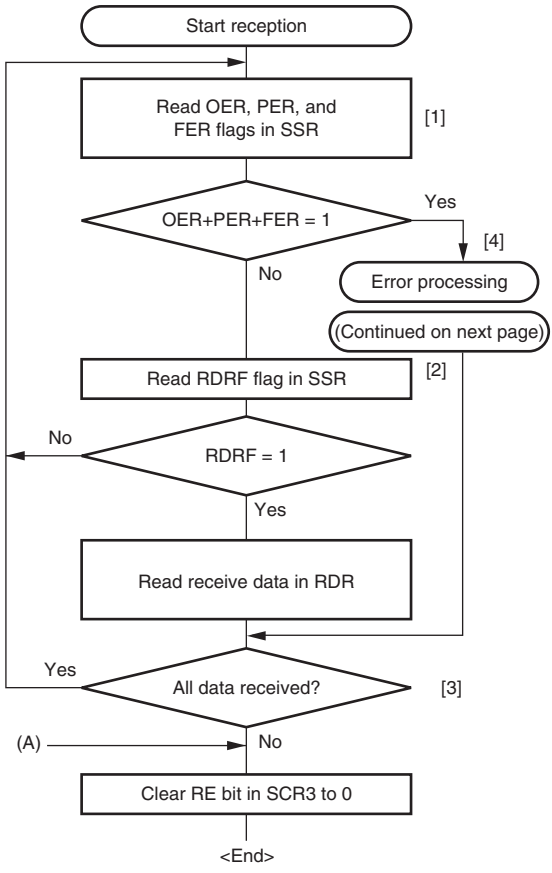

**Figure 13.7 Example SCI3 Operation in Reception in Asynchronous Mode (8-Bit Data, Parity, One Stop Bit)**

Table 13.5 shows the states of the SSR status flags and receive data handling when a receive error is detected. If a receive error is detected, the RDRF flag retains its state before receiving data. Reception cannot be resumed while a receive error flag is set to 1. Accordingly, clear the OER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 13.8 shows a sample flowchart for serial data reception.

**Table 13.5 SSR Status Flags and Receive Data Handling**

| SSR Status Flag | | | | | |
|---|---|---|---|---|---|
| **RDRF*** | **OER** | **FER** | **PER** | **Receive Data** | **Receive Error Type** |
| 1 | 1 | 0 | 0 | Lost | Overrun error |
| 0 | 0 | 1 | 0 | Transferred to RDR | Framing error |
| 0 | 0 | 0 | 1 | Transferred to RDR | Parity error |
| 1 | 1 | 1 | 0 | Lost | Overrun error + framing error |
| 1 | 1 | 0 | 1 | Lost | Overrun error + parity error |
| 0 | 0 | 1 | 1 | Transferred to RDR | Framing error + parity error |
| 1 | 1 | 1 | 1 | Lost | Overrun error + framing error + parity error |

Note: * The RDRF flag retains the state it had before data reception.




[1] Read the OER, PER, and FER flags in SSR to identify the error. If a receive error occurs, performs the appropriate error processing.

[2] Read SSR and check that RDRF = 1, then read the receive data in RDR. The RDRF flag is cleared automatically.

[3] To continue serial reception, before the stop bit for the current frame is received, read the RDRF flag and read RDR.
The RDRF flag is cleared automatically.

[4] If a receive error occurs, read the OER, PER, and FER flags in SSR to identify the error. After performing the appropriate error processing, ensure that the OER, PER, and FER flags are all cleared to 0. Reception cannot be resumed if any of these flags are set to 1. In the case of a framing error, a break can be detected by reading the value of the input port corresponding to the RxD pin.

**Figure 13.8 Sample Serial Data Reception Flowchart (Asynchronous mode) (1)**

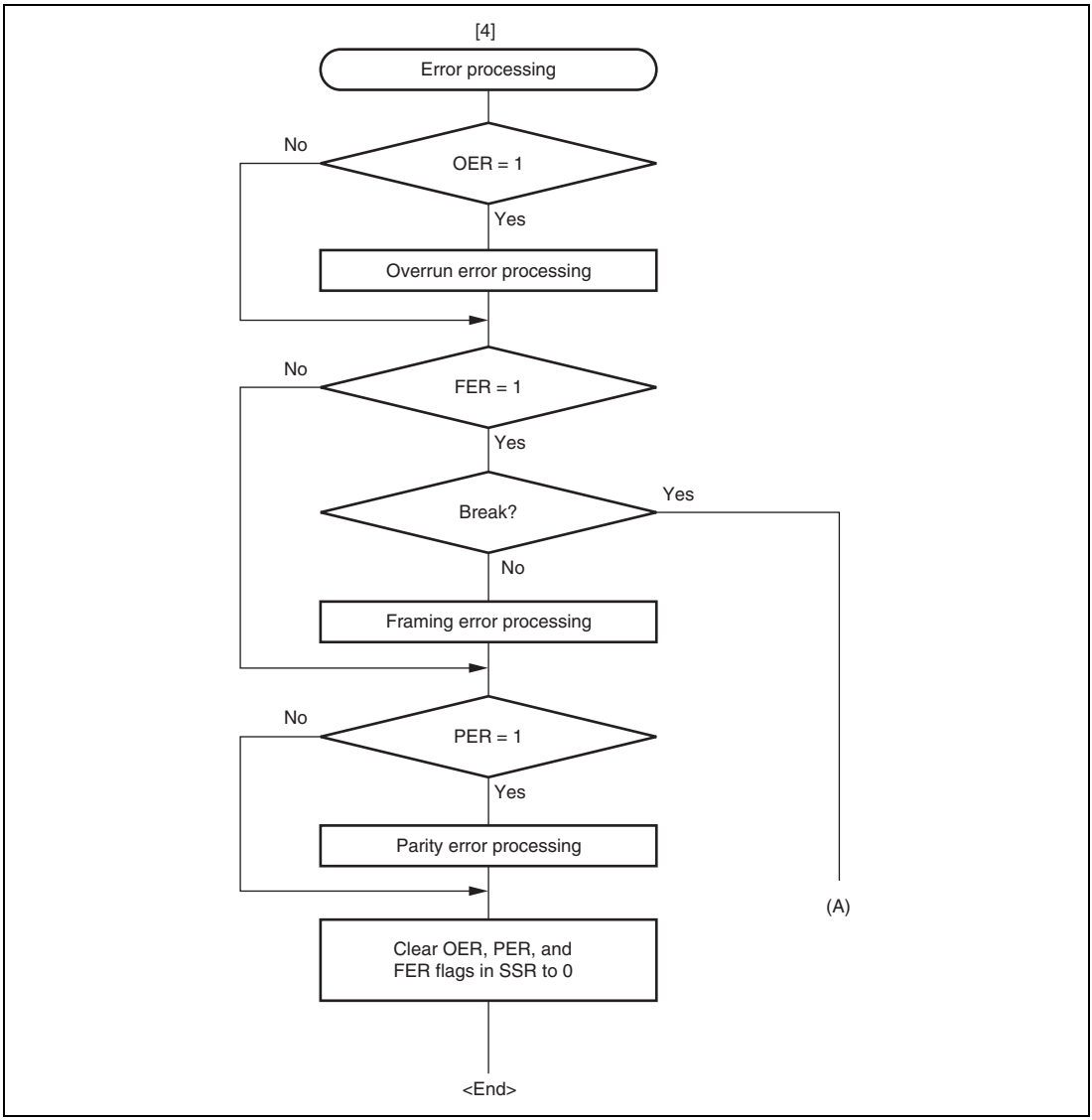



**Figure 13.8 Sample Serial Reception Data Flowchart (2)**

## 13.5 Operation in Clocked Synchronous Mode

Figure 13.9 shows the general format for clocked synchronous communication. In clocked synchronous mode, data is transmitted or received synchronous with clock pulses. A single character in the transmit data consists of the 8-bit data starting from the LSB. In clocked synchronous serial communication, data on the transmission line is output from one falling edge of the serial clock to the next. In clocked synchronous mode, the SCI3 receives data in synchronous with the rising edge of the serial clock. After 8-bit data is output, the transmission line holds the MSB state. In clocked synchronous mode, no parity or multiprocessor bit is added. Inside the SCI3, the transmitter and receiver are independent units, enabling full-duplex communication through the use of a common clock. Both the transmitter and the receiver also have a double-buffered structure, so data can be read or written during transmission or reception, enabling continuous data transfer.

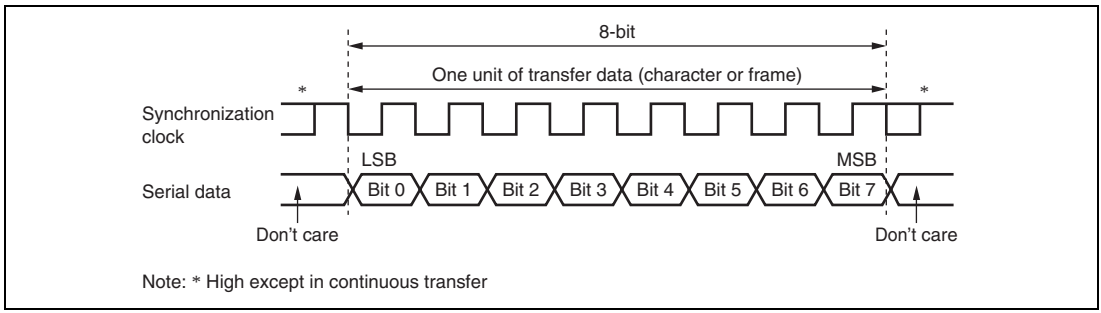



**Figure 13.9 Data Format in Clocked Synchronous Communication**

### 13.5.1 Clock

Either an internal clock generated by the on-chip baud rate generator or an external synchronization clock input at the SCK3 pin can be selected, according to the setting of the COM bit in SMR and CKE0 and CKE1 bits in SCR3. When the SCI3 is operated on an internal clock, the serial clock is output from the SCK3 pin. Eight serial clock pulses are output in the transfer of one character, and when no transfer is performed the clock is fixed high.

### 13.5.2 SCI3 Initialization

Before transmitting and receiving data, the SCI3 should be initialized as described in a sample flowchart in figure 13.4.

### 13.5.3 Serial Data Transmission

Figure 13.10 shows an example of SCI3 operation for transmission in clocked synchronous mode. In serial transmission, the SCI3 operates as described below.

1. The SCI3 monitors the TDRE flag in SSR, and if the flag is 0, the SCI recognizes that data has been written to TDR, and transfers the data from TDR to TSR.
2. The SCI3 sets the TDRE flag to 1 and starts transmission. If the TIE bit in SCR3 is set to 1 at this time, a transmit data empty interrupt (TXI) is generated.
3. 8-bit data is sent from the TxD pin synchronized with the output clock when output clock mode has been specified, and synchronized with the input clock when use of an external clock has been specified. Serial data is transmitted sequentially from the LSB (bit 0), from the TXD pin.
4. The SCI checks the TDRE flag at the timing for sending the MSB (bit 7).
5. If the TDRE flag is cleared to 0, data is transferred from TDR to TSR, and serial transmission of the next frame is started.
6. If the TDRE flag is set to 1, the TEND flag in SSR is set to 1, and the TDRE flag maintains the output state of the last bit. If the TEIE bit in SCR3 is set to 1 at this time, a TEI interrupt request is generated.
7. The SCK3 pin is fixed high.

Figure 13.11 shows a sample flowchart for serial data transmission. Even if the TDRE flag is cleared to 0, transmission will not start while a receive error flag (OER, FER, or PER) is set to 1. Make sure that the receive error flags are cleared to 0 before starting transmission.

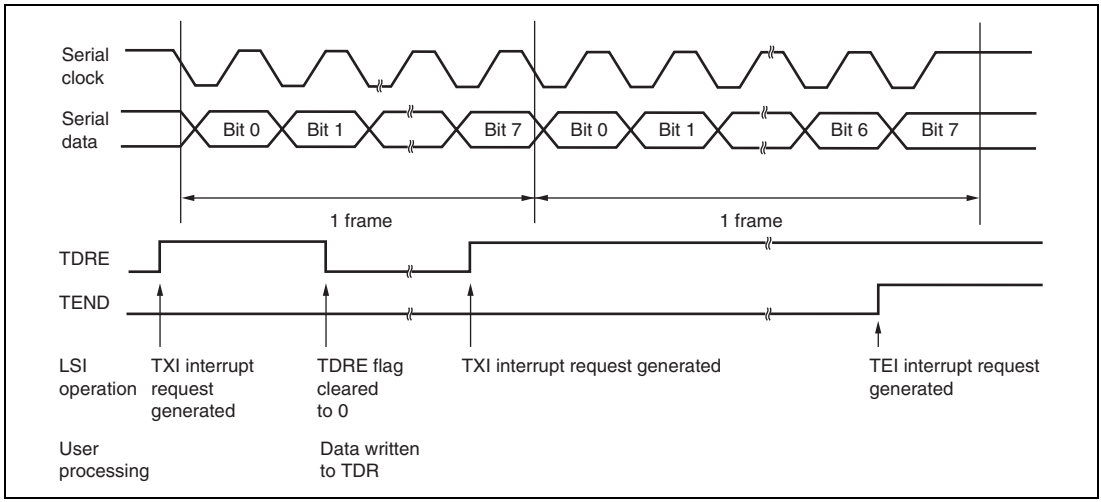
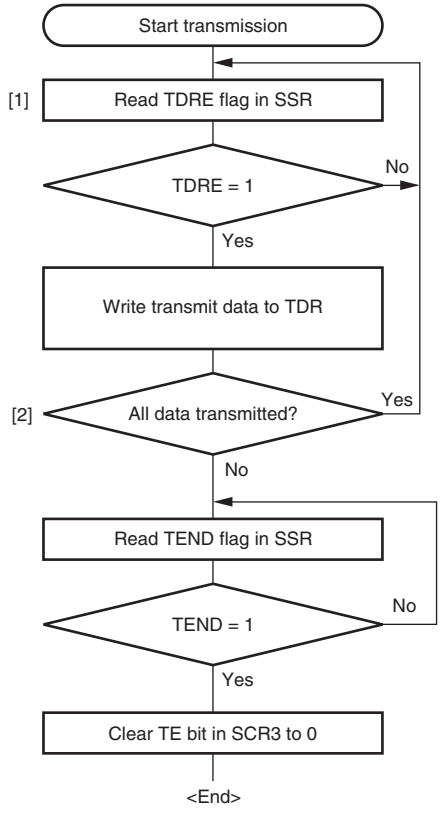



**Figure 13.10 Example of SCI3 Operation in Transmission in Clocked Synchronous Mode**




[1] Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR. When data is written to TDR, the TDRE flag is automatically cleared to 0 and clocks are output to start the data transmission.

[2] To continue serial transmission, be sure to read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR. When data is written to TDR, the TDRE flag is automatically cleared to 0.

**Figure 13.11 Sample Serial Transmission Flowchart (Clocked Synchronous Mode)**

### 13.5.4 Serial Data Reception (Clocked Synchronous Mode)

Figure 13.12 shows an example of SCI3 operation for reception in clocked synchronous mode. In serial reception, the SCI3 operates as described below.

1. The SCI3 performs internal initialization synchronous with a synchronous clock input or output, starts receiving data.
2. The SCI3 stores the received data in RSR.
3. If an overrun error occurs (when reception of the next data is completed while the RDRF flag in SSR is still set to 1), the OER bit in SSR is set to 1. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated, receive data is not transferred to RDR, and the RDRF flag remains to be set to 1.
4. If reception is completed successfully, the RDRF bit in SSR is set to 1, and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an RXI interrupt request is generated.

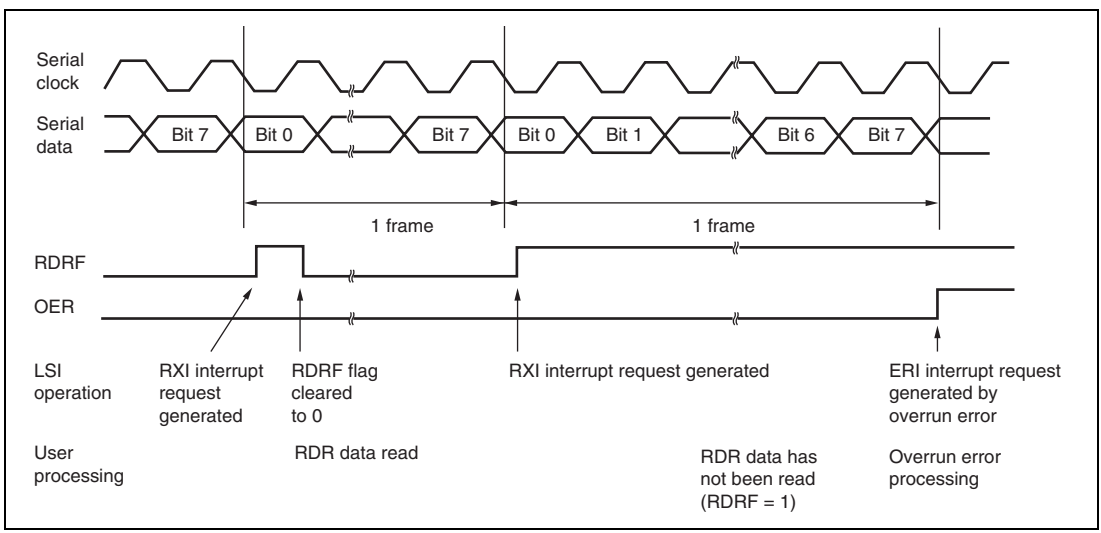
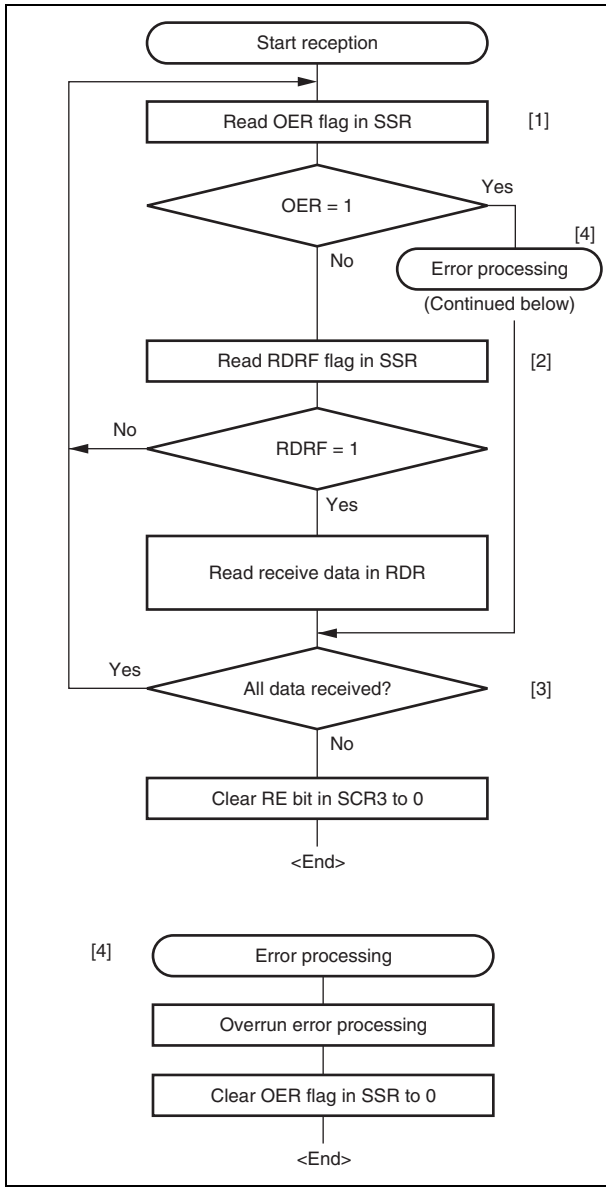



**Figure 13.12 Example of SCI3 Reception Operation in Clocked Synchronous Mode**

Reception cannot be resumed while a receive error flag is set to 1. Accordingly, clear the OER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 13.13 shows a sample flowchart for serial data reception.




[1] Read the OER flag in SSR to determine if there is an error. If an overrun error has occurred, execute overrun error processing.

[2] Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR. When data is read from RDR, the RDRF flag is automatically cleared to 0.

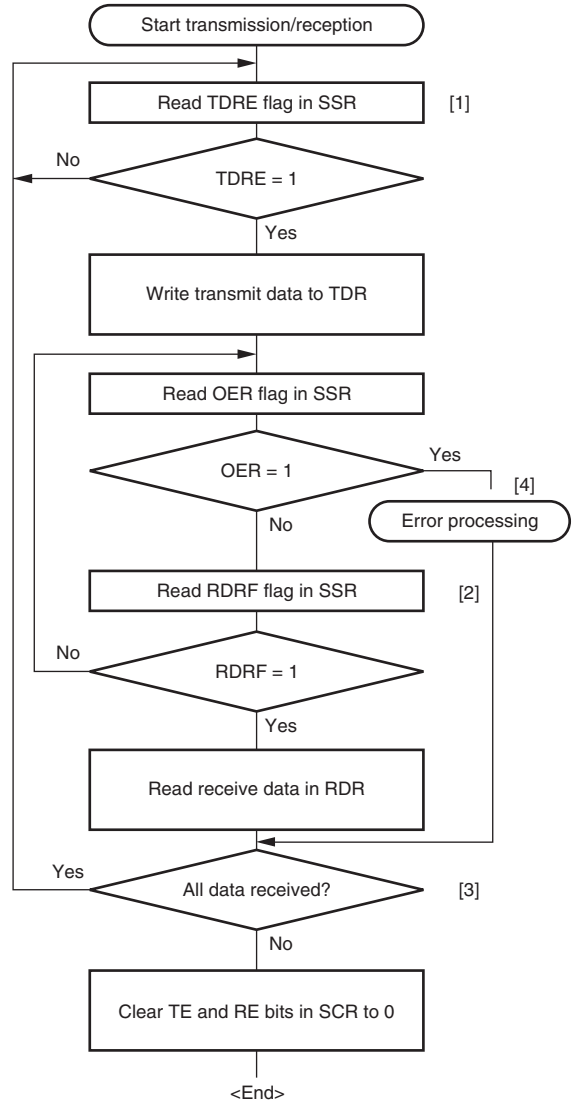
[3] To continue serial reception, before the MSB (bit 7) of the current frame is received, reading the RDRF flag and reading RDR should be finished. When data is read from RDR, the RDRF flag is automatically cleared to 0.

[4] If an overrun error occurs, read the OER flag in SSR, and after performing the appropriate error processing, clear the OER flag to 0. Reception cannot be resumed if the OER flag is set to 1.

**Figure 13.13 Sample Serial Reception Flowchart (Clocked Synchronous Mode)**

### 13.5.5 Simultaneous Serial Data Transmission and Reception

Figure 13.14 shows a sample flowchart for simultaneous serial transmit and receive operations. The following procedure should be used for simultaneous serial data transmit and receive operations. To switch from transmit mode to simultaneous transmit and receive mode, after checking that the SCI3 has finished transmission and the TDRE and TEND flags are set to 1, clear TE to 0. Then simultaneously set TE and RE to 1 with a single instruction. To switch from receive mode to simultaneous transmit and receive mode, after checking that the SCI3 has finished reception, clear RE to 0. Then after checking that the RDRF and receive error flags (OER, FER, and PER) are cleared to 0, simultaneously set TE and RE to 1 with a single instruction.




[1] Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR.
When data is written to TDR, the TDRE flag is automatically cleared to 0.

[2] Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR.
When data is read from RDR, the RDRF flag is automatically cleared to 0.

[3] To continue serial transmission/ reception, before the MSB (bit 7) of the current frame is received, finish reading the RDRF flag, reading RDR. Also, before the MSB (bit 7) of the current frame is transmitted, read 1 from the TDRE flag to confirm that writing is possible. Then write data to TDR.
When data is written to TDR, the TDRE flag is automatically cleared to 0. When data is read from RDR, the RDRF flag is automatically cleared to 0.

[4] If an overrun error occurs, read the OER flag in SSR, and after performing the appropriate error processing, clear the OER flag to 0. Transmission/reception cannot be resumed if the OER flag is set to 1. For overrun error processing, see figure 13.13.

**Figure 13.14 Sample Flowchart of Simultaneous Serial Transmit and Receive Operations (Clocked Synchronous Mode)**

## 13.6 Multiprocessor Communication Function

Use of the multiprocessor communication function enables data transfer between a number of processors sharing communication lines by asynchronous serial communication using the multiprocessor format, in which a multiprocessor bit is added to the transfer data. When multiprocessor communication is performed, each receiving station is addressed by a unique ID code. The serial communication cycle consists of two component cycles; an ID transmission cycle that specifies the receiving station, and a data transmission cycle. The multiprocessor bit is used to differentiate between the ID transmission cycle and the data transmission cycle. If the multiprocessor bit is 1, the cycle is an ID transmission cycle; if the multiprocessor bit is 0, the cycle is a data transmission cycle. Figure 13.15 shows an example of inter-processor communication using the multiprocessor format. The transmitting station first sends the ID code of the receiving station with which it wants to perform serial communication as data with a 1 multiprocessor bit added. It then sends transmit data as data with a 0 multiprocessor bit added. When data with a 1 multiprocessor bit is received, the receiving station compares that data with its own ID. The station whose ID matches then receives the data sent next. Stations whose IDs do not match continue to skip data until data with a 1 multiprocessor bit is again received.

The SCI3 uses the MPIE bit in SCR3 to implement this function. When the MPIE bit is set to 1, transfer of receive data from RSR to RDR, error flag detection, and setting the SSR status flags, RDRF, FER, and OER to 1, are inhibited until data with a 1 multiprocessor bit is received. On reception of a receive character with a 1 multiprocessor bit, the MPBR bit in SSR is set to 1 and the MPIE bit is automatically cleared, thus normal reception is resumed. If the RIE bit in SCR3 is set to 1 at this time, an RXI interrupt is generated.

When the multiprocessor format is selected, the parity bit setting is rendered invalid. All other bit settings are the same as those in normal asynchronous mode. The clock used for multiprocessor communication is the same as that in normal asynchronous mode.

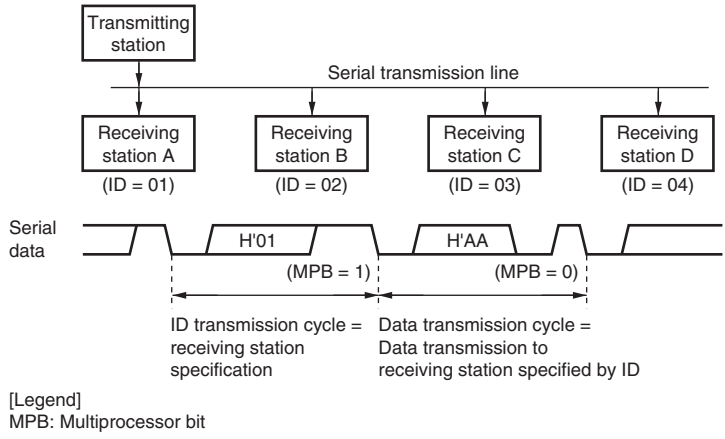



**Figure 13.15 Example of Communication Using Multiprocessor Format (Transmission of Data H'AA to Receiving Station A)**

### 13.6.1 Multiprocessor Serial Data Transmission

Figure 13.16 shows a sample flowchart for multiprocessor serial data transmission. For an ID transmission cycle, set the MPBT bit in SSR to 1 before transmission. For a data transmission cycle, clear the MPBT bit in SSR to 0 before transmission. All other SCI3 operations are the same as those in asynchronous mode.

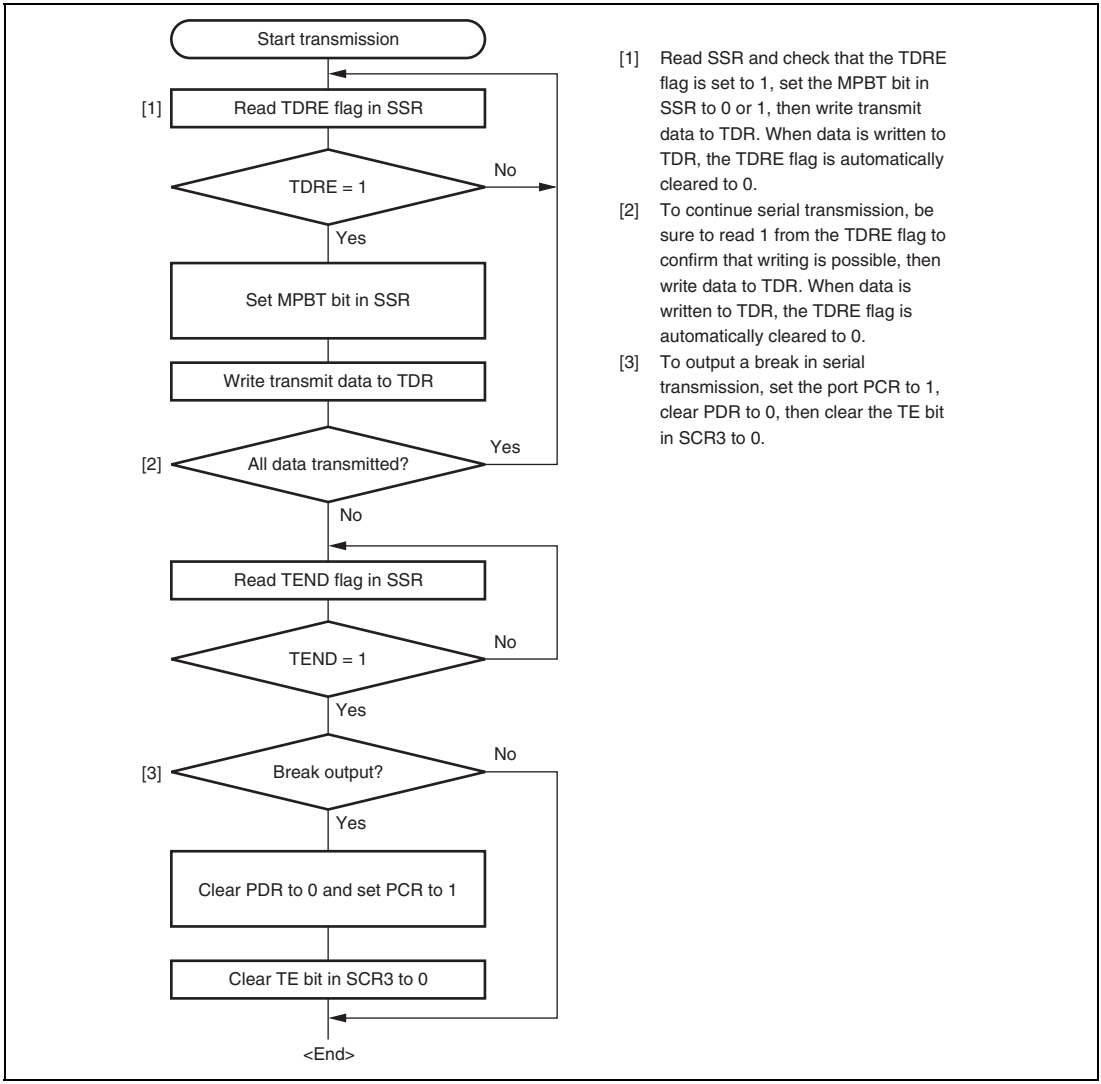
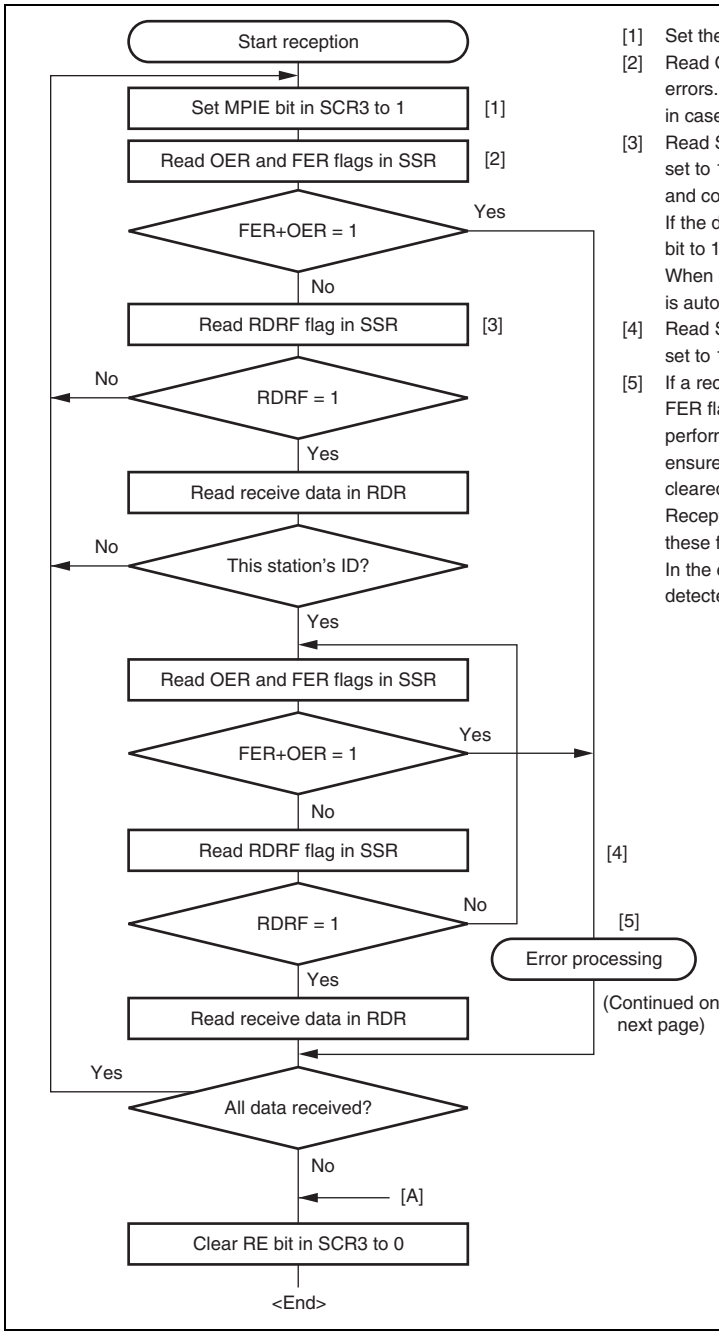



**Figure 13.16 Sample Multiprocessor Serial Transmission Flowchart**

### 13.6.2 Multiprocessor Serial Data Reception

Figure 13.17 shows a sample flowchart for multiprocessor serial data reception. If the MPIE bit in SCR3 is set to 1, data is skipped until data with a 1 multiprocessor bit is received. On receiving data with a 1 multiprocessor bit, the receive data is transferred to RDR. An RXI interrupt request is generated at this time. All other SCI3 operations are the same as in asynchronous mode. Figure 13.18 shows an example of SCI3 operation for multiprocessor format reception.




[1] Set the MPIE bit in SCR3 to 1.

[2] Read OER and FER in SSR to check for errors. Receive error processing is performed in cases where a receive error occurs.

[3] Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR and compare it with this station's ID.
If the data is not this station's ID, set the MPIE bit to 1 again.
When data is read from RDR, the RDRF flag is automatically cleared to 0.

[4] Read SSR and check that the RDRF flag is set to 1, then read the data in RDR.

[5] If a receive error occurs, read the OER and FER flags in SSR to identify the error. After performing the appropriate error processing, ensure that the OER and FER flags are all cleared to 0.
Reception cannot be resumed if either of these flags is set to 1.
In the case of a framing error, a break can be detected by reading the RxD pin value.

**Figure 13.17 Sample Multiprocessor Serial Reception Flowchart (1)**

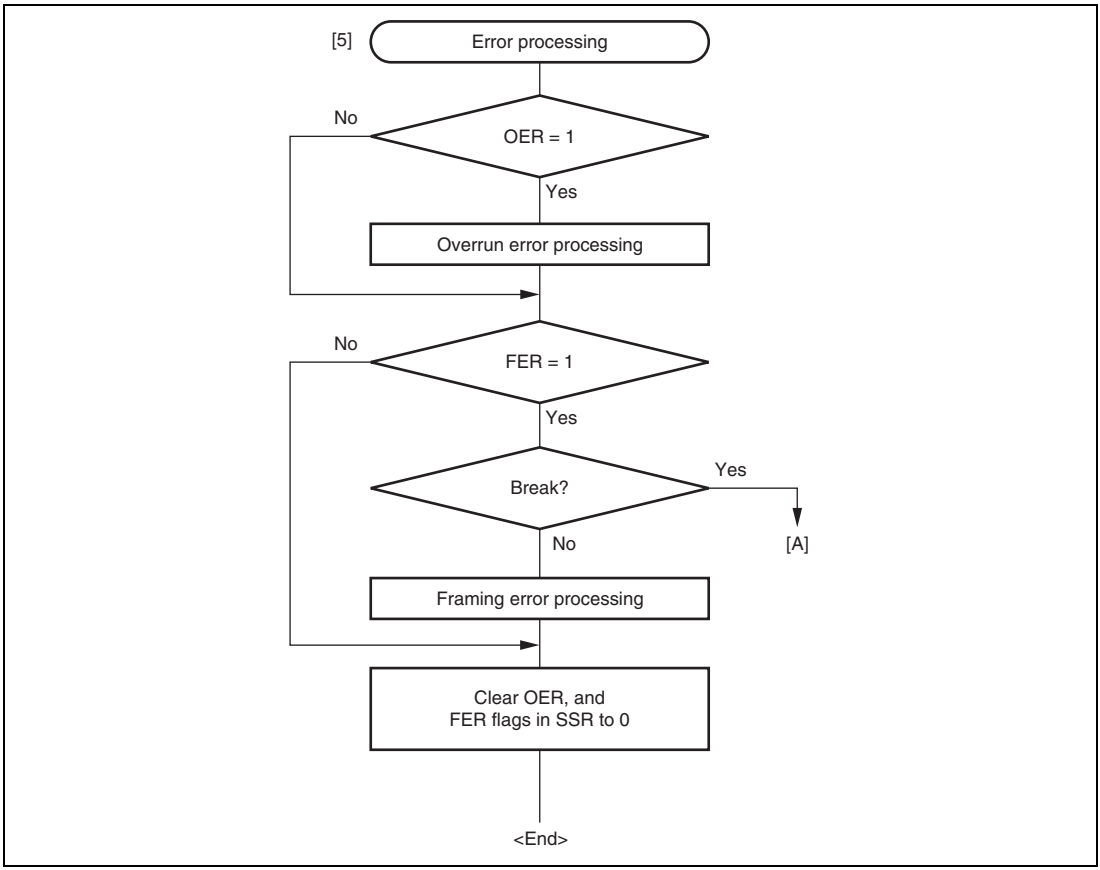



**Figure 13.17 Sample Multiprocessor Serial Reception Flowchart (2)**

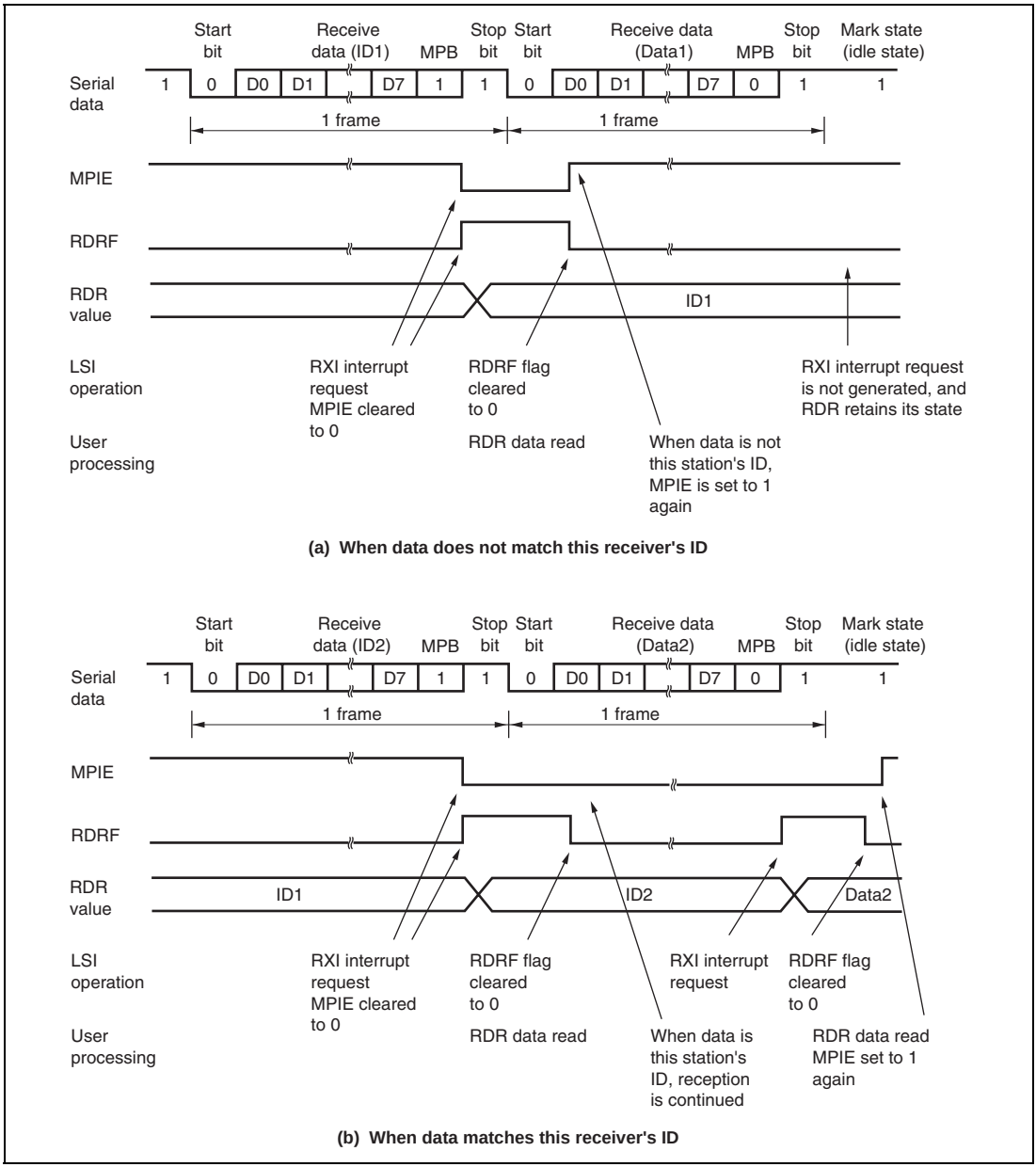



**Figure 13.18 Example of SCI3 Operation in Reception Using Multiprocessor Format (Example with 8-Bit Data, Multiprocessor Bit, One Stop Bit)**

## 13.7 Interrupts

The SCI3 creates the following six interrupt requests: transmission end, transmit data empty, receive data full, and receive errors (overrun error, framing error, and parity error). Table 13.6 shows the interrupt sources.

**Table 13.6 SCI3 Interrupt Requests**

| Interrupt Requests | Abbreviation | Interrupt Sources |
|---|---|---|
| Receive Data Full | RXI | Setting RDRF in SSR |
| Transmit Data Empty | TXI | Setting TDRE in SSR |
| Transmission End | TEI | Setting TEND in SSR |
| Receive Error | ERI | Setting OER, FER, and PER in SSR |

The initial value of the TDRE flag in SSR is 1. Thus, when the TIE bit in SCR3 is set to 1 before transferring the transmit data to TDR, a TXI interrupt request is generated even if the transmit data is not ready. The initial value of the TEND flag in SSR is 1. Thus, when the TEIE bit in SCR3 is set to 1 before transferring the transmit data to TDR, a TEI interrupt request is generated even if the transmit data has not been sent. It is possible to make use of the most of these interrupt requests efficiently by transferring the transmit data to TDR in the interrupt routine. To prevent the generation of these interrupt requests (TXI and TEI), set the enable bits (TIE and TEIE) that correspond to these interrupt requests to 1, after transferring the transmit data to TDR.

## 13.8 Usage Notes

### 13.8.1 Break Detection and Processing

When framing error detection is performed, a break can be detected by reading the RxD pin value directly. In a break, the input from the RxD pin becomes all 0, setting the FER flag, and possibly the PER flag. Note that as the SCI3 continues the receive operation after receiving a break, even if the FER flag is cleared to 0, it will be set to 1 again.

### 13.8.2 Mark State and Break Sending

When TE is 0, the TxD pin is used as an I/O port whose direction (input or output) and level are determined by PCR and PDR. This can be used to set the TxD pin to mark state (high level) or send a break during serial data transmission. To maintain the communication line at mark state until TE is set to 1, set both PCR and PDR to 1. As TE is cleared to 0 at this point, the TxD pin becomes an I/O port, and 1 is output from the TxD pin. To send a break during serial transmission, first set PCR to 1 and PDR to 0, and then clear TE to 0. When TE is cleared to 0, the transmitter is initialized regardless of the current transmission state, the TxD pin becomes an I/O port, and 0 is output from the TxD pin.

### 13.8.3 Receive Error Flags and Transmit Operations (Clocked Synchronous Mode Only)

Transmission cannot be started when a receive error flag (OER, PER, or FER) is set to 1, even if the TDRE flag is cleared to 0. Be sure to clear the receive error flags to 0 before starting transmission. Note also that receive error flags cannot be cleared to 0 even if the RE bit is cleared to 0.

### 13.8.4 Receive Data Sampling Timing and Reception Margin in Asynchronous Mode

In asynchronous mode, the SCI3 operates on a basic clock with a frequency of 16 times the transfer rate. In reception, the SCI3 samples the falling edge of the start bit using the basic clock, and performs internal synchronization. Receive data is latched internally at the rising edge of the 8th pulse of the basic clock as shown in figure 13.19.

Thus, the reception margin in asynchronous mode is given by formula (1) below.

$$M = \left\{ (0.5 - \frac{1}{2N}) - \frac{D - 0.5}{N} - (L - 0.5)\,F \right\} \times 100(\%)$$

... Formula (1)

Where 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 rate deviation

Assuming values of F (absolute value of clock rate deviation) = 0 and D (clock duty) = 0.5 in formula (1), the reception margin can be given by the formula.

$M = \{0.5 - 1/(2 \times 16)\} \times 100\ [\%] = 46.875\%$

However, this is only the computed value, and a margin of 20% to 30% should be allowed for in system design.

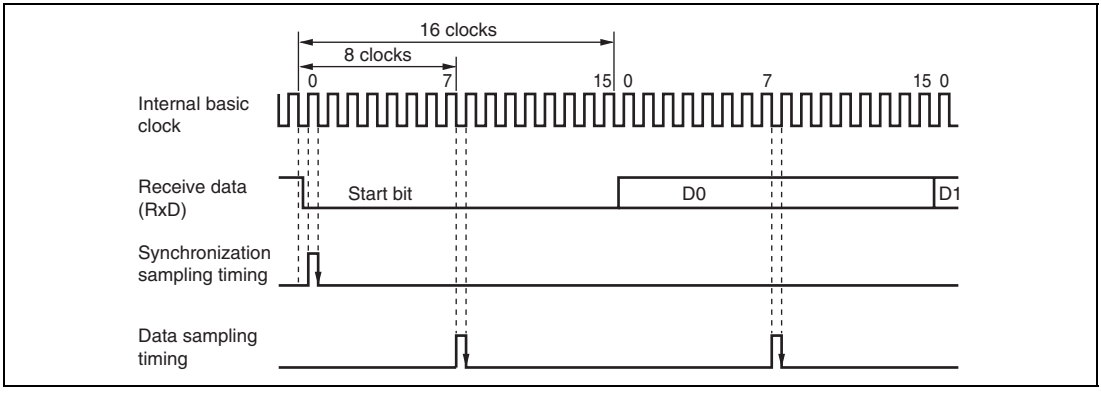



**Figure 13.19 Receive Data Sampling Timing in Asynchronous Mode**

# Section 14 A/D Converter

This LSI includes a successive approximation type 10-bit A/D converter that allows up to four analog input channels to be selected. The block diagram of the A/D converter is shown in figure 14.1.

## 14.1 Features

- 10-bit resolution
- Four input channels
- Conversion time: at least 4.4 μs per channel (at 16 MHz operation)
- Two operating modes
  - — Single mode: Single-channel A/D conversion
  - — Scan mode: Continuous A/D conversion on 1 to 4 channels
- Four data registers
  - — Conversion results are held in a 16-bit data register for each channel
- Sample and hold function
- Two conversion start methods
  - — Software
  - — External trigger signal
- Interrupt request
  - — An A/D conversion end interrupt request (ADI) can be generated

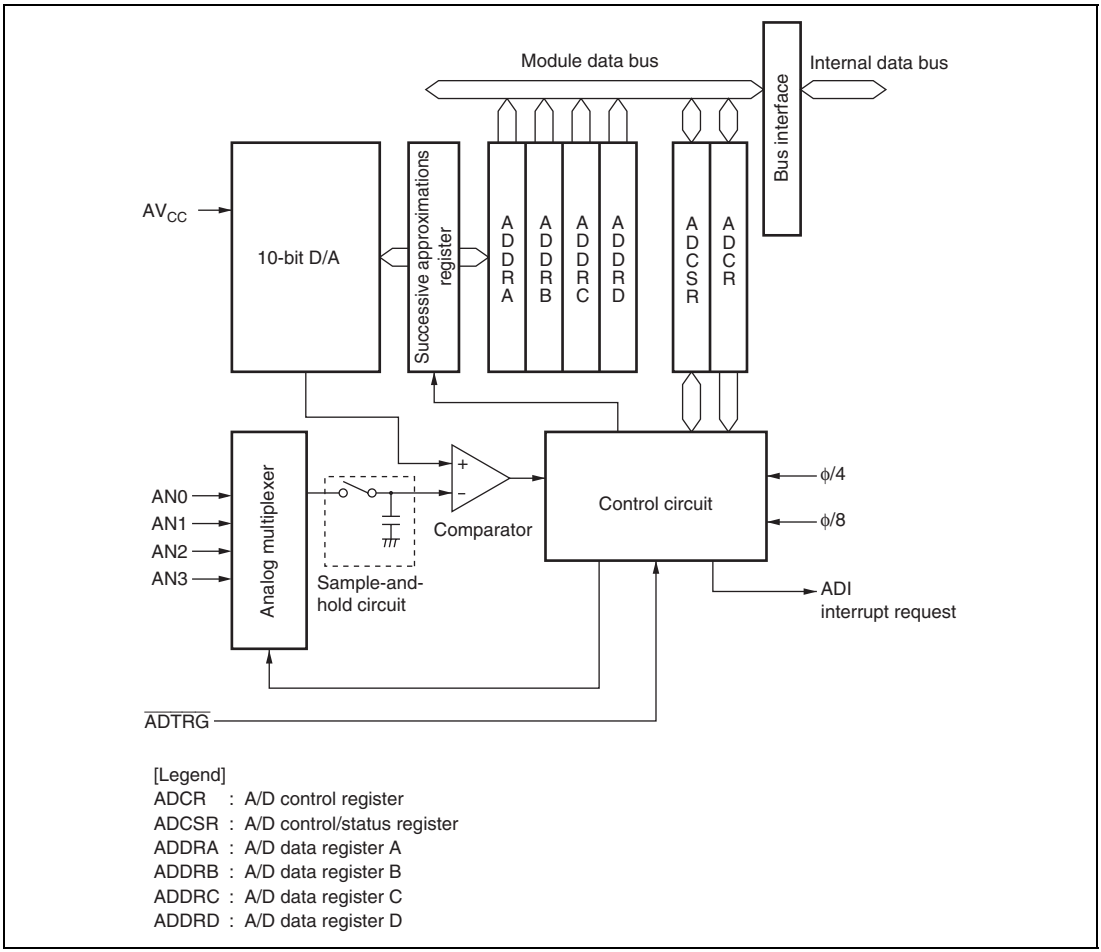



**Figure 14.1 Block Diagram of A/D Converter**

## 14.2 Input/Output Pins

Table 14.1 summarizes the input pins used by the A/D converter.

**Table 14.1 Pin Configuration**

| Pin Name | Symbol | I/O | Function |
|---|---|---|---|
| Analog power supply pin | $AV_{CC}$ | Input | Analog block power supply pin |
| Analog input pin 0 | AN0 | Input | Analog input pins |
| Analog input pin 1 | AN1 | Input | |
| Analog input pin 2 | AN2 | Input | |
| Analog input pin 3 | AN3 | Input | |
| A/D external trigger input pin | $\overline{\text{ADTRG}}$ | Input | External trigger input pin for starting A/D conversion |

## 14.3 Register Description

The A/D converter has the following registers.

- A/D data register A (ADDRA)
- A/D data register B (ADDRB)
- A/D data register C (ADDRC)
- A/D data register D (ADDRD)
- A/D control/status register (ADCSR)
- A/D control register (ADCR)

### 14.3.1 A/D Data Registers A to D (ADDRA to ADDRD)

There are four 16-bit read-only ADDR registers; ADDRA to ADDRD, used to store the results of A/D conversion. The ADDR registers, which store a conversion result for each channel, are shown in table 14.2.

The converted 10-bit data is stored in bits 6 to 15. The lower 6 bits are always read as 0.

The data bus between the CPU and the A/D converter is 8 bits wide. The upper byte can be read directly from the CPU, however the lower byte should be read via a temporary register. The temporary register contents are transferred from the ADDR when the upper byte data is read. Therefore byte access to ADDR should be done by reading the upper byte first then the lower one. Word access is also possible. ADDR is initialized to H'0000.

**Table 14.2 Analog Input Channels and Corresponding ADDR Registers**

| Analog Input Channel | A/D Data Register to Be Stored Results of A/D Conversion |
|---|---|
| AN0 | ADDRA |
| AN1 | ADDRB |
| AN2 | ADDRC |
| AN3 | ADDRD |

### 14.3.2 A/D Control/Status Register (ADCSR)

ADCSR consists of the control bits and conversion end status bits of the A/D converter.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | ADF | 0 | R/W | A/D End Flag<br>[Setting conditions]<br>• When A/D conversion ends in single mode<br>• When A/D conversion ends on all the channels selected in scan mode<br>[Clearing condition]<br>• When 0 is written after reading ADF = 1 |
| 6 | ADIE | 0 | R/W | A/D Interrupt Enable<br>A/D conversion end interrupt (ADI) request enabled by ADF when 1 is set |
| 5 | ADST | 0 | R/W | A/D Start<br>Setting this bit to 1 starts A/D conversion. In single mode, this bit is cleared to 0 automatically when conversion on the specified channel is complete. In scan mode, conversion continues sequentially on the specified channels until this bit is cleared to 0 by software, a reset, or a transition to standby mode. |
| 4 | SCAN | 0 | R/W | Scan Mode<br>Selects single mode or scan mode as the A/D conversion operating mode.<br>0: Single mode<br>1: Scan mode |
| 3 | CKS | 0 | R/W | Clock Select<br>Selects the A/D conversions time<br>0: Conversion time = 134 states (max.)<br>1: Conversion time = 70 states (max.)<br>Clear the ADST bit to 0 before switching the conversion time. |

| Bit | Bit Name | Initial Value | R/W | Description | |
|---|---|---|---|---|---|
| 2 | CH2 | 0 | R/W | Channel Select 0 to 2 | |
| 1 | CH1 | 0 | R/W | Select analog input channels. | |
| 0 | CH0 | 0 | R/W | When SCAN = 0 | When SCAN = 1 |
| | | | | X00: AN0 | X00: AN0 |
| | | | | X01: AN1 | X01: AN0 to AN1 |
| | | | | X10: AN2 | X10: AN0 to AN2 |
| | | | | X11: AN3 | X11: AN0 to AN3 |

Legend: X: Don't care.

### 14.3.3 A/D Control Register (ADCR)

ADCR enables A/D conversion started by an external trigger signal.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TRGE | 0 | R/W | Trigger Enable<br>A/D conversion is started at the falling edge and the rising edge of the external trigger signal ($\overline{\text{ADTRG}}$) when this bit is set to 1.<br>The selection between the falling edge and rising edge of the external trigger pin ($\overline{\text{ADTRG}}$) conforms to the WPEG5 bit in the interrupt edge select register 2 (IEGR2) |
| 6 to 1 | — | All 1 | — | Reserved<br>These bits are always read as 1. |
| 0 | — | 0 | R/W | Reserved<br>Do not set this bit to 1, though the bit is readable/writable. |

## 14.4 Operation

The A/D converter operates by successive approximation with 10-bit resolution. It has two operating modes; single mode and scan mode. When changing the operating mode or analog input channel, in order to prevent incorrect operation, first clear the bit ADST to 0 in ADCSR. The ADST bit can be set at the same time as the operating mode or analog input channel is changed.

### 14.4.1 Single Mode

In single mode, A/D conversion is performed once for the analog input on the specified single channel as follows:

1. A/D conversion is started from the first channel when the ADST bit in ADCSR is set to 1, according to software or external trigger input.
2. When A/D conversion is completed, the result is transferred to the corresponding A/D data register to the channel.
3. On completion of conversion, the ADF bit in ADCSR is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt request is generated.
4. The ADST bit remains set to 1 during A/D conversion. When A/D conversion ends, the ADST bit is automatically cleared to 0 and the A/D converter enters the wait state.

### 14.4.2 Scan Mode

In scan mode, A/D conversion is performed sequentially for the analog input on the specified channels (four channels maximum) as follows:

1. When the ADST bit is set to 1 by software, or external trigger input, A/D conversion starts on the first channel in the group.
2. When A/D conversion for each channel is completed, the result is sequentially transferred to the A/D data register corresponding to each channel.
3. When conversion of all the selected channels is completed, the ADF flag in ADCSR is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt is requested. Conversion of the first channel in the group starts again.
4. The ADST bit is not automatically cleared to 0. Steps [2] to [3] are repeated as long as the ADST bit remains set to 1. When the ADST bit is cleared to 0, A/D conversion stops.

### 14.4.3 Input Sampling and A/D Conversion Time

The A/D converter has a built-in sample-and-hold circuit. The A/D converter samples the analog input when the A/D conversion start delay time ($t_D$) has passed after the ADST bit is set to 1, then starts conversion. Figure 14.2 shows the A/D conversion timing. Table 14.3 shows the A/D conversion time.

As indicated in figure 14.2, the A/D conversion time includes $t_D$ and the input sampling time. The length of $t_D$ varies depending on the timing of the write access to ADCSR. The total conversion time therefore varies within the ranges indicated in table 14.3.

In scan mode, the values given in table 14.3 apply to the first conversion time. In the second and subsequent conversions, the conversion time is 128 states (fixed) when CKS = 0 and 66 states (fixed) when CKS = 1.

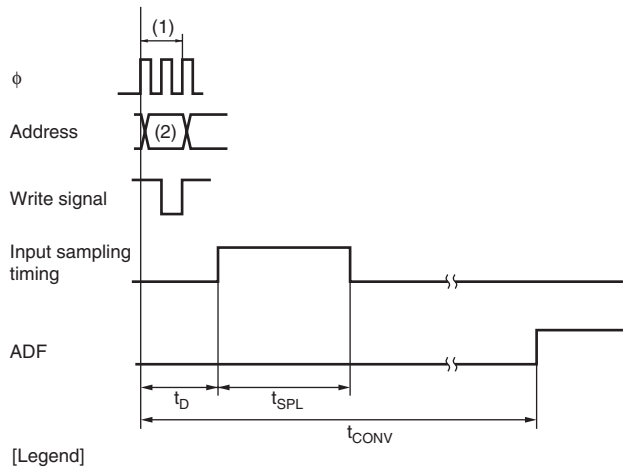



**Figure 14.2 A/D Conversion Timing**

**Table 14.3 A/D Conversion Time (Single Mode)**

| | | CKS = 0 | | | CKS = 1 | | |
|---|---|---|---|---|---|---|---|
| **Item** | **Symbol** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** |
| A/D conversion start delay | $t_D$ | 6 | — | 9 | 4 | — | 5 |
| Input sampling time | $t_{SPL}$ | — | 31 | — | — | 15 | — |
| A/D conversion time | $t_{CONV}$ | 131 | — | 134 | 69 | — | 70 |

Note: All values represent the number of states.

### 14.4.4 External Trigger Input Timing

A/D conversion can also be started by an external trigger input. When the TRGE bit is set to 1 in ADCR, external trigger input is enabled at the $\overline{\text{ADTRG}}$ pin. A falling edge at the $\overline{\text{ADTRG}}$ input pin sets the ADST bit to 1 in ADCSR, starting A/D conversion. Other operations, in both single and scan modes, are the same as when the bit ADST has been set to 1 by software. Figure 14.3 shows the timing.

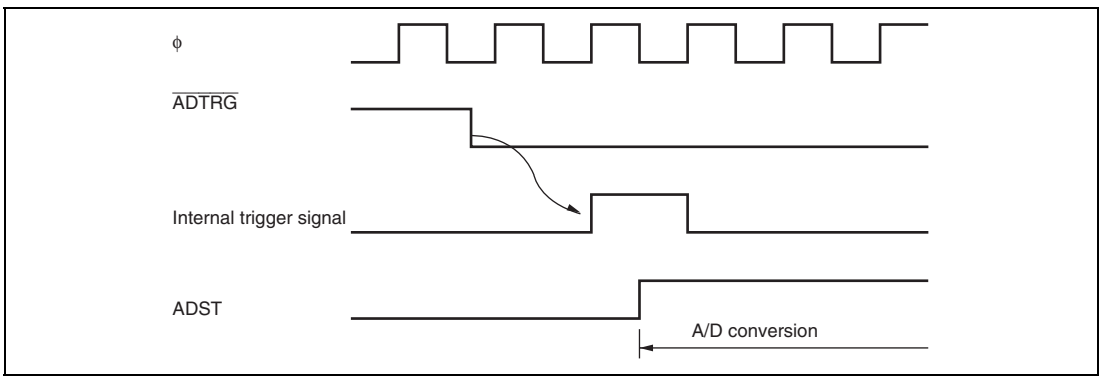



**Figure 14.3 External Trigger Input Timing**

## 14.5 A/D Conversion Accuracy Definitions

This LSI's A/D conversion accuracy definitions are given below.

- Resolution

  The number of A/D converter digital output codes
- Quantization error

  The deviation inherent in the A/D converter, given by 1/2 LSB (see figure 14.4).
- Offset error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from the minimum voltage value 0000000000 to 0000000001 (see figure 14.5).
- Full-scale error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from 1111111110 to 1111111111 (see figure 14.5).
- Nonlinearity error

  The error with respect to the ideal A/D conversion characteristics between zero voltage and full-scale voltage. Does not include offset error, full-scale error, or quantization error.
- Absolute accuracy

  The deviation between the digital value and the analog input value. Includes offset error, full-scale error, quantization error, and nonlinearity error.

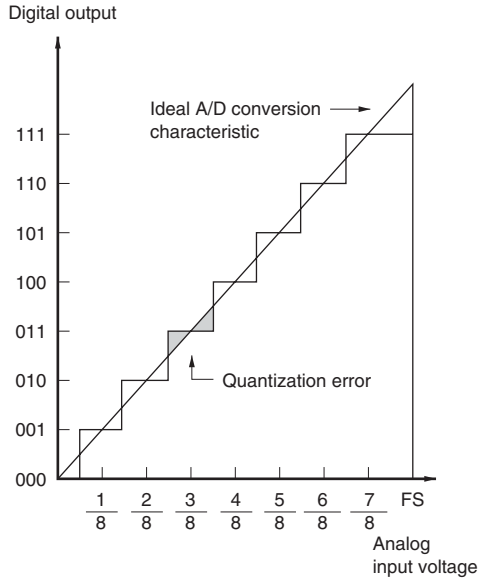



**Figure 14.4 A/D Conversion Accuracy Definitions (1)**

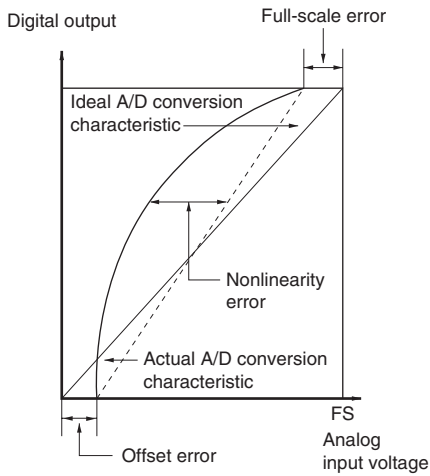



**Figure 14.5 A/D Conversion Accuracy Definitions (2)**

## 14.6 Usage Notes

### 14.6.1 Permissible Signal Source Impedance

This LSI's analog input is designed such that conversion accuracy is guaranteed for an input signal for which the signal source impedance is 5 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 5 kΩ, charging may be insufficient and it may not be possible to guarantee A/D conversion accuracy. However, for A/D conversion in single mode with 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 14.6). When converting a high-speed analog signal or converting in scan mode, a low-impedance buffer should be inserted.

### 14.6.2 Influences on Absolute Accuracy

Adding capacitance results in coupling with GND, and therefore noise in GND may adversely affect absolute accuracy. 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.

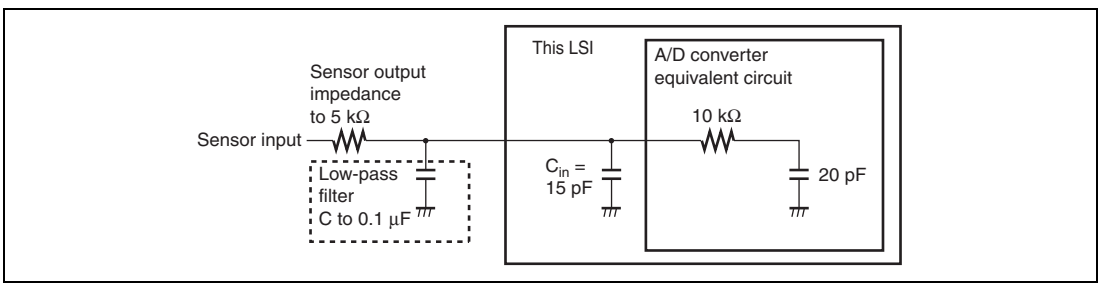



**Figure 14.6 Analog Input Circuit Example**

# Section 15 Power Supply Circuit

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, 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.

## 15.1 When Using Internal Power Supply Step-Down Circuit

Connect the external power supply to the $V_{CC}$ pin, and connect a capacitance of approximately 0.1 µF between $V_{CL}$ and $V_{SS}$, as shown in figure 15.1. 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 A/D converter analog power supply is not affected by the internal step-down circuit.

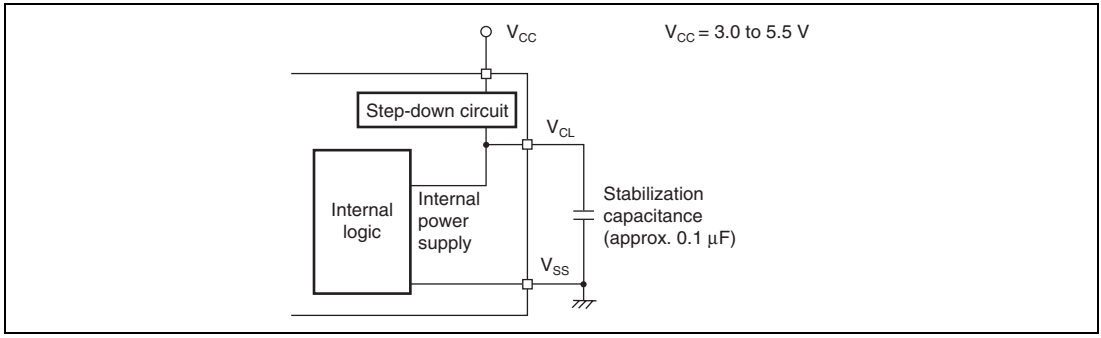



**Figure 15.1 Power Supply Connection when Internal Step-Down Circuit is Used**

## 15.2 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 $V_{CL}$ pin and $V_{CC}$ pin, as shown in figure 15.2. The external power supply is then input directly to the internal power supply. The permissible range for the power supply voltage is 3.0 V to 3.6 V. Operation cannot be guaranteed if a voltage outside this range (less than 3.0 V or more than 3.6 V) is input.

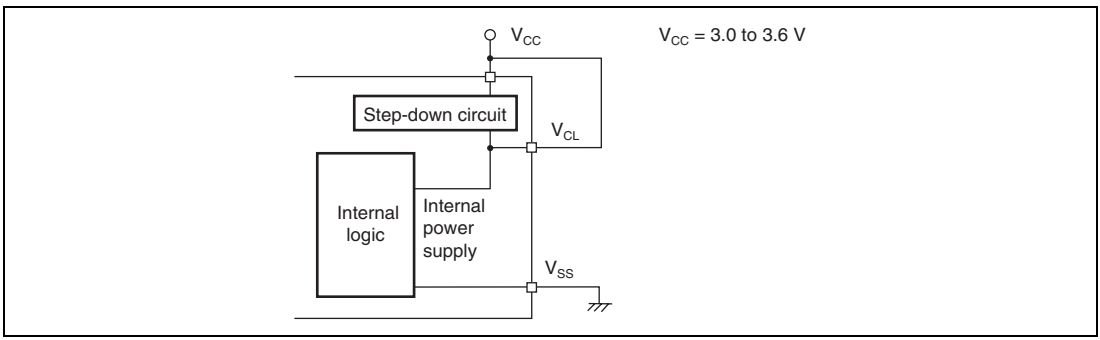



**Figure 15.2 Power Supply Connection when Internal Step-Down Circuit is Not Used**

# Section 16 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.

## 16.1 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 |
|---|---|---|---|---|---|---|
| Timer mode register W | TMRW | 8 | H'FF80 | Timer W | 8 | 2 |
| Timer control register W | TCRW | 8 | H'FF81 | Timer W | 8 | 2 |
| Timer interrupt enable register W | TIERW | 8 | H'FF82 | Timer W | 8 | 2 |
| Timer status register W | TSRW | 8 | H'FF83 | Timer W | 8 | 2 |
| Timer I/O control register 0 | TIOR0 | 8 | H'FF84 | Timer W | 8 | 2 |
| Timer I/O control register 1 | TIOR1 | 8 | H'FF85 | Timer W | 8 | 2 |
| Timer counter | TCNT | 16 | H'FF86 | Timer W | 16*[1] | 2 |
| General register A | GRA | 16 | H'FF88 | Timer W | 16*[1] | 2 |
| General register B | GRB | 16 | H'FF8A | Timer W | 16*[1] | 2 |
| General register C | GRC | 16 | H'FF8C | Timer W | 16*[1] | 2 |
| General register D | GRD | 16 | H'FF8E | Timer W | 16*[1] | 2 |
| Flash memory control register 1 | FLMCR1 | 8 | H'FF90 | ROM | 8 | 2 |
| Flash memory control register 2 | FLMCR2 | 8 | H'FF91 | ROM | 8 | 2 |
| Erase block register 1 | EBR1 | 8 | H'FF93 | ROM | 8 | 2 |
| Flash memory enable register | FENR | 8 | H'FF9B | ROM | 8 | 2 |
| Timer control register V0 | TCRV0 | 8 | H'FFA0 | Timer V | 8 | 3 |
| Timer control/status register V | TCSRV | 8 | H'FFA1 | Timer V | 8 | 3 |
| Timer constant register A | TCORA | 8 | H'FFA2 | Timer V | 8 | 3 |
| Timer constant register B | TCORB | 8 | H'FFA3 | Timer V | 8 | 3 |
| Timer counter V | TCNTV | 8 | H'FFA4 | Timer V | 8 | 3 |
| Timer control register V1 | TCRV1 | 8 | H'FFA5 | Timer V | 8 | 3 |
| Serial mode register | SMR | 8 | H'FFA8 | SCI3 | 8 | 3 |
| Bit rate register | BRR | 8 | H'FFA9 | SCI3 | 8 | 3 |
| Serial control register 3 | SCR3 | 8 | H'FFAA | SCI3 | 8 | 3 |

| Register Name | Abbre-viation | Bit No | Address | Module Name | Data Bus Width | Access State |
|---|---|---|---|---|---|---|
| Transmit data register | TDR | 8 | H'FFAB | SCI3 | 8 | 3 |
| Serial status register | SSR | 8 | H'FFAC | SCI3 | 8 | 3 |
| Receive data register | RDR | 8 | H'FFAD | SCI3 | 8 | 3 |
| A/D data register A | ADDRA | 16 | H'FFB0 | A/D converter | 8 | 3 |
| A/D data register B | ADDRB | 16 | H'FFB2 | A/D converter | 8 | 3 |
| A/D data register C | ADDRC | 16 | H'FFB4 | A/D converter | 8 | 3 |
| A/D data register D | ADDRD | 16 | H'FFB6 | A/D converter | 8 | 3 |
| A/D control/status register | ADCSR | 8 | H'FFB8 | A/D converter | 8 | 3 |
| A/D control register | ADCR | 8 | H'FFB9 | A/D converter | 8 | 3 |
| Timer control/status register WD | TCSRWD | 8 | H'FFC0 | WDT*[2] | 8 | 2 |
| Timer counter WD | TCWD | 8 | H'FFC1 | WDT*[2] | 8 | 2 |
| Timer mode register WD | TMWD | 8 | H'FFC2 | WDT*[2] | 8 | 2 |
| Address break control register | ABRKCR | 8 | H'FFC8 | Address break | 8 | 2 |
| Address break status register | ABRKSR | 8 | H'FFC9 | Address break | 8 | 2 |
| Break address register H | BARH | 8 | H'FFCA | Address break | 8 | 2 |
| Break address register L | BARL | 8 | H'FFCB | Address break | 8 | 2 |
| Break data register H | BDRH | 8 | H'FFCC | Address break | 8 | 2 |
| Break data register L | BDRL | 8 | H'FFCD | Address break | 8 | 2 |
| Port pull-up control register 1 | PUCR1 | 8 | H'FFD0 | I/O port | 8 | 2 |
| Port pull-up control register 5 | PUCR5 | 8 | H'FFD1 | I/O port | 8 | 2 |
| Port data register 1 | PDR1 | 8 | H'FFD4 | I/O port | 8 | 2 |
| Port data register 2 | PDR2 | 8 | H'FFD5 | I/O port | 8 | 2 |
| Port data register 5 | PDR5 | 8 | H'FFD8 | 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 |

| Register Name | Abbre-viation | Bit No | Address | Module Name | Data Bus Width | Access State |
|---|---|---|---|---|---|---|
| Port data register B | PDRB | 8 | H'FFDD | I/O port | 8 | 2 |
| Port mode register 1 | PMR1 | 8 | H'FFE0 | I/O port | 8 | 2 |
| Port mode register 5 | PMR5 | 8 | H'FFE1 | I/O port | 8 | 2 |
| Port control register 1 | PCR1 | 8 | H'FFE4 | I/O port | 8 | 2 |
| Port control register 2 | PCR2 | 8 | H'FFE5 | I/O port | 8 | 2 |
| Port control register 5 | PCR5 | 8 | H'FFE8 | 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 |
| System control register 1 | SYSCR1 | 8 | H'FFF0 | Power-down | 8 | 2 |
| System control register 2 | SYSCR2 | 8 | H'FFF1 | Power-down | 8 | 2 |
| Interrupt edge select register 1 | IEGR1 | 8 | H'FFF2 | Interrupts | 8 | 2 |
| Interrupt edge select register 2 | IEGR2 | 8 | H'FFF3 | Interrupts | 8 | 2 |
| Interrupt enable register 1 | IENR1 | 8 | H'FFF4 | Interrupts | 8 | 2 |
| Interrupt flag register 1 | IRR1 | 8 | H'FFF6 | Interrupts | 8 | 2 |
| Wake-up interrupt flag register | IWPR | 8 | H'FFF8 | Interrupts | 8 | 2 |
| Module standby control register 1 | MSTCR1 | 8 | H'FFF9 | Power-down | 8 | 2 |

Notes: 1. Only word access can be used.
2. WDT: Watchdog timer

## 16.2 Register Bits

Register bit names of the on-chip peripheral modules are described below.

Each line covers eight bits, and 16-bit registers are shown as 2 lines.

| Register Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| TMRW | CTS | — | BUFEB | BUFEA | — | PWMD | PWMC | PWMB | Timer W |
| TCRW | CCLR | CKS2 | CKS1 | CKS0 | TOD | TOC | TOB | TOA | |
| TIERW | OVIE | — | — | — | IMIED | IMIEC | IMIEB | IMIEA | |
| TSRW | OVF | — | — | — | IMFD | IMFC | IMFB | IMFA | |
| TIOR0 | — | IOB2 | IOB1 | IOB0 | — | IOA2 | IOA1 | IOA0 | |
| TIOR1 | — | IOD2 | IOD1 | IOD0 | — | IOC2 | IOC1 | IOC0 | |
| TCNT | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 | |
| | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 | |
| GRA | GRA15 | GRA14 | GRA13 | GRA12 | GRA11 | GRA10 | GRA9 | GRA8 | |
| | GRA7 | GRA6 | GRA5 | GRA4 | GRA3 | GRA2 | GRA1 | GRA0 | |
| GRB | GRB15 | GRB14 | GRB13 | GRB12 | GRB11 | GRB10 | GRB9 | GRB8 | |
| | GRB7 | GRB6 | GRB5 | GRB4 | GRB3 | GRB2 | GRB1 | GRB0 | |
| GRC | GRC15 | GRC14 | GRC13 | GRC12 | GRC11 | GRC10 | GRC9 | GRC8 | |
| | GRC7 | GRC6 | GRC5 | GRC4 | GRC3 | GRC2 | GRC1 | GRC0 | |
| GRD | GRD15 | GRD14 | GRD13 | GRD12 | GRD11 | GRD10 | GRD9 | GRD8 | |
| | GRD7 | GRD6 | GRD5 | GRD4 | GRD3 | GRD2 | GRD1 | GRD0 | |
| FLMCR1 | — | SWE | ESU | PSU | EV | PV | E | P | ROM |
| FLMCR2 | FLER | — | — | — | — | — | — | — | |
| EBR1 | — | — | — | EB4 | EB3 | EB2 | EB1 | EB0 | |
| FENR | FLSHE | — | — | — | — | — | — | — | |
| TCRV0 | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | Timer V |
| TCSRV | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 | |
| TCORA | TCORA7 | TCORA6 | TCORA5 | TCORA4 | TCORA3 | TCORA2 | TCORA1 | TCORA0 | |
| TCORB | TCORB7 | TCORB6 | TCORB5 | TCORB4 | TCORB3 | TCORB2 | TCORB1 | TCORB0 | |
| TCNTV | TCNTV7 | TCNTV6 | TCNTV5 | TCNTV4 | TCNTV3 | TCNTV2 | TCNTV1 | TCNTV0 | |
| TCRV1 | — | — | — | TVEG1 | TVEG0 | TRGE | — | ICKS0 | |
| SMR | COM | CHR | PE | PM | STOP | MP | CKS1 | CKS0 | SCI3 |
| BRR | BRR7 | BRR6 | BRR5 | BRR4 | BRR3 | BRR2 | BRR1 | BRR0 | |

| Register Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| SCR3 | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 | SCI3 |
| TDR | TDR7 | TDR6 | TDR5 | TDR4 | TDR3 | TDR2 | TDR1 | TDR0 | |
| SSR | TDRE | RDRF | OER | FER | PER | TEND | MPBR | MPBT | |
| RDR | RDR7 | RDR6 | RDR5 | RDR4 | RDR3 | RDR2 | RDR1 | RDR0 | |
| ADDRA | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | A/D converter |
| | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRB | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRC | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRD | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| | AD1 | AD0 | — | — | — | — | — | — | |
| ADCSR | ADF | ADIE | ADST | SCAN | CKS | CH2 | CH1 | CH0 | |
| ADCR | TRGE | — | — | — | — | — | — | — | |
| TCSRWD | B6WI | TCWE | B4WI | TCSRWE | B2WI | WDON | B0WI | WRST | WDT* |
| TCWD | TCWD7 | TCWD6 | TCWD5 | TCWD4 | TCWD3 | TCWD2 | TCWD1 | TCWD0 | |
| TMWD | — | — | — | — | CKS3 | CKS2 | CKS1 | CKS0 | |
| ABRKCR | RTINTE | CSEL1 | CSEL0 | ACMP2 | ACMP1 | ACMP0 | DCMP1 | DCMP0 | Address break |
| ABRKSR | ABIF | ABIE | — | — | — | — | — | — | |
| BARH | BARH7 | BARH6 | BARH5 | BARH4 | BARH3 | BARH2 | BARH1 | BARH0 | |
| BARL | BARL7 | BARL6 | BARL5 | BARL4 | BARL3 | BARL2 | BARL1 | BARL0 | |
| BDRH | BDRH7 | BDRH6 | BDRH5 | BDRH4 | BDRH3 | BDRH2 | BDRH1 | BDRH0 | |
| BDRL | BDRL7 | BDRL6 | BDRL5 | BDRL4 | BDRL3 | BDRL2 | BDRL1 | BDRL0 | |
| PUCR1 | PUCR17 | PUCR16 | PUCR15 | PUCR14 | — | PUCR12 | PUCR11 | PUCR10 | I/O port |
| PUCR5 | — | — | PUCR55 | PUCR54 | PUCR53 | PUCR52 | PUCR51 | PUCR50 | |
| PDR1 | P17 | P16 | P15 | P14 | — | P12 | P11 | P10 | |
| PDR2 | — | — | — | — | — | P22 | P21 | P20 | |
| PDR5 | P57 | P56 | P55 | P54 | P53 | P52 | P51 | P50 | |
| PDR7 | — | P76 | P75 | P74 | — | — | — | — | |
| PDR8 | — | — | — | P84 | P83 | P82 | P81 | P80 | |

| Register Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module Name |
|---|---|---|---|---|---|---|---|---|---|
| PDRB | — | — | — | — | PB3 | PB2 | PB1 | PB0 | I/O port |
| PMR1 | IRQ3 | — | — | IRQ0 | — | — | TXD | — | |
| PMR5 | POF7 | POF6 | WKP5 | WKP4 | WKP3 | WKP2 | WKP1 | WKP0 | |
| PCR1 | PCR17 | PCR16 | PCR15 | PCR14 | — | PCR12 | PCR11 | PCR10 | |
| PCR2 | — | — | — | — | — | PCR22 | PCR21 | PCR20 | |
| PCR5 | PCR57 | PCR56 | PCR55 | PCR54 | PCR53 | PCR52 | PCR51 | PCR50 | |
| PCR7 | — | PCR76 | PCR75 | PCR74 | — | — | — | — | |
| PCR8 | — | — | — | PCR84 | PCR83 | PCR82 | PCR81 | PCR80 | |
| SYSCR1 | SSBY | STS2 | STS1 | STS0 | — | — | — | — | Power-down |
| SYSCR2 | SMSEL | — | DTON | MA2 | MA1 | MA0 | — | — | |
| IEGR1 | — | — | — | — | IEG3 | — | — | IEG0 | Interrupts |
| IEGR2 | — | — | WPEG5 | WPEG4 | WPEG3 | WPEG2 | WPEG1 | WPEG0 | |
| IENR1 | IENDT | — | IENWP | — | IEN3 | — | — | IEN0 | |
| IRR1 | IRRDT | — | — | — | IRRI3 | — | — | IRRI0 | |
| IWPR | — | — | IWPF5 | IWPF4 | IWPF3 | IWPF2 | IWPF1 | IWPF0 | |
| MSTCR1 | — | — | MSTS3 | MSTAD | MSTWD | MSTTW | MSTTV | — | Power-down |

Note: * WDT: Watchdog timer

## 16.3 Register States in Each Operating Mode

| Register Name | Reset | Active | Sleep | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|
| TMRW | Initialized | — | — | — | — | Timer W |
| TCRW | Initialized | — | — | — | — | |
| TIERW | Initialized | — | — | — | — | |
| TSRW | Initialized | — | — | — | — | |
| TIOR0 | Initialized | — | — | — | — | |
| TIOR1 | Initialized | — | — | — | — | |
| TCNT | Initialized | — | — | — | — | |
| GRA | Initialized | — | — | — | — | |
| GRB | Initialized | — | — | — | — | |
| GRC | Initialized | — | — | — | — | |
| GRD | Initialized | — | — | — | — | |
| FLMCR1 | Initialized | — | — | Initialized | Initialized | ROM |
| FLMCR2 | Initialized | — | — | — | — | |
| EBR1 | Initialized | — | — | Initialized | Initialized | |
| FENR | Initialized | — | — | — | — | |
| TCRV0 | Initialized | — | — | Initialized | Initialized | Timer V |
| TCSRV | Initialized | — | — | Initialized | Initialized | |
| TCORA | Initialized | — | — | Initialized | Initialized | |
| TCORB | Initialized | — | — | Initialized | Initialized | |
| TCNTV | Initialized | — | — | Initialized | Initialized | |
| TCRV1 | Initialized | — | — | Initialized | Initialized | |
| SMR | Initialized | — | — | Initialized | Initialized | SCI3 |
| BRR | Initialized | — | — | Initialized | Initialized | |
| SCR3 | Initialized | — | — | Initialized | Initialized | |
| TDR | Initialized | — | — | Initialized | Initialized | |
| SSR | Initialized | — | — | Initialized | Initialized | |
| RDR | Initialized | — | — | Initialized | Initialized | |

| Register Name | Reset | Active | Sleep | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|
| ADDRA | Initialized | — | — | Initialized | Initialized | A/D converter |
| ADDRB | Initialized | — | — | Initialized | Initialized | |
| ADDRC | Initialized | — | — | Initialized | Initialized | |
| ADDRD | Initialized | — | — | Initialized | Initialized | |
| ADCSR | Initialized | — | — | Initialized | Initialized | |
| ADCR | Initialized | — | — | Initialized | Initialized | |
| TCSRWD | Initialized | — | — | — | — | WDT* |
| TCWD | Initialized | — | — | — | — | |
| TMWD | Initialized | — | — | — | — | |
| ABRKCR | Initialized | — | — | — | — | Address Break |
| ABRKSR | Initialized | — | — | — | — | |
| BARH | Initialized | — | — | — | — | |
| BARL | Initialized | — | — | — | — | |
| BDRH | Initialized | — | — | — | — | |
| BDRL | Initialized | — | — | — | — | |
| PUCR1 | Initialized | — | — | — | — | I/O port |
| PUCR5 | Initialized | — | — | — | — | |
| PDR1 | Initialized | — | — | — | — | |
| PDR2 | Initialized | — | — | — | — | |
| PDR5 | Initialized | — | — | — | — | |
| PDR7 | Initialized | — | — | — | — | |
| PDR8 | Initialized | — | — | — | — | |
| PDRB | Initialized | — | — | — | — | |
| PMR1 | Initialized | — | — | — | — | |
| PMR5 | Initialized | — | — | — | — | |
| PCR1 | Initialized | — | — | — | — | |
| PCR2 | Initialized | — | — | — | — | |
| PCR5 | Initialized | — | — | — | — | |
| PCR7 | Initialized | — | — | — | — | |
| PCR8 | Initialized | — | — | — | — | |

| Register Name | Reset | Active | Sleep | Subsleep | Standby | Module |
|---|---|---|---|---|---|---|
| SYSCR1 | Initialized | — | — | — | — | Power-down |
| SYSCR2 | Initialized | — | — | — | — | |
| IEGR1 | Initialized | — | — | — | — | Interrupts |
| IEGR2 | Initialized | — | — | — | — | |
| IENR1 | Initialized | — | — | — | — | |
| IRR1 | Initialized | — | — | — | — | |
| IWPR | Initialized | — | — | — | — | |
| MSTCR1 | Initialized | — | — | — | — | Power-down |

Notes: — is not initialized

* WDT: Watchdog timer

# Section 17 Electrical Characteristics

## 17.1 Absolute Maximum Ratings

**Table 17.1 Absolute Maximum Ratings**

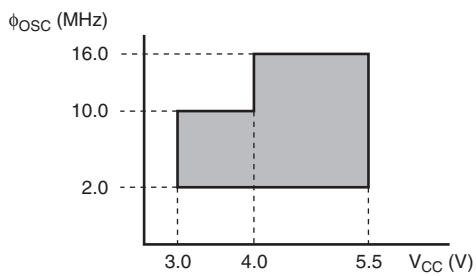
| Item | | Symbol | Value | Unit | Note |
|---|---|---|---|---|---|
| Power supply voltage | | $V_{CC}$ | –0.3 to +7.0 | V | * |
| Analog power supply voltage | | $AV_{CC}$ | –0.3 to +7.0 | V | |
| Input voltage | Ports other than Port B | $V_{IN}$ | –0.3 to $V_{CC}$ +0.3 | V | |
| | Port B | | –0.3 to $AV_{CC}$ +0.3 | V | |
| Operating temperature | | $T_{opr}$ | –20 to +75 | °C | |
| Storage temperature | | $T_{stg}$ | –55 to +125 | °C | |

Note: * 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.

## 17.2 Electrical Characteristics

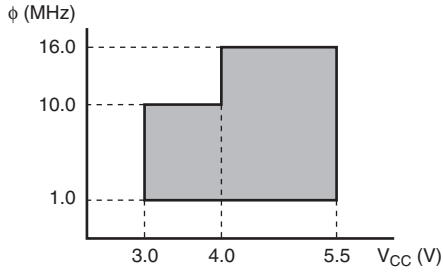
### 17.2.1 Power Supply Voltage and Operating Ranges

**Power Supply Voltage and Oscillation Frequency Range**

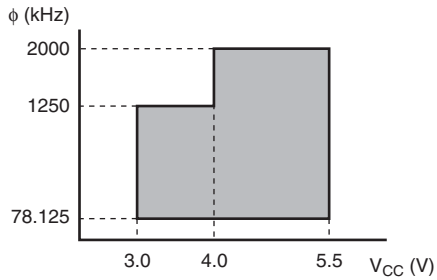



- $AV_{CC}$ = 3.3 V to 5.5 V
- Active mode
- Sleep mode

## Power Supply Voltage and Operating Frequency Range

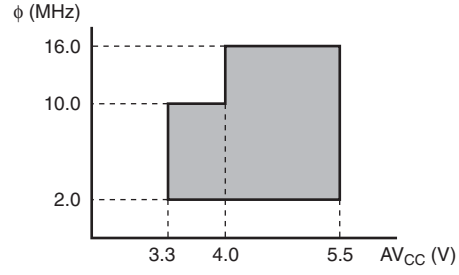



- $AV_{CC}$ = 3.3 V to 5.5 V
- Active mode
- Sleep mode
  (When MA2 = 0 in SYSCR2)




- $AV_{CC}$ = 3.3 V to 5.5 V
- Active mode
- Sleep mode
  (When MA2 = 1 in SYSCR2)

## Analog Power Supply Voltage and A/D Converter Accuracy Guarantee Range




- $V_{CC}$ = 3.0 V to 5.5 V
- Active mode
- Sleep mode

### 17.2.2 DC Characteristics

**Table 17.2 DC Characteristics (1)**

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C unless otherwise indicated.

| | | | | Values | | | | |
|---|---|---|---|---|---|---|---|---|
| **Item** | **Symbol** | **Applicable Pins** | **Test Condition** | **Min** | **Typ** | **Max** | **Unit** | **Notes** |
| Input high voltage | $V_{IH}$ | $\overline{RES}$, $\overline{NMI}$, $\overline{WKP0}$ to $\overline{WKP5}$, $\overline{IRQ0}$, $\overline{IRQ3}$, $\overline{ADTRG}$,TMRIV, TMCIV, FTCI, FTIOA to FTIOD, SCK3, TRGV | $V_{CC}$ = 4.0 V to 5.5 V | $V_{CC}\times 0.8$ | — | $V_{CC}$ + 0.3 | V | |
| | | | | $V_{CC}\times 0.9$ | — | $V_{CC}$ + 0.3 | | |
| | | RXD, P12 to P10, P17 to P14, P22 to P20, P57 to P50, P76 to P74, P84 to P80 | $V_{CC}$ = 4.0 V to 5.5 V | $V_{CC}\times 0.7$ | — | $V_{CC}$ + 0.3 | V | |
| | | | | $V_{CC}\times 0.8$ | — | $V_{CC}$ + 0.3 | | |
| | | PB3 to PB0 | $V_{CC}$ = 4.0 V to 5.5 V | $V_{CC}\times 0.7$ | — | $AV_{CC}$ + 0.3 | V | |
| | | | | $V_{CC}\times 0.8$ | — | $AV_{CC}$ + 0.3 | | |
| | | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | $V_{CC}$ – 0.5 | — | $V_{CC}$ + 0.3 | V | |
| | | | | $V_{CC}$ – 0.3 | — | $V_{CC}$ + 0.3 | | |
| Input low voltage | $V_{IL}$ | $\overline{RES}$, $\overline{NMI}$, $\overline{WKP0}$ to $\overline{WKP5}$, $\overline{IRQ0}$, $\overline{IRQ3}$, $\overline{ADTRG}$,TMRIV, TMCIV, FTCI, FTIOA to FTIOD, SCK3, TRGV | $V_{CC}$ = 4.0 V to 5.5 V | –0.3 | — | $V_{CC}\times 0.2$ | V | |
| | | | | –0.3 | — | $V_{CC}\times 0.1$ | | |
| | | RXD, P12 to P10, P17 to P14, P22 to P20, P57 to P50, P76 to P74, P84 to P80 PB3 to PB0 | $V_{CC}$ = 4.0 V to 5.5 V | –0.3 | — | $V_{CC}\times 0.3$ | V | |
| | | | | –0.3 | — | $V_{CC}\times 0.2$ | | |
| | | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | –0.3 | — | 0.5 | V | |
| | | | | –0.3 | — | 0.3 | | |

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Notes |
|---|---|---|---|---|---|---|---|---|
| Output high voltage | $V_{OH}$ | P12 to P10, P17 to P14, P22 to P20, P57 to P50, P76 to P74, P84 to P80 | $V_{CC}$ = 4.0 V to 5.5 V<br>$-I_{OH}$ = 1.5 mA | $V_{CC}$ – 1.0 | — | — | V | |
| | | | $-I_{OH}$ = 0.1 mA | $V_{CC}$ – 0.5 | — | — | | |
| Output low voltage | $V_{OL}$ | P12 to P10, P17 to P14, P22 to P20, P55 to P50, P76 to P74 | $V_{CC}$ = 4.0 V to 5.5 V<br>$I_{OL}$ = 1.6 mA | — | — | 0.6 | V | |
| | | | $I_{OL}$ = 0.4 mA | — | — | 0.4 | | |
| | | P84 to P80 | $V_{CC}$ = 4.0 V to 5.5 V<br>$I_{OL}$ = 20.0 mA | — | — | 1.5 | V | |
| | | | $V_{CC}$ = 4.0 V to 5.5 V<br>$I_{OL}$ = 10.0 mA | — | — | 1.0 | | |
| | | | $V_{CC}$ = 4.0 V to 5.5 V<br>$I_{OL}$ = 1.6 mA | — | — | 0.4 | | |
| | | | $I_{OL}$ = 0.4 mA | — | — | 0.4 | | |

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Notes |
|---|---|---|---|---|---|---|---|---|
| Input/ output leakage current | $\lvert I_{IL} \rvert$ | OSC1, $\overline{RES}$, ,$\overline{NMI}$, $\overline{WKP0}$ to $\overline{WKP5}$, $\overline{IRQ0}$, $\overline{IRQ3}$, $\overline{ADTRG}$, TRGV, TMRIV, TMCIV, FTCI, FTIOA to FTIOD, RXD, SCK3 | $V_{IN}$ = 0.5 V to ($V_{CC}$ – 0.5 V) | — | — | 1.0 | µA | |
| | | P12 to P10, P17 to P14, P22 to P20, P57 to P50, P76 to P74, P84 to P80 | $V_{IN}$ = 0.5 V to ($V_{CC}$ – 0.5 V) | — | — | 1.0 | µA | |
| | | PB3 to PB0 | $V_{IN}$ = 0.5 V to ($AV_{CC}$ – 0.5 V) | — | — | 1.0 | µA | |
| Pull-up MOS current | $-I_p$ | P12 to P10, P17 to P14, P55 to P50 | $V_{CC}$ = 5.0 V, $V_{IN}$ = 0.0 V | 50.0 | — | 300.0 | µA | |
| | | | $V_{CC}$ = 3.0 V, $V_{IN}$ = 0.0 V | — | 60.0 | — | | Reference value |
| Input capacitance | $C_{in}$ | All input pins except power supply pins | f = 1 MHz, $V_{IN}$ = 0.0 V, $T_a$ = 25°C | — | — | 15.0 | pF | |
| Active mode current consumption | $I_{OPE1}$ | $V_{CC}$ | Active mode 1 $V_{CC}$ = 5.0 V, $f_{OSC}$ = 16 MHz | — | 15.0 | 22.5 | mA | * |
| | | | Active mode 1 $V_{CC}$ = 3.0 V, $f_{OSC}$ = 10 MHz | — | 8.0 | — | | * Reference value |
| | $I_{OPE2}$ | $V_{CC}$ | Active mode 2 $V_{CC}$ = 5.0 V, $f_{OSC}$ = 16 MHz | — | 1.8 | 2.7 | mA | * |
| | | | Active mode 2 $V_{CC}$ = 3.0 V, $f_{OSC}$ = 10 MHz | — | 1.2 | — | | * Reference value |

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Notes |
|---|---|---|---|---|---|---|---|---|
| Sleep mode current consumption | $I_{SLEEP1}$ | $V_{CC}$ | Sleep mode 1 $V_{CC}$ = 5.0 V, $f_{OSC}$ = 16 MHz | — | 11.5 | 17.0 | mA | * |
| | | | Sleep mode 1 $V_{CC}$ = 3.0 V, $f_{OSC}$ = 10 MHz | — | 6.5 | — | | * Reference value |
| | $I_{SLEEP2}$ | $V_{CC}$ | Sleep mode 2 $V_{CC}$ = 5.0 V, $f_{OSC}$ = 16 MHz | — | 1.7 | 2.5 | mA | * |
| | | | Sleep mode 2 $V_{CC}$ = 3.0 V, $f_{OSC}$ = 10 MHz | — | 1.1 | — | | * Reference value |
| Standby mode current consumption | $I_{STBY}$ | $V_{CC}$ | 32-kHz crystal resonator not used | — | — | 5.0 | µA | * |
| RAM data retaining voltage | $V_{RAM}$ | $V_{CC}$ | | 2.0 | — | — | V | |

Note: * Pin states during current consumption measurement are given below (excluding current in the pull-up MOS transistors and output buffers).

| Mode | $\overline{RES}$ Pin | Internal State | Other Pins | Oscillator Pins |
|---|---|---|---|---|
| Active mode 1 | $V_{CC}$ | Operates | $V_{CC}$ | Main clock: ceramic or crystal resonator |
| Active mode 2 | | Operates ($\phi$OSC/64) | | |
| Sleep mode 1 | $V_{CC}$ | Only timers operate | $V_{CC}$ | |
| Sleep mode 2 | | Only timers operate ($\phi$OSC/64) | | |
| Standby mode | $V_{CC}$ | CPU and timers both stop | $V_{CC}$ | Main clock: ceramic or crystal resonator |

## Table 17.2 DC Characteristics (2)

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise indicated.

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Allowable output low current (per pin) | $I_{OL}$ | Output pins except port 8 | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 2.0 | mA |
| | | Port 8 | | — | — | 20.0 | mA |
| | | Port 8 | | — | — | 10.0 | mA |
| | | Output pins except port 8 | | — | — | 0.5 | mA |
| Allowable output low current (total) | $\sum I_{OL}$ | Output pins except port 8 | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 40.0 | mA |
| | | Port 8 | | — | — | 80.0 | mA |
| | | Output pins except port 8 | | — | — | 20.0 | mA |
| | | Port 8 | | — | — | 40.0 | mA |
| Allowable output high current (per pin) | $\lvert -I_{OH} \rvert$ | All output pins | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 2.0 | mA |
| | | | | — | — | 0.2 | mA |
| Allowable output high current (total) | $\lvert -\sum -I_{OH} \rvert$ | All output pins | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 30.0 | mA |
| | | | | — | — | 8.0 | mA |

### 17.2.3 AC Characteristics

**Table 17.3 AC Characteristics**

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise specified.

| Item | Symbol | Applicable Pins | Test Condition | Values | | | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| | | | | Min | Typ | Max | | |
| System clock oscillation frequency | $f_{OSC}$ | OSC1, OSC2 | $V_{CC}$ = 4.0 V to 5.5 V | 2.0 | — | 16.0 | MHz | *[1] |
| | | | | 2.0 | — | 10.0 | MHz | |
| System clock (ø) cycle time | $t_{cyc}$ | | | 1 | — | 64 | $t_{OSC}$ | *[2] |
| | | | | — | — | 12.8 | µs | |
| Instruction cycle time | | | | 2 | — | — | $t_{cyc}$ | |
| Oscillation stabilization time (crystal resonator) | $t_{rc}$ | OSC1, OSC2 | | — | — | 10.0 | ms | |
| Oscillation stabilization time (ceramic resonator) | $t_{rc}$ | OSC1, OSC2 | | — | — | 5.0 | ms | |
| External clock high width | $t_{CPH}$ | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | 25.0 | — | — | ns | Figure 17.1 |
| | | | | 40.0 | — | — | ns | |
| External clock low width | $t_{CPL}$ | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | 25.0 | — | — | ns | |
| | | | | 40.0 | — | — | ns | |
| External clock rise time | $t_{CPr}$ | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 10.0 | ns | |
| | | | | — | — | 15.0 | ns | |
| External clock fall time | $t_{CPf}$ | OSC1 | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 10.0 | ns | |
| | | | | — | — | 15.0 | ns | |

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| $\overline{\text{RES}}$ pin low width | $t_{REL}$ | $\overline{\text{RES}}$ | At power-on and in modes other than those below | $t_{rc}$ | — | — | ms | Figure 17.2 |
| | | | In active mode and sleep mode operation | 10 | — | — | $t_{cyc}$ | |
| Input pin high width | $t_{IH}$ | $\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, $\overline{\text{IRQ3}}$, $\overline{\text{WKP0}}$ to $\overline{\text{WKP5}}$, TMCIV, TMRIV, TRGV, $\overline{\text{ADTRG}}$, FTCI, FTIOA to FTIOD | | 2 | — | — | $t_{cyc}$ | Figure 17.3 |
| Input pin low width | $t_{IL}$ | $\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, $\overline{\text{IRQ3}}$, $\overline{\text{WKP0}}$ to $\overline{\text{WKP5}}$, TMCIV, TMRIV, TRGV, $\overline{\text{ADTRG}}$, FTCI, FTIOA to FTIOD | | 2 | — | — | $t_{cyc}$ | |

Notes: 1. When an external clock is input, the minimum system clock oscillator frequency is 1.0 MHz.

2. Determined by MA2 to MA0 in system control register 2 (SYSCR2).

## Table 17.4 Serial Interface (SCI3) Timing

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise specified.

| Item | | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|---|
| Input clock cycle | Asynchronous | $t_{Scyc}$ | SCK3 | | 4 | — | — | $t_{cyc}$ | Figure 17.4 |
| | Clocked synchronous | | | | 6 | — | — | $t_{cyc}$ | |
| Input clock pulse width | | $t_{SCKW}$ | SCK3 | | 0.4 | — | 0.6 | $t_{Scyc}$ | |
| Transmit data delay time (clocked synchronous) | | $t_{TXD}$ | TXD | $V_{CC}$ = 4.0 V to 5.5 V | — | — | 1 | $t_{cyc}$ | Figure 17.5 |
| | | | | | — | — | 1 | $t_{cyc}$ | |
| Receive data setup time (clocked synchronous) | | $t_{RXS}$ | RXD | $V_{CC}$ = 4.0 V to 5.5 V | 62.5 | — | — | ns | |
| | | | | | 100.0 | — | — | ns | |
| Receive data hold time (clocked synchronous) | | $t_{RXH}$ | RXD | $V_{CC}$ = 4.0 V to 5.5 V | 62.5 | — | — | ns | |
| | | | | | 100.0 | — | — | ns | |

### 17.2.4 A/D Converter Characteristics

**Table 17.5 A/D Converter Characteristics**

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise specified.

| Item | Symbol | Applicable Pins | Test Condition | Values | | | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| | | | | Min | Typ | Max | | |
| Analog power supply voltage | $AV_{CC}$ | $AV_{CC}$ | | 3.3 | $V_{CC}$ | 5.5 | V | *[1] |
| Analog input voltage | $AV_{IN}$ | AN3 to AN0 | | $V_{SS}$ – 0.3 | — | $AV_{CC}$ + 0.3 | V | |
| Analog power supply current | $AI_{OPE}$ | $AV_{CC}$ | $AV_{CC}$ = 5.0 V<br>$f_{OSC}$ = 16 MHz | — | — | 2.0 | mA | |
| | $AI_{STOP1}$ | $AV_{CC}$ | | — | 50 | — | µA | *[2] Reference value |
| | $AI_{STOP2}$ | $AV_{CC}$ | | — | — | 5.0 | µA | *[3] |
| Analog input capacitance | $C_{AIN}$ | AN3 to AN0 | | — | — | 30.0 | pF | |
| Allowable signal source impedance | $R_{AIN}$ | AN3 to AN0 | | — | — | 5.0 | kΩ | |
| Resolution (data length) | | | | 10 | 10 | 10 | bit | |
| Conversion time (single mode) | | | $AV_{CC}$ = 3.3 V to 5.5 V | 134 | — | — | $t_{cyc}$ | |
| Nonlinearity error | | | | — | — | ±7.5 | LSB | |
| Offset error | | | | — | — | ±7.5 | LSB | |
| Full-scale error | | | | — | — | ±7.5 | LSB | |
| Quantization error | | | | — | — | ±0.5 | LSB | |
| Absolute accuracy | | | | — | — | ±8.0 | LSB | |
| Conversion time (single mode) | | | $AV_{CC}$ = 4.0 V to 5.5 V | 70 | — | — | $t_{cyc}$ | |
| Nonlinearity error | | | | — | — | ±7.5 | LSB | |
| Offset error | | | | — | — | ±7.5 | LSB | |
| Full-scale error | | | | — | — | ±7.5 | LSB | |
| Quantization error | | | | — | — | ±0.5 | LSB | |
| Absolute accuracy | | | | — | — | ±8.0 | LSB | |

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| Conversion time (single mode) | | | $AV_{CC}$ = 4.0 V to 5.5 V | 134 | — | — | $t_{cyc}$ | |
| Nonlinearity error | | | | — | — | ±3.5 | LSB | |
| Offset error | | | | — | — | ±3.5 | LSB | |
| Full-scale error | | | | — | — | ±3.5 | LSB | |
| Quantization error | | | | — | — | ±0.5 | LSB | |
| Absolute accuracy | | | | — | — | ±4.0 | LSB | |

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 and subsleep modes while the A/D converter is idle.

### 17.2.5 Watchdog Timer

**Table 17.6 Watchdog Timer Characteristics**

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise specified.

| Item | Symbol | Applicable Pins | Test Condition | Values Min | Typ | Max | Unit | Reference Figure |
|---|---|---|---|---|---|---|---|---|
| On-chip oscillator overflow time | $t_{OVF}$ | | | 0.2 | 0.4 | — | s | * |

Note: * Shows the time to count from 0 to 255, at which point an internal reset is generated, when the internal oscillator is selected.

### 17.2.6 Flash Memory Characteristics

**Table 17.7 Flash Memory Characteristics**

$V_{CC}$ = 3.0 V to 5.5 V, $V_{SS}$ = 0.0 V, $T_a$ = –20°C to +75°C, unless otherwise specified.

| Item | | Symbol | Test Condition | Values Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Programming time (per 128 bytes)*1*2*4 | | $t_P$ | | — | 7 | 200 | ms |
| Erase time (per block) *1*3*6 | | $t_E$ | | — | 100 | 1200 | ms |
| Reprogramming count | | $N_{WEC}$ | | 1000 | 10000 | — | Times |
| 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 | 1 ≤ n ≤ 6 | 28 | 30 | 32 | µs |
| | | z2 | 7 ≤ n ≤ 1000 | 198 | 200 | 202 | µs |
| | | z3 | Additional-programming | 8 | 10 | 12 | µs |
| | 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 | Test Condition | Values Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 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. Make the time settings in accordance with the program/erase algorithms.
2. The programming time for 128 bytes. (Indicates the total time for which the P bit in flash memory control register 1 (FLMCR1) is set. The program-verify time is not included.)
3. The time required to erase one block. (Indicates the time for which the E bit in flash memory control register 1 (FLMCR1) is set. The erase-verify time is not included.)
4. Programming time maximum value ($t_P$(MAX)) = wait time after P bit setting (z) × maximum programming count (N)
5. Set the maximum programming count (N) according to the actual set values of z1, z2, and z3, so that it does not exceed the programming time maximum value ($t_P$(MAX)). The wait time after P bit setting (z1, z2) should be changed as follows according to the value of the programming count (n).

   Programming count (n)

   $1 \leq n \leq 6$ z1 = 30 µs

   $7 \leq n \leq 1000$ z2 = 200 µs
6. Erase time maximum value ($t_E$(max)) = wait time after E bit setting (z) × maximum erase count (N)
7. Set the maximum maximum erase count (N) according to the actual set value of (z), so that it does not exceed the erase time maximum value ($t_E$(max)).

## 17.3 Operation Timing

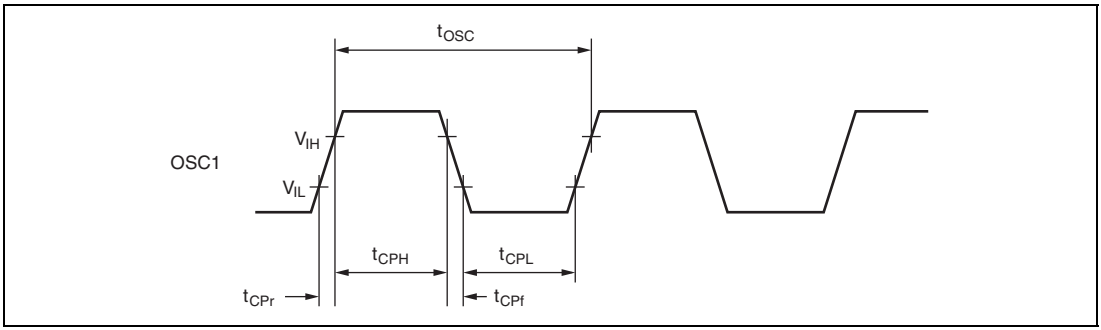



**Figure 17.1 System Clock Input Timing**

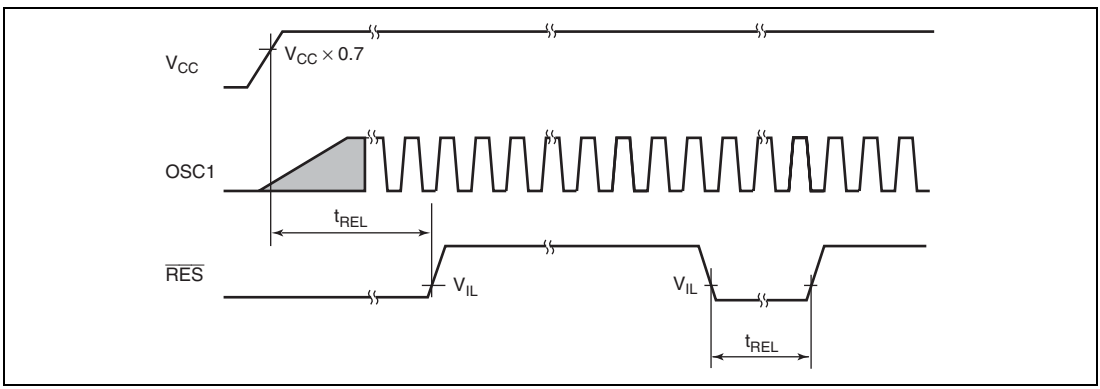



**Figure 17.2 $\overline{RES}$ Low Width Timing**

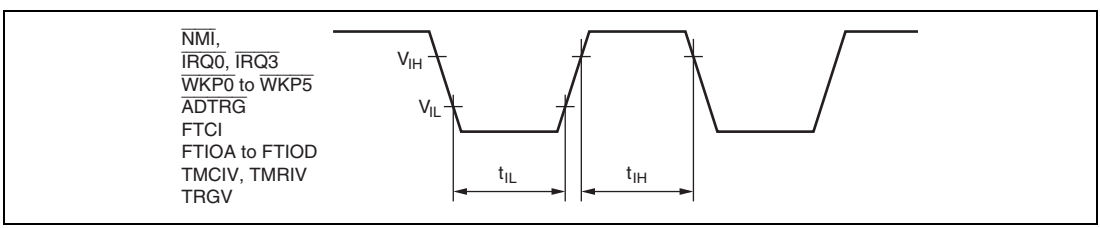



**Figure 17.3 Input Timing**

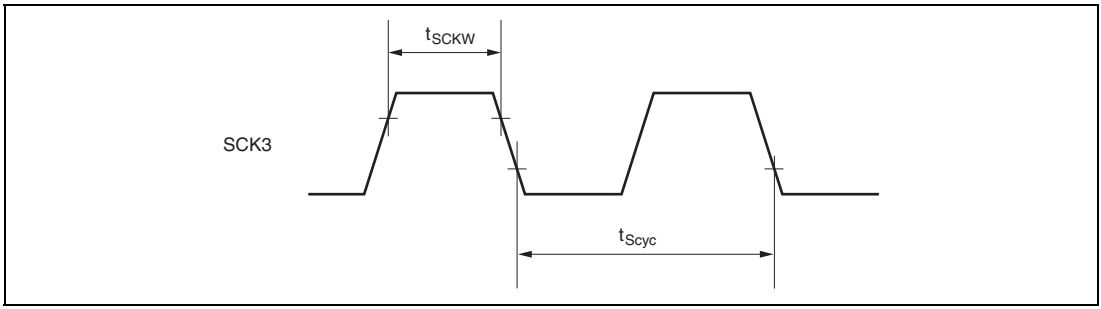
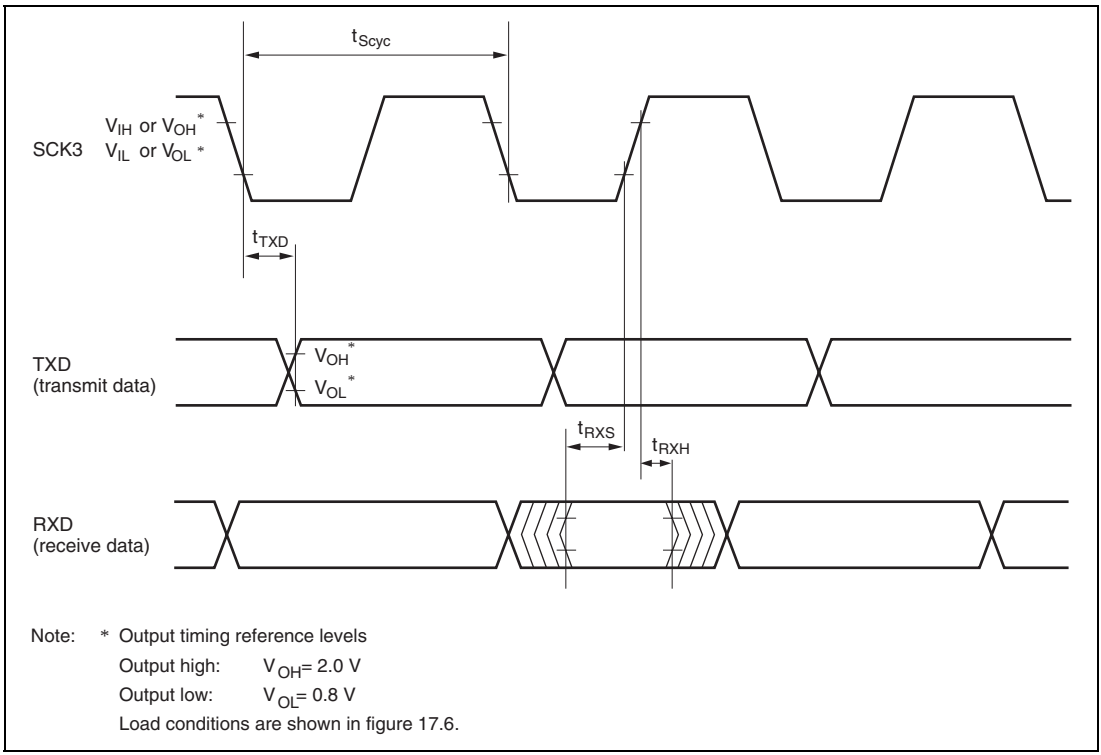



**Figure 17.4 SCK3 Input Clock Timing**




Note: * Output timing reference levels
Output high: $V_{OH}$= 2.0 V
Output low: $V_{OL}$= 0.8 V
Load conditions are shown in figure 17.6.

**Figure 17.5 SCI3 Input/Output Timing in Clocked Synchronous Mode**

## 17.4 Output Load Condition

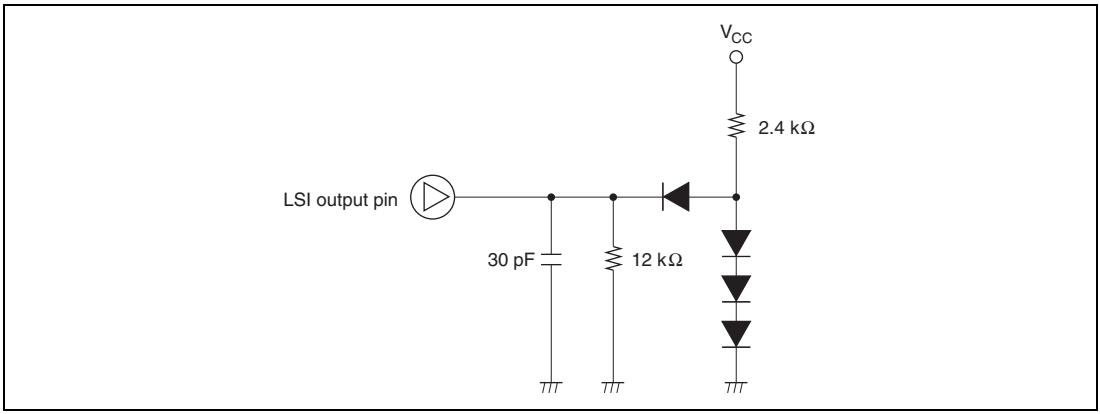



**Figure 17.6 Output Load Circuit**

# Appendix A Instruction Set

## A.1 Instruction List

### Operand Notation

| 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 |
| ¬ | 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

| 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, Rd | L | 6 | | | | | | | | | #xx:32 → Rd32 | — | — | ↕ | ↕ | 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 | |

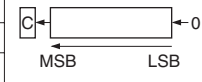
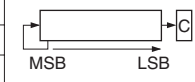
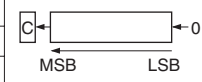
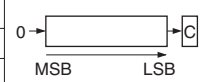
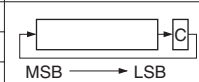
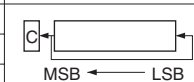
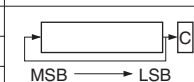
| Mnemonic | | Operand Size | Addressing Mode and Instruction Length (bytes) | | | | | | | | | Operation | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, 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 | Operation | Addressing Mode and Instruction Length (bytes) | | | | | | | | | 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 | 0–Rd8 → Rd8 | | 2 | | | | | | | | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | NEG.W Rd | W | 0–Rd16 → Rd16 | | 2 | | | | | | | | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| | NEG.L ERd | L | 0–ERd32 → ERd32 | | 2 | | | | | | | | — | ↕ | ↕ | ↕ | ↕ | ↕ | 2 | |
| EXTU | EXTU.W Rd | W | 0 → (<bits 15 to 8> of Rd16) | | 2 | | | | | | | | — | — | 0 | ↕ | 0 | — | 2 | |
| | EXTU.L ERd | L | 0 → (<bits 31 to 16> of ERd32) | | 2 | | | | | | | | — | — | 0 | ↕ | 0 | — | 2 | |
| EXTS | EXTS.W Rd | W | (<bit 7> of Rd16) → (<bits 15 to 8> of Rd16) | | 2 | | | | | | | | — | — | ↕ | ↕ | 0 | — | 2 | |
| | EXTS.L ERd | L | (<bit 15> of ERd32) → (<bits 31 to 16> of ERd32) | | 2 | | | | | | | | — | — | ↕ | ↕ | 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 | |
| BIST | BST #xx:3, @aa:8 | B | | | | | | 4 | | | | C → (#xx:3 of @aa:8) | — | — | — | — | — | — | 8 | |
| | 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 | |
| BIAND | BAND #xx:3, @aa:8 | B | | | | | | 4 | | | | C∧(#xx:3 of @aa:8) → C | — | — | — | — | — | ↕ | 6 | |
| | 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) | | | | | | | | | Operation | | Condition Code | | | | | | No. of States*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #xx | Rn | @ERn | @(d, ERn) | @–ERn/@ERn+ | @aa | @(d, PC) | @@aa | — | | Branch Condition | I | H | N | Z | V | C | 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 |
| TRAPA | TRAPA #x:2 | — | | | | | | | | | 2 | PC → @–SP<br>CCR → @–SP<br><vector> → PC | 1 | — | — | — | — | — | 14 | 16 |
| 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:

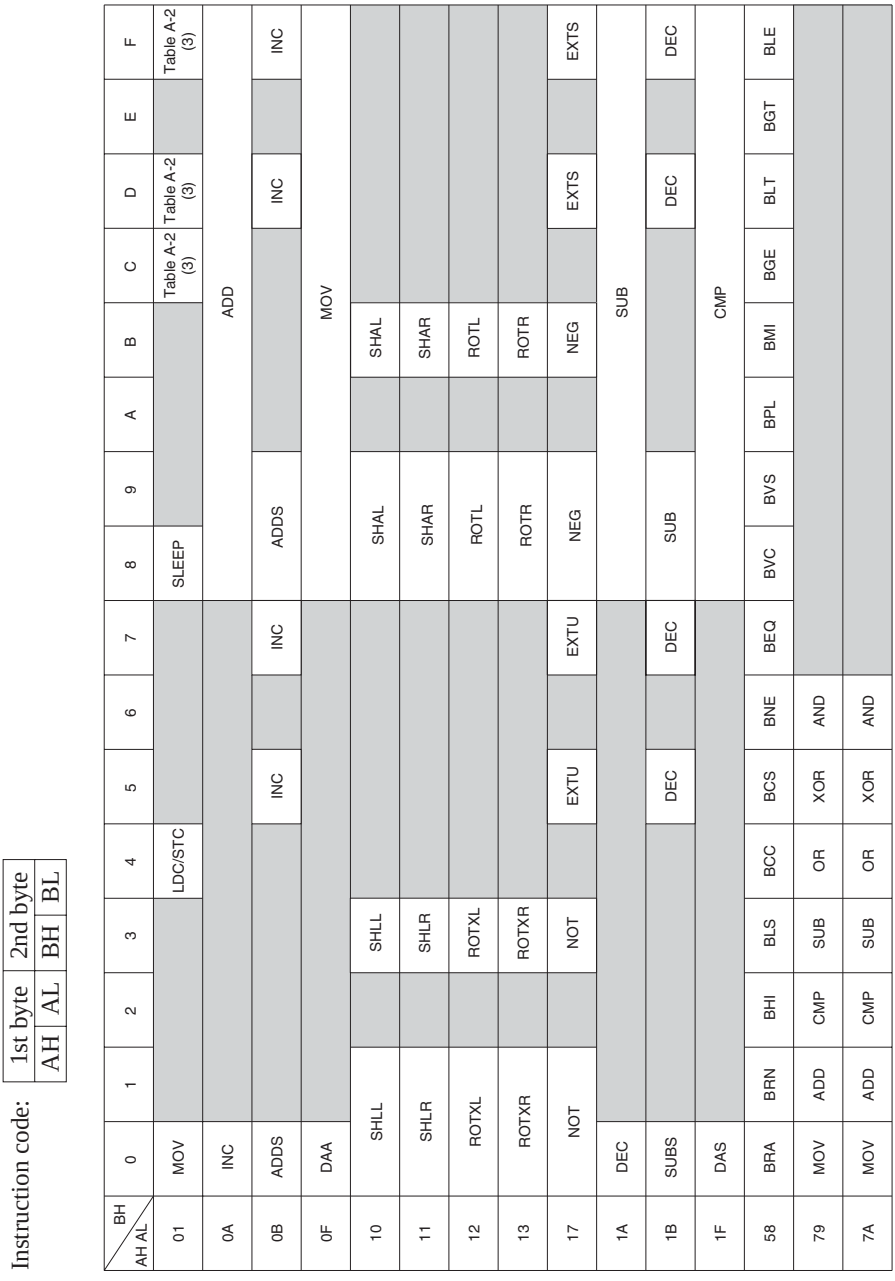
| 1st byte | | 2nd byte | |
|---|---|---|---|
| AH | AL | BH | BL |

Instruction when most significant bit of BH is 0.

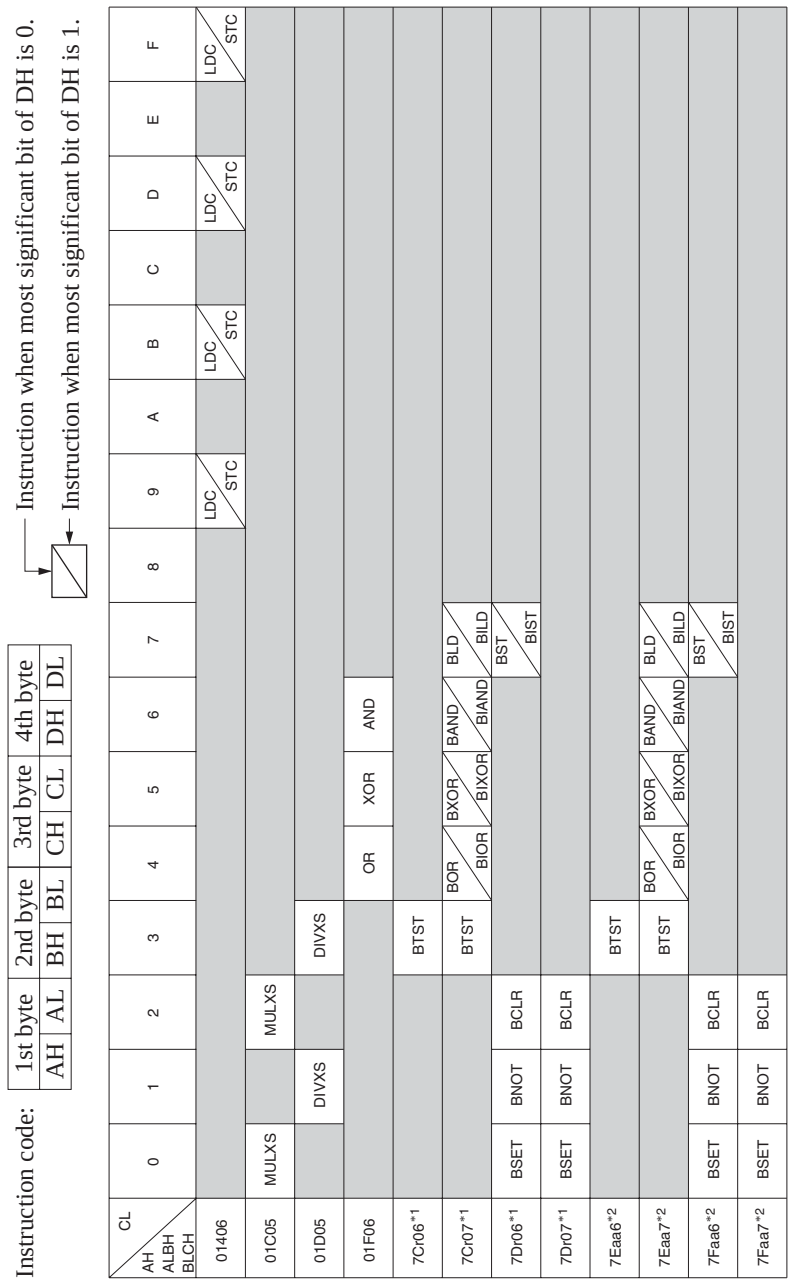
Instruction when most significant bit of BH is 1.

| 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 | TRAPA | Table A-2 (2) | JMP | | | BSR | JSR | | |
| 6 | BSET | BNOT | BCLR | BTST | OR / BOR / BIOR | XOR / BXOR / BIXOR | AND / BAND / BIAND | BST / BLD / BILD | MOV | Table A-2 (2) | Table A-2 (2) | EEPMOV | Table A-2 (3) | | | |
| 7 | | | | | | | | | | | | | | | | |
| 8 | ADD | | | | | | | | | | | | | | | |
| 9 | ADDX | | | | | | | | | | | | | | | |
| A | CMP | | | | | | | | | | | | | | | |
| B | SUBX | | | | | | | | | | | | | | | |
| C | OR | | | | | | | | | | | | | | | |
| D | XOR | | | | | | | | | | | | | | | |
| E | AND | | | | | | | | | | | | | | | |
| F | MOV | | | | | | | | | | | | | | | |

## Table A.2 Operation Code Map (2)



## Table A.2 Operation Code Map (3)




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, J = K = M = N= 0

From table A.3:
$S_I = 2$, $S_L = 2$

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, J = K = 1, 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 16.1, Register Addresses (Address Order).

## Table A.4 Number of Cycles in Each Instruction

| Instruction | Mnemonic | Instruction Fetchl | 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:8, Rd | 1 | | | | | |
| | BIOR #xx:8, @ERd | 2 | | | 1 | | |
| | BIOR #xx:8, @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 | |
| | | 5 | | | | 1 | |
| | STC CCR, @(d:24,ERd) | 2 | | | | 1 | 2 |
| | | 3 | | | | 1 | |
| | STC CCR,@-ERd | 4 | | | | 1 | |
| | STC CCR, @aa:16 | | | | | | |
| | STC CCR, @aa:24 | | | | | | |
| 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 | | | | | |
| TRAPA | TRAPA #xx:2 | 2 | 1 | 2 | | | 4 |
| 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 and R4. The source and destination operands are accessed n+1 times respectively.

2. 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 | TRAPA | — | — | — | — | — | — | — | — | — | — | — | — | ◯ |
| | 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 |

# Appendix B I/O Port Block Diagrams

## B.1 I/O Port Block

$\overline{\text{RES}}$ goes low in a reset, and $\overline{\text{SBY}}$ goes low in a reset and in standby mode.

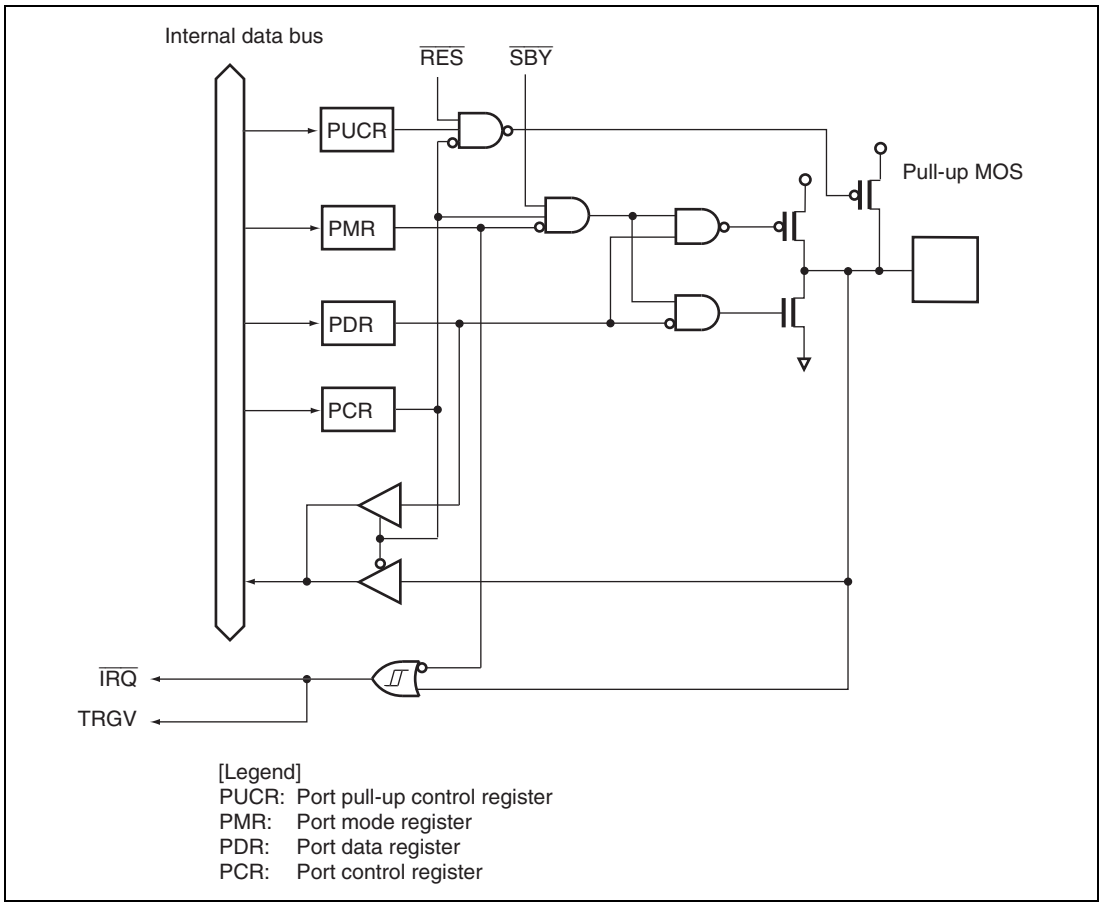



**Figure B.1 Port 1 Block Diagram (P17)**

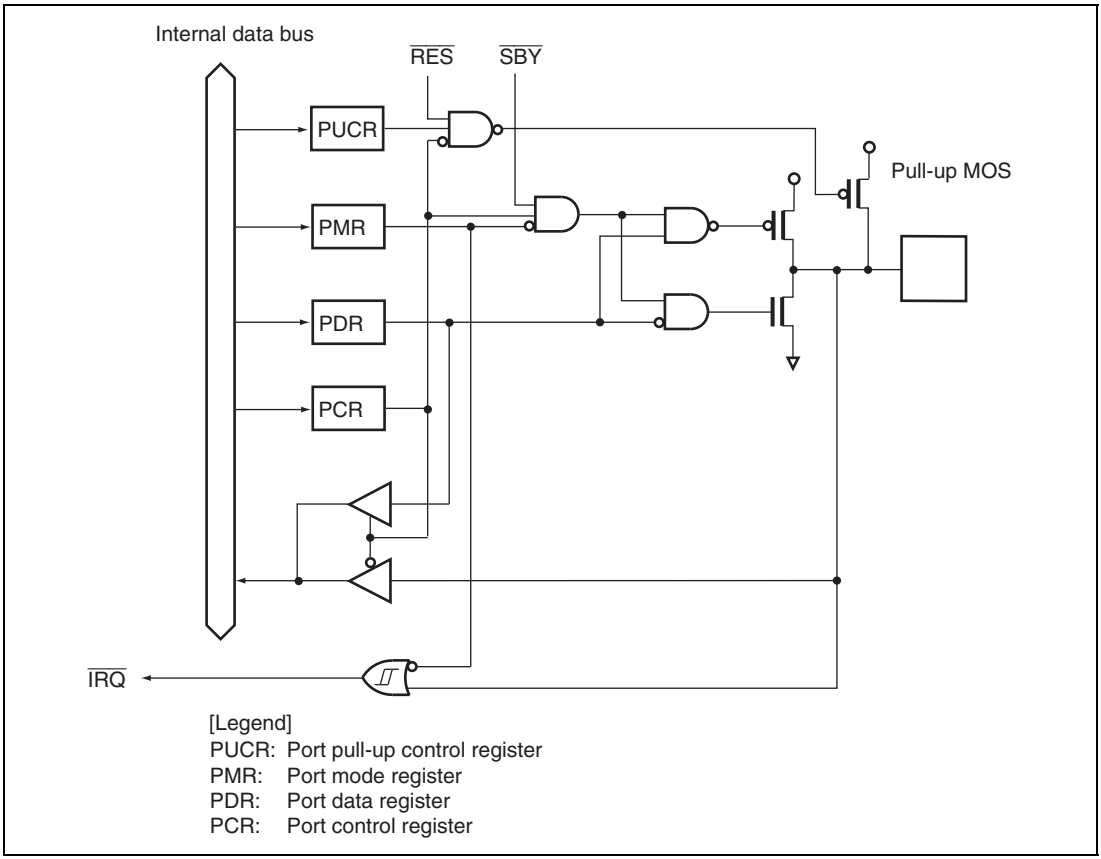



**Figure B.2 Port 1 Block Diagram (P14)**

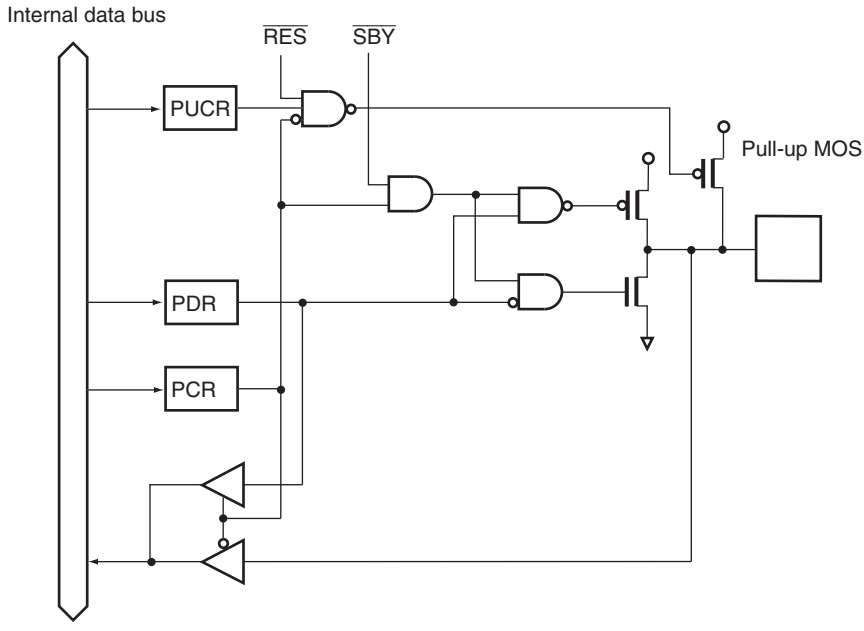



**Figure B.3 Port 1 Block Diagram (P16, P15, P12, P10)**

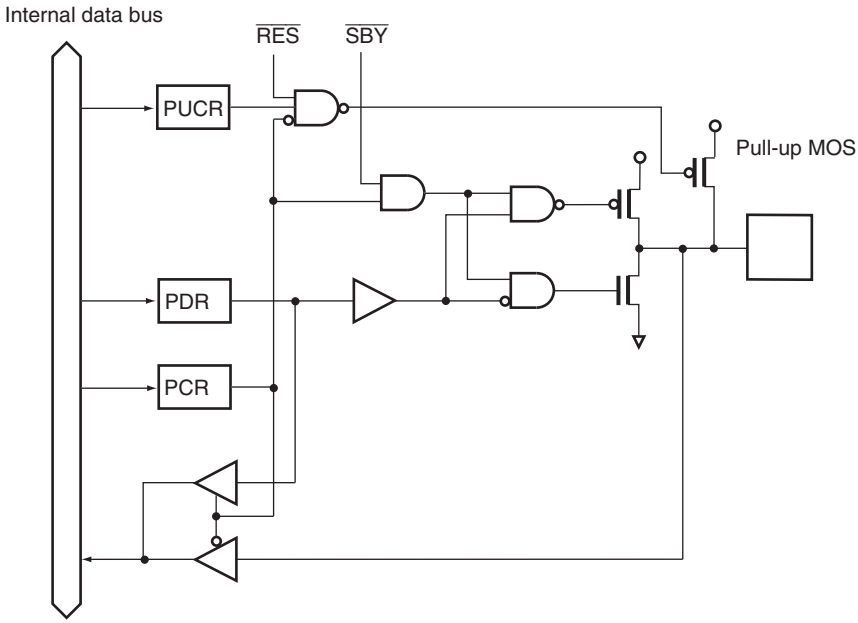



**Figure B.4 Port 1 Block Diagram (P11)**

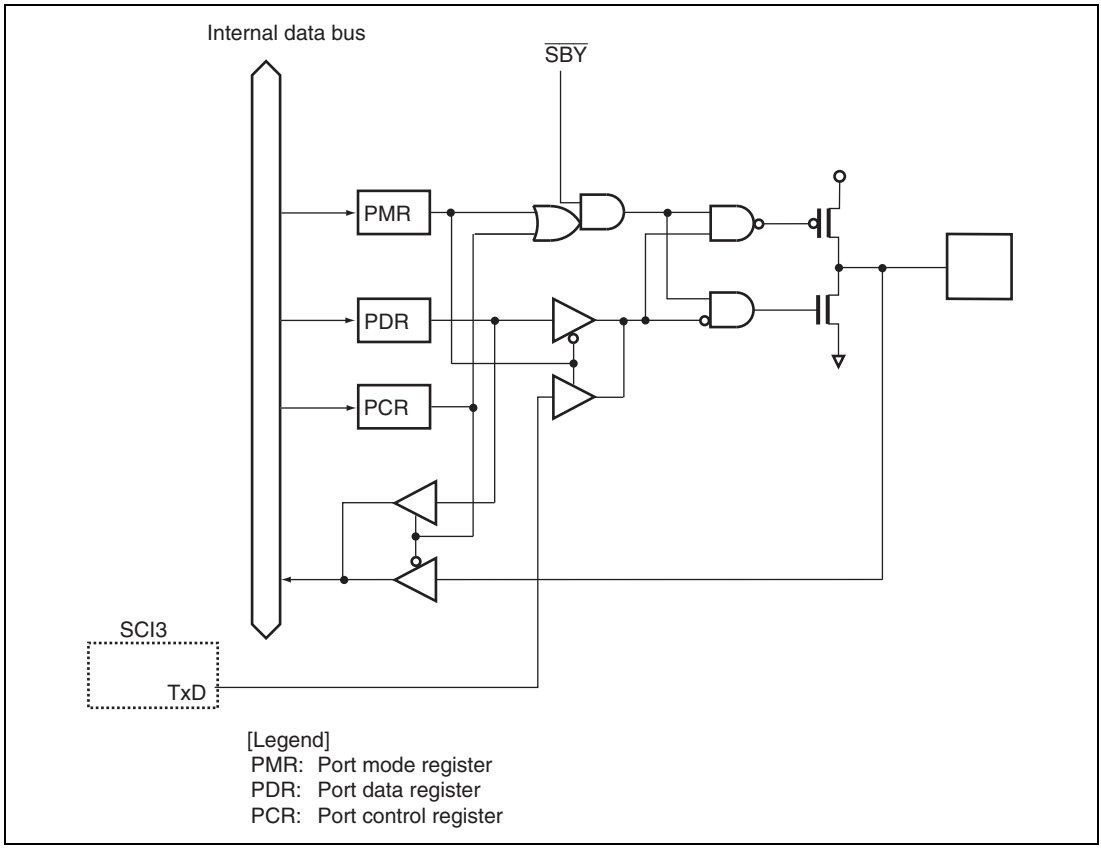



**Figure B.5 Port 2 Block Diagram (P22)**

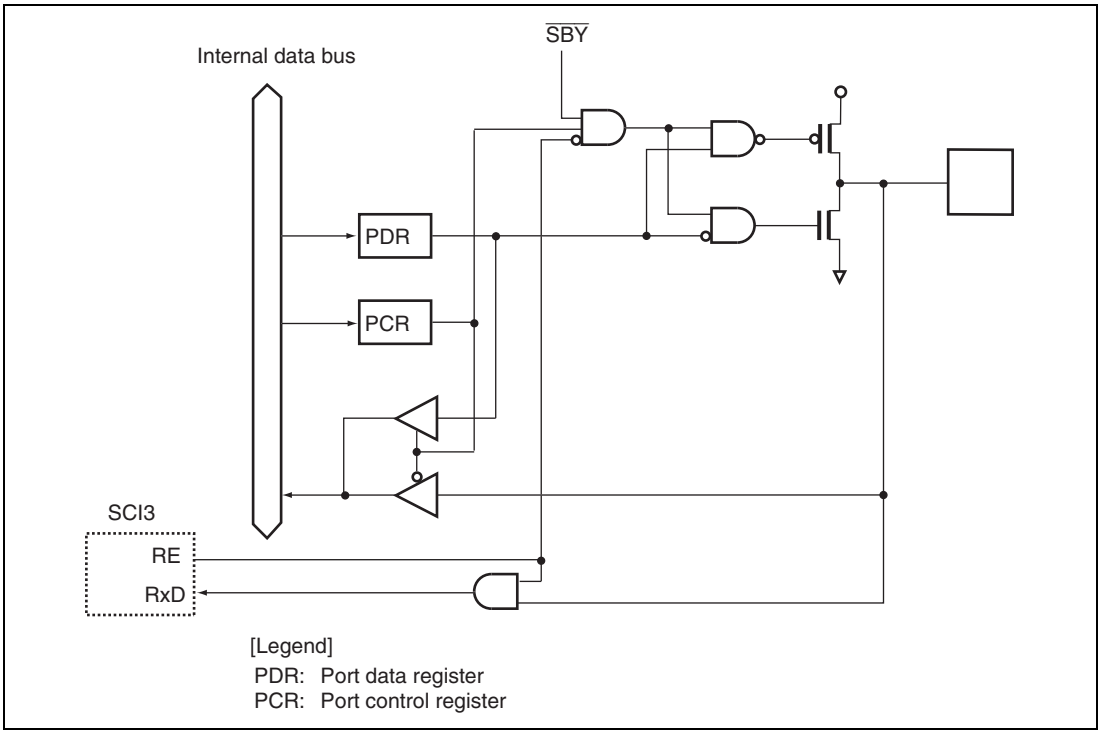



**Figure B.6 Port 2 Block Diagram (P21)**

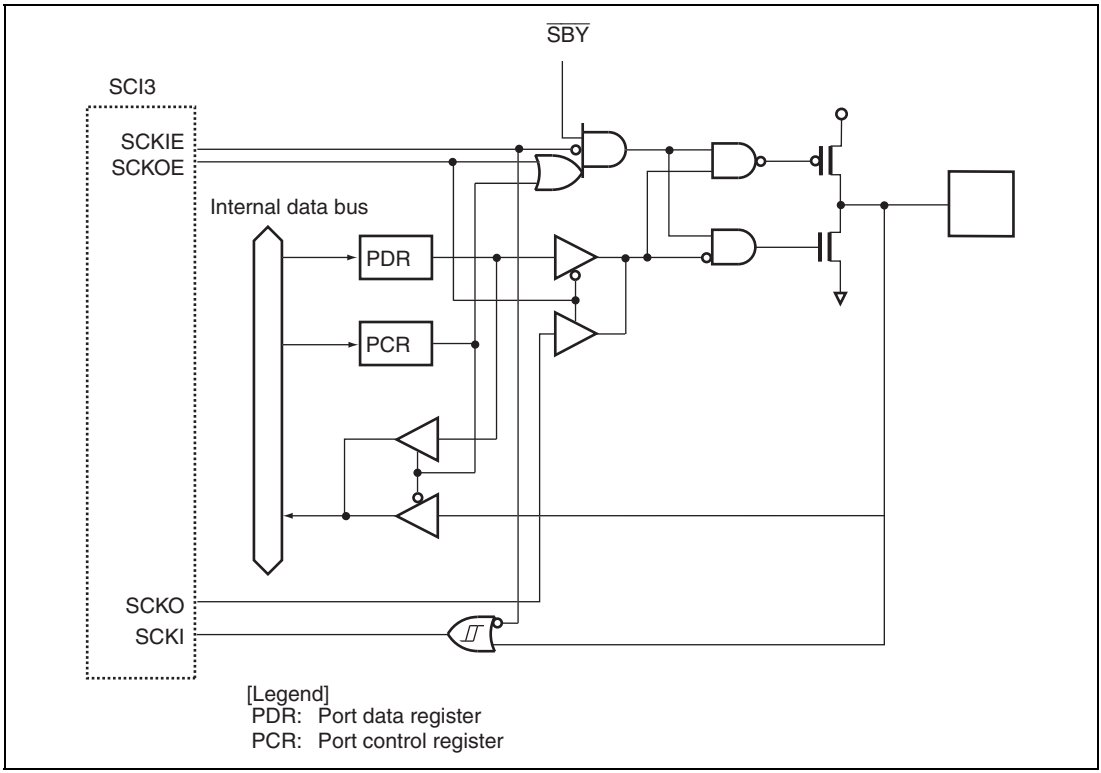



**Figure B.7 Port 2 Block Diagram (P20)**

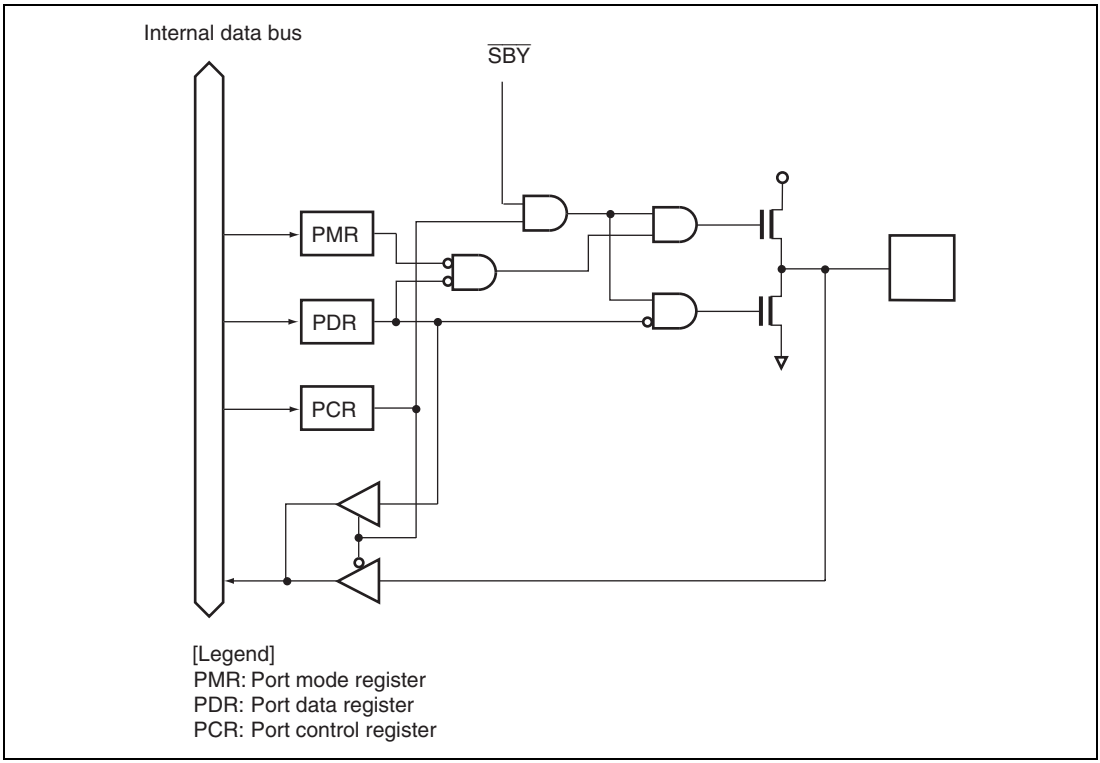



**Figure B.8 Port 5 Block Diagram (P57, P56)**

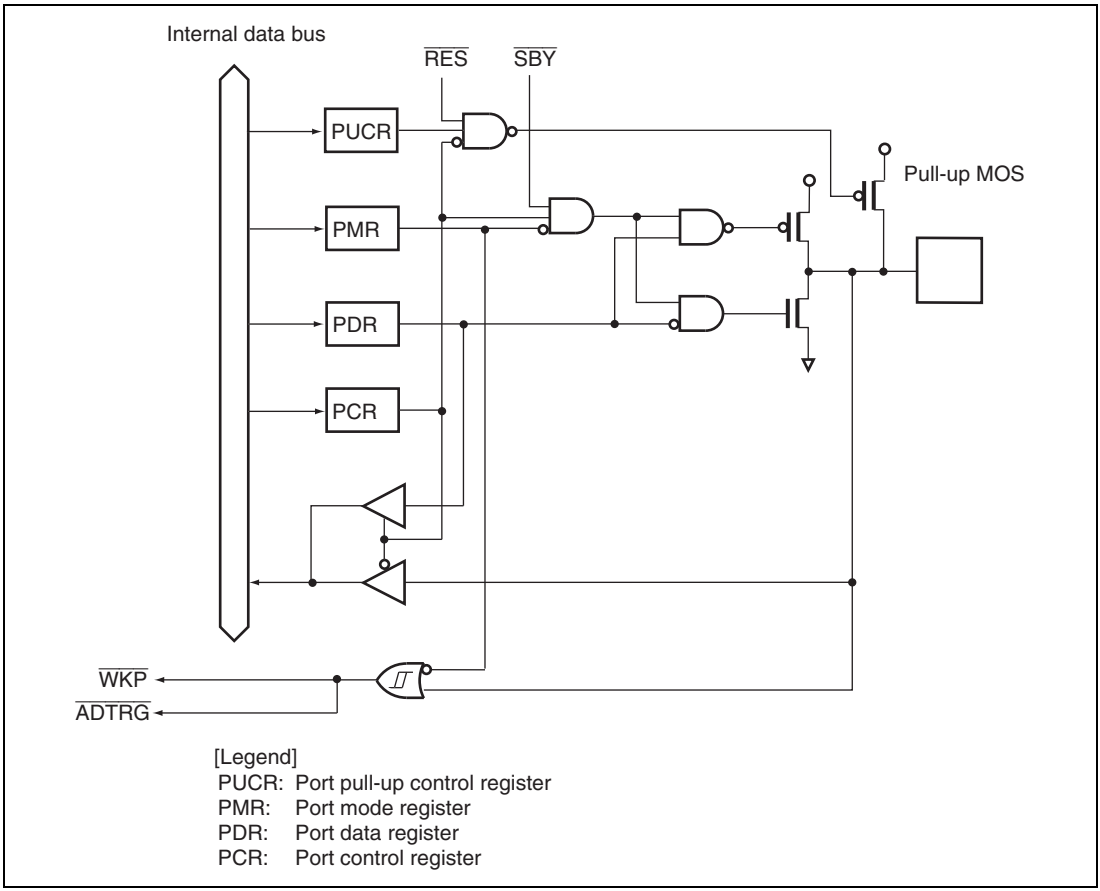



**Figure B.9 Port 5 Block Diagram (P55)**

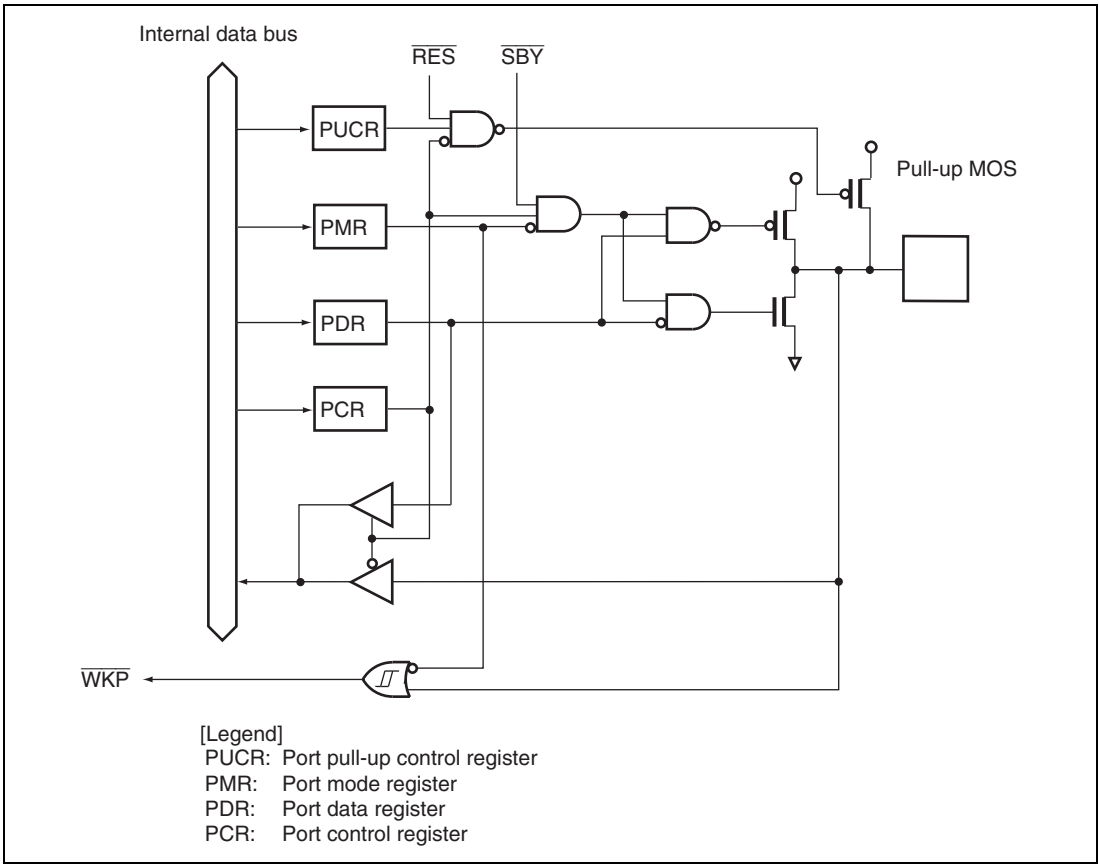



**Figure B.10 Port 5 Block Diagram (P54 to P50)**

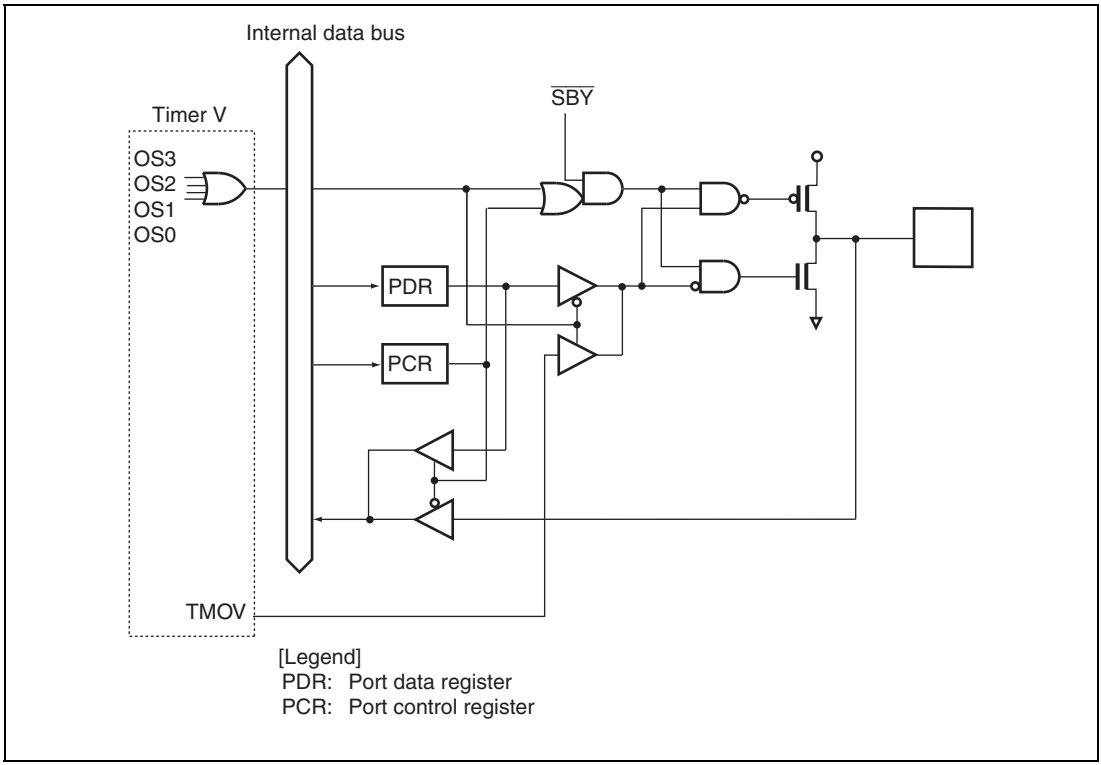



**Figure B.11 Port 7 Block Diagram (P76)**

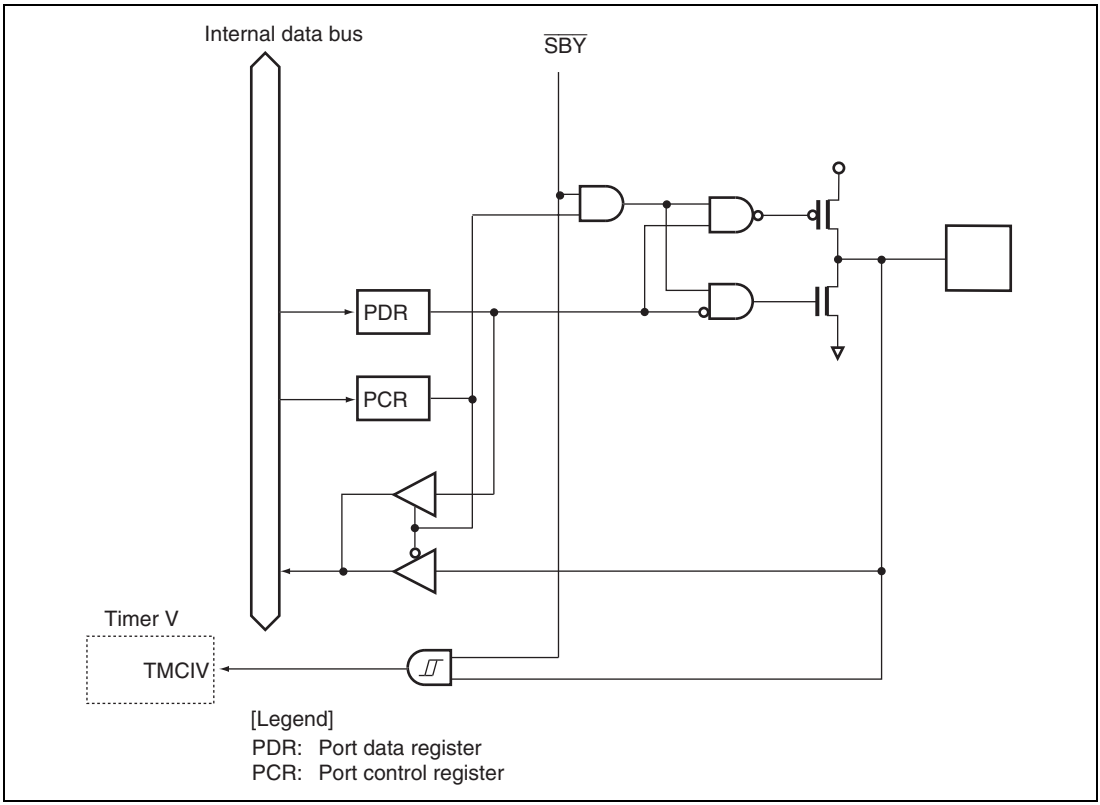



**Figure B.12 Port 7 Block Diagram (P75)**

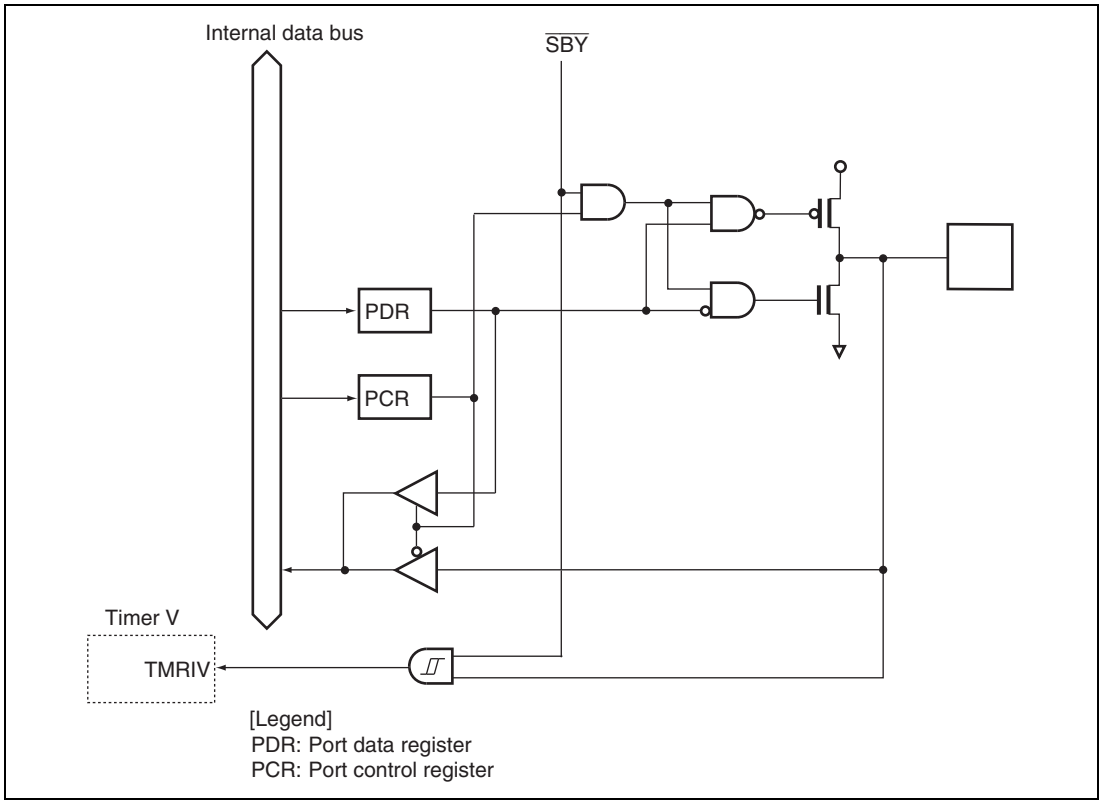



**Figure B.13 Port 7 Block Diagram (P74)**

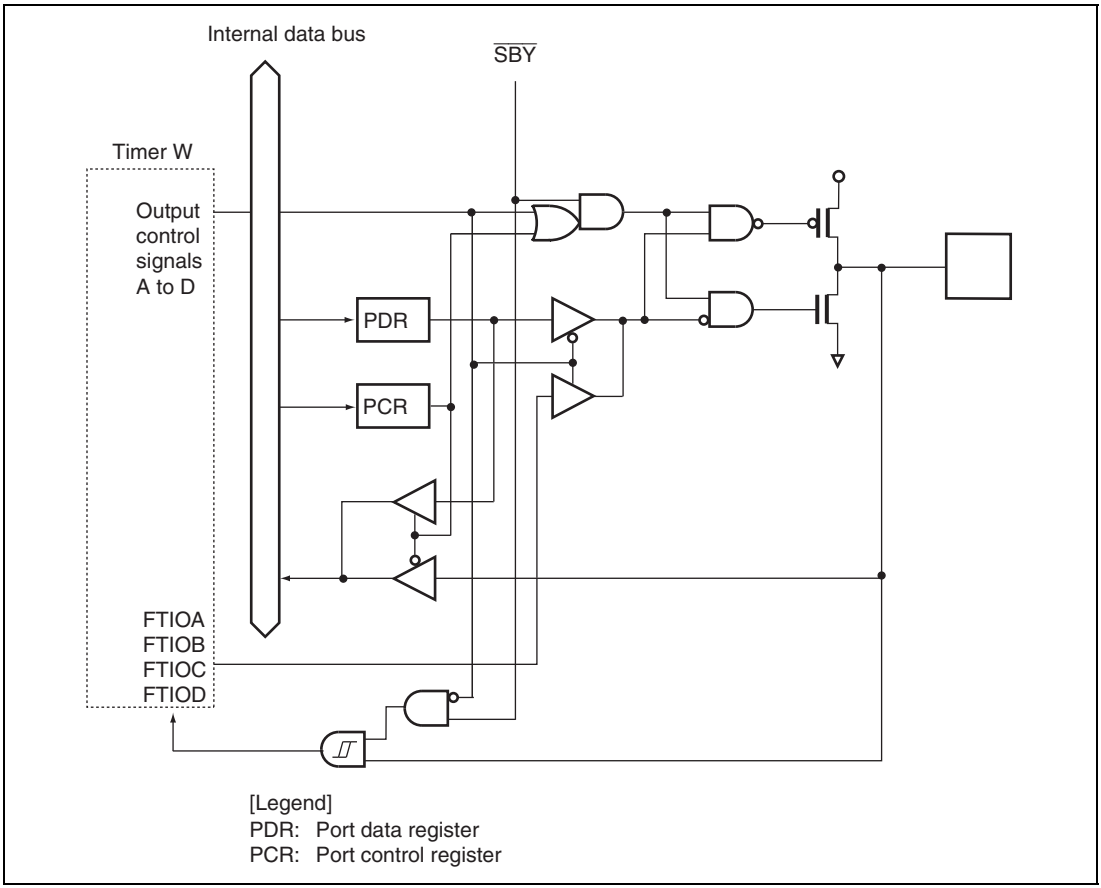



**Figure B.14 Port 8 Block Diagram (P84 to P81)**

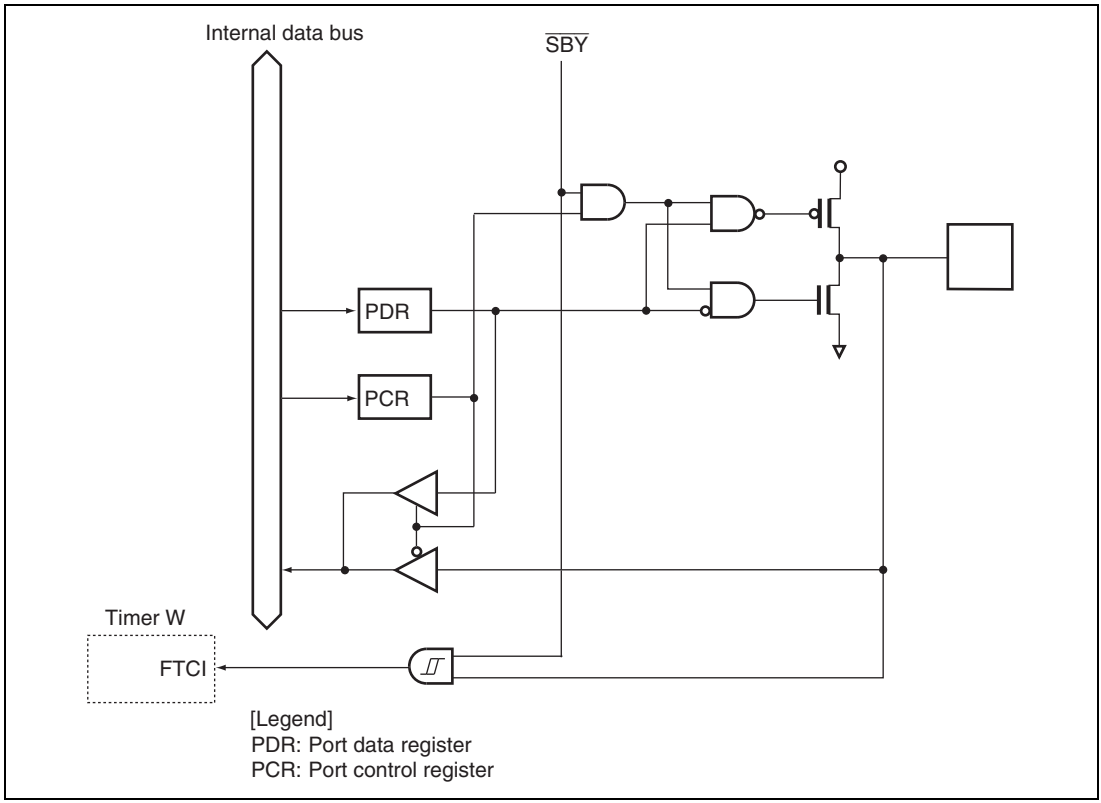



**Figure B.15 Port 8 Block Diagram (P80)**

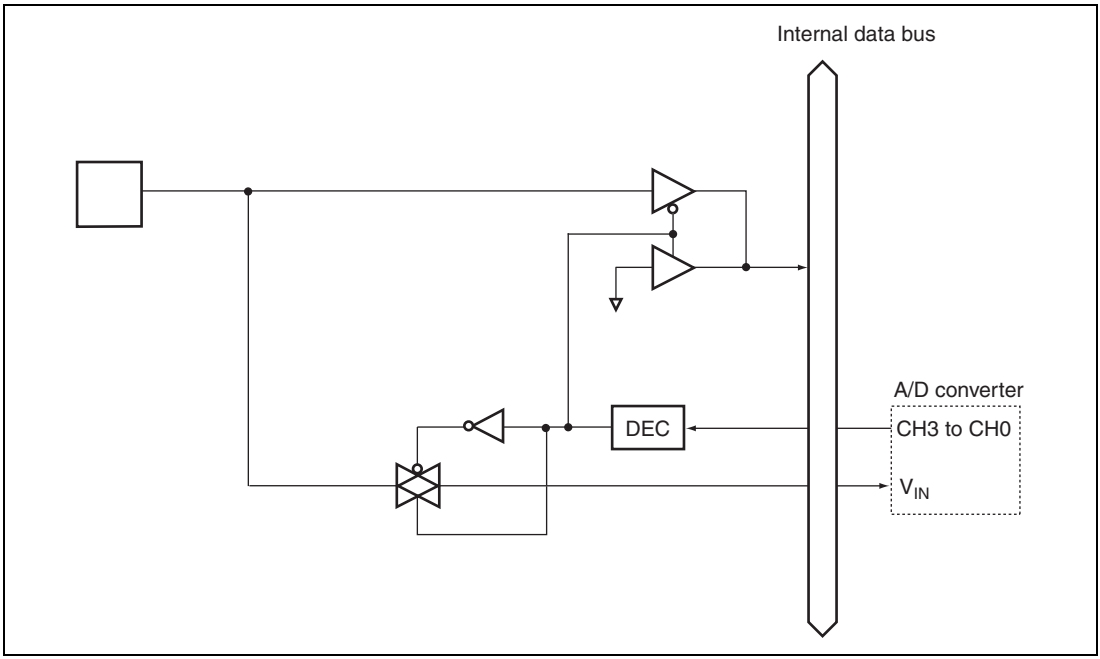



**Figure B.16 Port B Block Diagram (PB3 to PB0)**

## B.2 Port States in Each Operating State

| Port | Reset | Active | Sleep | Subsleep | Standby |
|---|---|---|---|---|---|
| P17 to P14, P12 to P10 | High impedance | Functioning | Retained | Retained | High impedance* |
| P22 to P20 | High impedance | Functioning | Retained | Retained | High impedance |
| P57 to P50 | High impedance | Functioning | Retained | Retained | High impedance* |
| P76 to P74 | High impedance | Functioning | Retained | Retained | High impedance |
| P84 to P80 | High impedance | Functioning | Retained | Retained | High impedance |
| PB3 to PB0 | High impedance | High impedance | High impedance | Retained | High impedance |

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

# Appendix C Product Code Lineup

| Product Type | | | Product Code | Model Marking | Package Code |
|---|---|---|---|---|---|
| H8/3672 | Flash memory version | Standard product | HD64F3672FP | HD64F3672FP | LQFP-64 (FP-64E) |
| | | | HD64F3672FX | HD64F3672FX | LQFP-48 (FP-48F) |
| | | | HD64F3672FY | HD64F3672FY | LQFP-48 (FP-48B) |
| H8/3670 | Flash memory version | Standard product | HD64F3672FP | HD64F3672FP | LQFP-64 (FP-64E) |
| | | | HD64F3670FX | HD64F3670FX | LQFP-48 (FP-48F) |
| | | | HD64F3670FY | HD64F3670FY | LQFP-48 (FP-48B) |

# Appendix D Package Dimensions

The package dimensions that are shows in the Renesas Semiconductor Packages Data Book have priority.

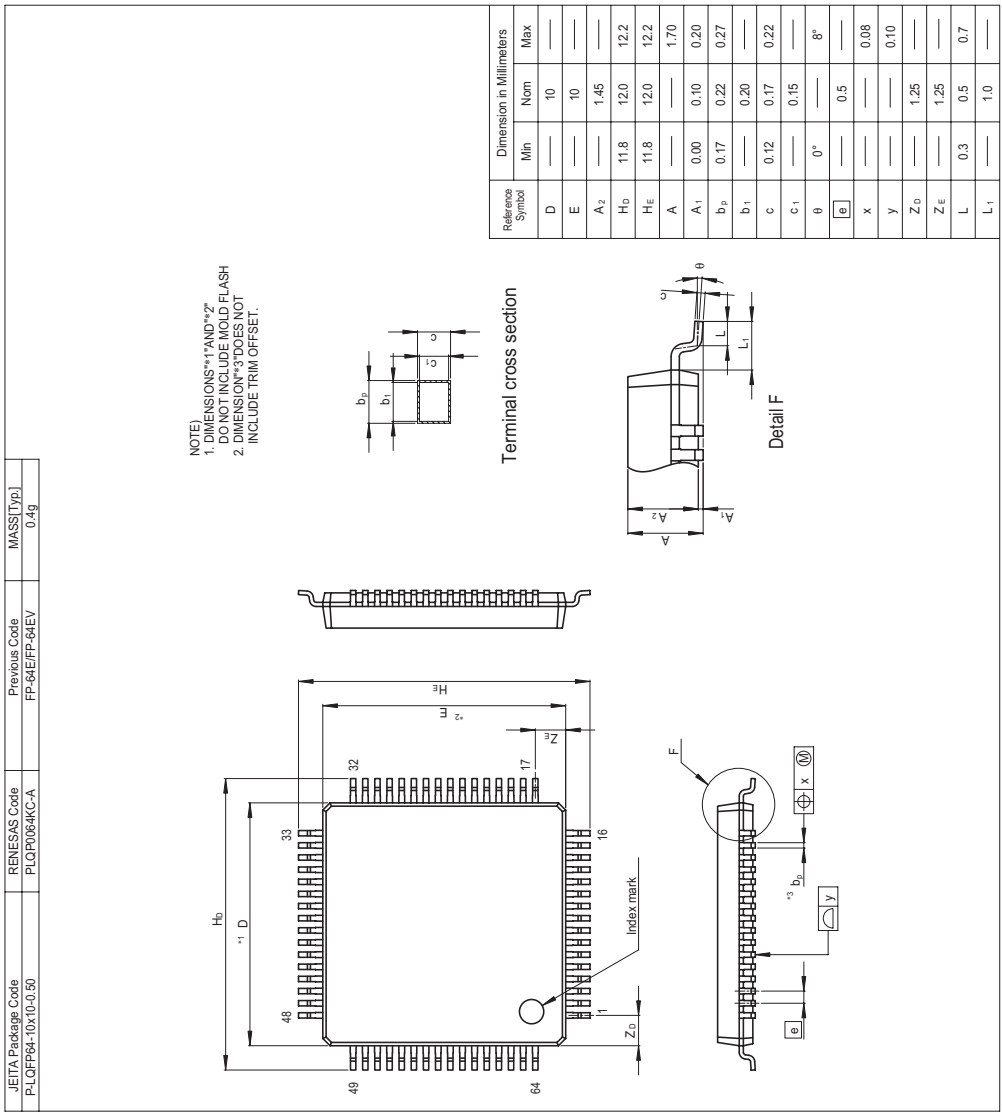



**Figure D.1 FP-64E Package Dimensions**

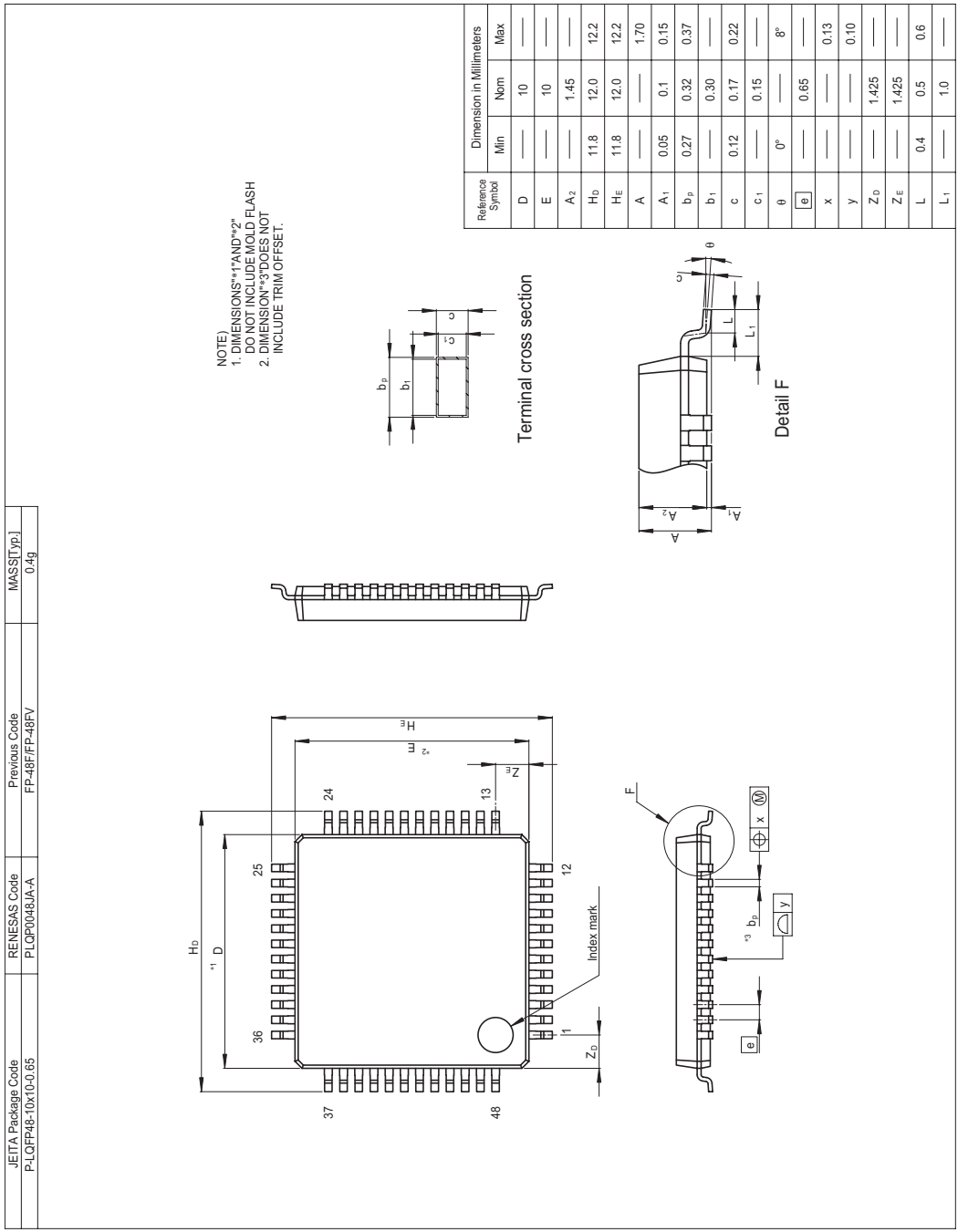



**Figure D.2 FP-48F Package Dimensions**

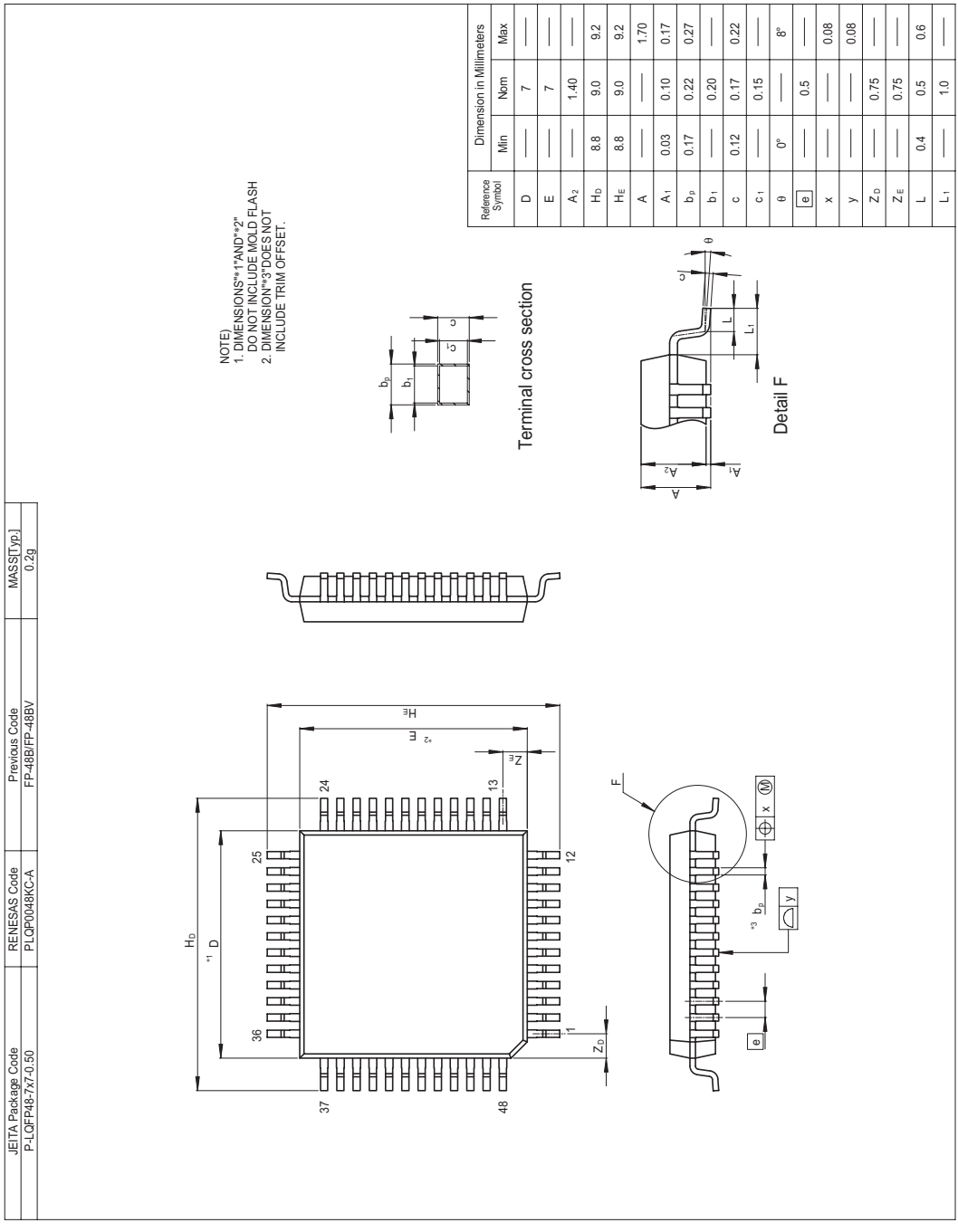
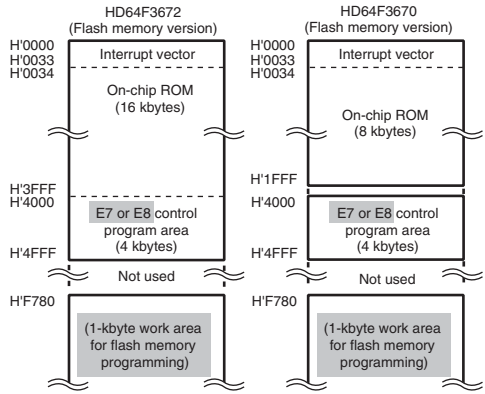
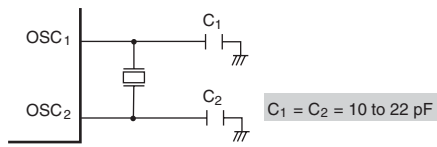
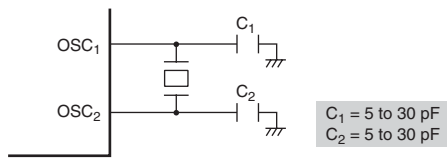
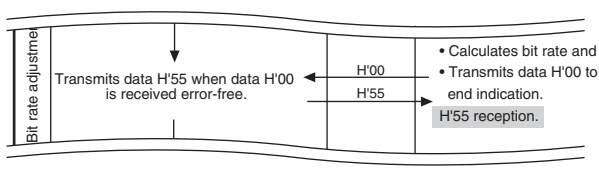



**Figure D.3 FP-48B Package Dimensions**

# Main Revisions and Additions in this Edition

| Item | Page | Revisions (See Manual for Details) |
|---|---|---|
| Preface | vi, vii | When using the on-chip emulator (E7, E8) for H8/3672 program development and debugging, the following restrictions must be noted.<br>1. The $\overline{\text{NMI}}$ pin is reserved for the E7 or E8, and cannot be used.<br>2. Area H'4000 to H'4FFF is used by the E7 or E8, and is not available to the user.<br>4. When the E7 or E8 is used, address breaks can be set as either available to the user or for use by the E7 or E8. If address breaks are set as being used by the E7 or E8, the address break control registers must not be accessed.<br>5. When the E7 or E8 is used, $\overline{\text{NMI}}$ is an input/output pin (open-drain in output mode).<br>Note has been deleted. |
| Section 1 Overview<br>Figure 1.1 Internal Block Diagram<br>Figure 1.2 Pin Arrangement (FP-64E)<br>Figure 1.3 Pin Arrangement (FP-48F, FP-48B) | 2 to 4 | Note: * Can also be used for the E7 or E8 emulator. |
| 1.4 Pin Functions<br>Table 1.1 Pin Functions | 6 | **Type** / **Functions**<br>E10T / Interface pin for the E10T, E7, or E8 emulator |

| Item | Page | Revisions (See Manual for Details) |
|---|---|---|
| Section 2 CPU<br>Figure 2.1 Memory Map | 8 |  <br>Note has been deleted. |
| Section 5 Clock Pulse Generators<br>Figure 5.3 Typical Connection to Crystal Resonator | 66 |  |
| Figure 5.5 Typical Connection to Ceramic Resonator | 66 |  |
| Section 7 ROM | 79 | The features of the 20-kbyte (4 kbytes of them are the E7 or E8 control program area) flash memory built into HD64F3672 are summarized below. |
| Table 7.2 Boot Mode Operation | 85 |  |

| Item | Page | Revisions (See Manual for Details) |
|---|---|---|
| Section 8 RAM | 93 | Note has been added. |

Section 10 Timer V

10.3.4 Timer Control/Status Register V (TCSRV) — 120

| Bit | Bit Name | Description |
|---|---|---|
| 3 | OS3 | Output Select 3 and 2<br>These bits select an output method for the TMOV pin by the compare match of TCORB and TCNTV.<br>00: No change<br>01: 0 output<br>10: 1 output<br>11: Output toggles |
| 2 | OS2 | |

Section 12 Watchdog Timer

12.2.1 Timer Control/Status Register WD (TCSRWD) — 160

| Bit | Bit Name | Description |
|---|---|---|
| 4 | TCSRWE | Timer Control/Status Register WD Write Enable<br>: |

Section 14 A/D Converter

14.3.1 A/D Data Registers A to D (ADDRA to ADDRD) — 206

Therefore byte access to ADDR should be done by reading the upper byte first then the lower one. Word access is also possible. ADDR is initialized to H'0000.

Section 17 Electrical Characteristics

Table 17.2 DC Characteristics (1) — 230

| Item | Symbol | Applicable Pins | Test Condition | Values Min |
|---|---|---|---|---|
| Input high voltage | $V_{IH}$ | PB3 to PB0 | $V_{CC}$ = 4.0 V to 5.5 V | $V_{CC} \times 0.7$ |
| | | | | $V_{CC} \times 0.8$ |
| Input low voltage | $V_{IL}$ | RXD, P12 to P10, P17 to P14, P22 to P20, P57 to P50, P76 to P74, P84 to P80 | $V_{CC}$ = 4.0 V to 5.5 V | –0.3 |
| | | PB3 to PB0 | | –0.3 |

| Item | Page | Revisions (See Manual for Details) |
|---|---|---|
| Table 17.2 DC Characteristics (1) | 234 | |

| Mode | $\overline{RES}$ Pin | Internal State |
|---|---|---|
| Active mode 1 | $V_{CC}$ | Operates |
| Active mode 2 | | Operates (φOSC/64) |
| Sleep mode 1 | $V_{CC}$ | Only timers operate |
| Sleep mode 2 | | Only timers operate (φOSC/64) |

| | | |
|---|---|---|
| Appendix D Package Dimensions | 293 to 295 | Swapped with new ones. |

# Index


**A**

A/D Converter ..... 203
A/D conversion time ..... 210
external trigger input ..... 211
sample-and-hold circuit ..... 210
Scan Mode ..... 209
Single Mode ..... 209
Absolute Maximum Ratings ..... 227
Address Break ..... 55
Addressing Modes ..... 27
Absolute Address ..... 28
Immediate ..... 29
Memory Indirect ..... 29
Program-Counter Relative ..... 29
Register Direct ..... 27
Register Indirect ..... 28
Register Indirect with Displacement ..... 28
Register Indirect with Post-Increment ..... 28
Register Indirect with Pre-Decrement ..... 28

**C**

Clock Pulse Generators ..... 65
Condition Field ..... 26
Condition-Code Register (CCR) ..... 11
CPU ..... 7

**E**

Effective Address ..... 30
Effective Address Extension ..... 26
Electrical Characteristics ..... 227
AC Characteristics ..... 235
DC Characteristics ..... 230
Exception Handling ..... 41
NMI ..... 48
Reset Exception Handling ..... 48
Stack Status ..... 51
Trap Instruction ..... 41

**F**

flash memory ..... 79
Boot Mode ..... 84
boot program ..... 83
Erase/Erase-Verify ..... 89
erasing units ..... 79
Error Protection ..... 92
Hardware Protection ..... 92
Program/Program-Verify ..... 87
programming units ..... 79
Programming/Erasing in User Program Mode ..... 86
Software Protection ..... 92

**G**

General Registers ..... 10

**I**

I/O Ports ..... 95
I/O Port Block Diagrams ..... 275
Instruction Set ..... 16
Arithmetic Operations Instructions ..... 18, 19
Bit Manipulation Instructions ..... 21, 22
Block Data Transfer Instructions ..... 25
Branch Instructions ..... 23
Data Transfer Instructions ..... 17
Logic Operations Instructions ..... 20
Shift Instructions ..... 20
System Control Instructions ..... 24

Internal Power Supply
Step-Down Circuit ..... 215
Interrupt
Internal Interrupts ..... 49
Interrupt Response Time ..... 51
IRQ3 to IRQ0 Interrupts ..... 48
NMI interrupt ..... 48
WKP5 to WKP0 Interrupts ..... 49

## L

large current ports ..... 1

## M

Memory Map ..... 8
Module Standby Function ..... 77

## O

On-Board Programming Modes ..... 83
Operation Field ..... 26

## P

Package ..... 1
Package Dimensions ..... 293
Pin Arrangement ..... 3
Power-down Modes ..... 69
Sleep Mode ..... 76
Standby Mode ..... 76
Subsleep Mode ..... 76
Prescaler S ..... 67
Product Code Lineup ..... 292
Program Counter (PC) ..... 11
PWM Operation ..... 146

## R

Register
ABRKCR ..... 56, 219, 222, 225
ABRKSR ..... 57, 219, 222, 225
ADCR ..... 208, 219, 222, 225
ADCSR ..... 207, 219, 222, 225
ADDRA ..... 206, 219, 222, 225
ADDRB ..... 206, 219, 222, 225
ADDRC ..... 206, 219, 222, 225
ADDRD ..... 206, 219, 222, 225
BARH ..... 58, 219, 222, 225
BARL ..... 58, 219, 222, 225
BDRH ..... 58, 219, 222, 225
BDRL ..... 58, 219, 222, 225
BRR ..... 172, 218, 221, 224
EBR1 ..... 82, 218, 221, 224
FENR ..... 83, 218, 221, 224
FLMCR1 ..... 81, 218, 221, 224
FLMCR2 ..... 82, 218, 221, 224
GRA ..... 141, 218, 221, 224
GRB ..... 141, 218, 221, 224
GRC ..... 141, 218, 221, 224
GRD ..... 141, 218, 221, 224
IEGR1 ..... 43, 220, 223, 226
IEGR2 ..... 44, 220, 223, 226
IENR1 ..... 45, 220, 223, 226
IRR1 ..... 46, 220, 223, 226
IWPR ..... 47, 220, 223, 226
MSTCR1 ..... 73, 220, 223, 226
PCR1 ..... 97, 220, 223, 225
PCR2 ..... 100, 220, 223, 225
PCR5 ..... 104, 220, 223, 225
PCR7 ..... 108, 220, 223, 225
PCR8 ..... 110, 220, 223, 225
PDR1 ..... 97, 219, 222, 225
PDR2 ..... 101, 219, 222, 225
PDR5 ..... 104, 219, 222, 225
PDR7 ..... 108, 219, 222, 225
PDR8 ..... 111, 219, 222, 225
PDRB ..... 114, 220, 223, 225

PMR1 ..... 96, 220, 223, 225
PMR5 ..... 103, 220, 223, 225
PUCR1 ..... 98, 219, 222, 225
PUCR5 ..... 105, 219, 222, 225
RDR ..... 166, 219, 222, 224
RSR ..... 166
SCR3 ..... 168, 218, 222, 224
SMR ..... 167, 218, 221, 224
SSR ..... 170, 219, 222, 224
SYSCR1 ..... 70, 220, 223, 226
SYSCR2 ..... 72, 220, 223, 226
TCNT ..... 141, 218, 221, 224
TCNTV ..... 117, 218, 221, 224
TCORA ..... 118, 218, 221, 224
TCORB ..... 118, 218, 221, 224
TCRV0 ..... 118, 218, 221, 224
TCRV1 ..... 121, 218, 221, 224
TCRW ..... 136, 218, 221, 224
TCSRV ..... 120, 218, 221, 224
TCSRWD ..... 160, 219, 222, 225
TCWD ..... 161, 219, 222, 225
TDR ..... 166, 219, 222, 224
TIERW ..... 137, 218, 221, 224
TIOR0 ..... 139, 218, 221, 224
TIOR1 ..... 140, 218, 221, 224
TMRW ..... 135, 218, 221, 224
TMWD ..... 161, 219, 222, 225
TSR ..... 166
TSRW ..... 137, 218, 221, 224
Register Field ..... 26

## S

Serial Communication Interface 3 (SCI3) ..... 163
Asynchronous Mode ..... 177
bit rate ..... 172
Break Detection ..... 201
Clocked Synchronous Mode ..... 185
framing error ..... 181
Mark State ..... 201
Multiprocessor Communication Function ..... 193
overrun error ..... 181
parity error ..... 181
Stack Pointer (SP) ..... 11
System Clock Generator ..... 65

## T

Timer V ..... 115
Timer W ..... 131

## V

Vector Address ..... 42

## W

Watchdog Timer ..... 159

**Renesas 16-Bit Single-Chip Microcomputer**
**Hardware Manual**
**H8/3672 Group**

Publication Date: 1st Edition, Mar, 2001
Rev.4.00, Nov. 02, 2005
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Customer Support Department
Global Strategic Communication Div.
Renesas Solutions Corp.

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, United Kingdom
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

**Renesas Technology (Shanghai) Co., Ltd.**
Unit2607 Ruijing Building, No.205 Maoming Road (S), Shanghai 200020, China
Tel: <86> (21) 6472-1001, Fax: <86> (21) 6415-2952

**Renesas Technology Hong Kong Ltd.**
7th Floor, North Tower, World Finance Centre, Harbour City, 1 Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2730-6071

**Renesas Technology Taiwan Co., Ltd.**
10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 2713-2999

**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, Jalan Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

# H8/3672 Group
# Hardware Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan