To all our customers

## Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

## Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation 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 Corporation 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 Corporation or a third party.
2. Renesas Technology Corporation 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation for further details on these materials or the products contained therein.

Hitachi 16-bit Single-Chip Microcomputer

# H8S/2628 Series

## H8S/2628
## HD64F2628, HD6432628
## H8S/2627
## HD6432627

Hardware Manual

ADE-602-278

Rev. 1.0
09/13/02
Hitachi, Ltd.

## Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

## 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 H8S/2628 Series are single-chip microcomputers made up of the high-speed H8S/2600 CPU as its core, and the peripheral functions required to configure a system. The H8S/2600 CPU has an instruction set that is compatible with the H8/300 and H8/300H CPUs.

This LSI is equipped with a data transfer controller (DTC), ROM and RAM memory, a PC break controller (PBC), a 16-bit timer pulse unit (TPU), a programmable pulse generator (PPG), a watchdog timer (WDT), a serial communication interface (SCI), a Hitachi controller area network (HCAN), a synchronous serial communication unit (SSU), an A/D converter, and I/O ports as on-chip peripheral modules required for system configuration. This LSI is suitable for use as an embedded microcomputer for high-level control systems. A single-power flash memory (F-ZTAT™) version is available for this LSI's ROM. This provides flexibility as it can be reprogrammed in no time to cope with all situations from the early stages of mass production to full-scale mass production. This is particularly applicable to application devices with specifications that will most probably change.

Note:* F-ZTAT™ is a trademark of Hitachi, Ltd.

Target Users: This manual was written for users who will be using the H8S/2628 Series 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 H8S/2628 Series to the target users.
Refer to the H8S/2600 Series Programming 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 Programming 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 22, List of Registers.

Example: Register name: The following notation is used for cases when the same or a similar function, e.g. 16-bit timer pulse unit or serial communication, is implemented on more than one channel: XXX_N (XXX is the register name and N is the channel number)

Bit order: The MSB is on the left and the LSB is on the right.

Notes:

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.hitachisemiconductor.com/

H8S/2628 Series manuals:

| Manual Title | ADE No. |
|---|---|
| H8S/2628 Series Hardware Manual | This manual |
| H8S/2600 Series, H8S/2000 Series Programming Manual | ADE-602-083 |

User's manuals for development tools:

| Manual Title | ADE No. |
|---|---|
| H8S, H8/300 Series C/C++ Compiler, Assembler, Optimizing Linkage Editor User's Manual | ADE-702-247 |
| H8S, H8/300 Series Simulator/Debugger User's Manual | ADE-702-282 |
| H8S, H8/300 Series Hitachi Embedded Workshop, Hitachi Debugging Interface Tutorial | ADE-702-231 |
| Hitachi Embedded Workshop User's Manual | ADE-702-201 |

# Contents


Section 1 Overview ..... 1
1.1 Overview ..... 1
1.2 Internal Block Diagram ..... 2
1.3 Pin Arrangement ..... 3
1.4 Pin Functions ..... 4

Section 2 CPU ..... 9
2.1 Features ..... 9
2.1.1 Differences between H8S/2600 CPU and H8S/2000 CPU ..... 10
2.1.2 Differences from H8/300 CPU ..... 11
2.1.3 Differences from H8/300H CPU ..... 11
2.2 CPU Operating Modes ..... 12
2.2.1 Normal Mode ..... 12
2.2.2 Advanced Mode ..... 13
2.3 Address Space ..... 16
2.4 Registers ..... 17
2.4.1 General Registers ..... 18
2.4.2 Program Counter (PC) ..... 19
2.4.3 Extended Control Register (EXR) ..... 19
2.4.4 Condition-Code Register (CCR) ..... 20
2.4.5 Multiply-Accumulate Register (MAC) ..... 21
2.4.6 Initial Values of CPU Registers ..... 21
2.5 Data Formats ..... 22
2.5.1 General Register Data Formats ..... 22
2.5.2 Memory Data Formats ..... 24
2.6 Instruction Set ..... 25
2.6.1 Table of Instructions Classified by Function ..... 26
2.6.2 Basic Instruction Formats ..... 36
2.7 Addressing Modes and Effective Address Calculation ..... 38
2.7.1 Register Direct—Rn ..... 38
2.7.2 Register Indirect—@ERn ..... 38
2.7.3 Register Indirect with Displacement—@(d:16, ERn) or @(d:32, ERn) ..... 38
2.7.4 Register Indirect with Post-Increment or Pre-Decrement—@ERn+ or @-ERn ..... 39
2.7.5 Absolute Address—@aa:8, @aa:16, @aa:24, or @aa:32 ..... 39
2.7.6 Immediate—#xx:8, #xx:16, or #xx:32 ..... 40
2.7.7 Program-Counter Relative—@(d:8, PC) or @(d:16, PC) ..... 40
2.7.8 Memory Indirect—@@aa:8 ..... 40
2.7.9 Effective Address Calculation ..... 41
2.8 Processing States ..... 44

2.9 Usage Note ..... 45
2.9.1 Notes on Using the Bit Operation Instruction ..... 45

Section 3 MCU Operating Modes ..... 47
3.1 Operating Mode Selection ..... 47
3.2 Register Descriptions ..... 47
3.2.1 Mode Control Register (MDCR) ..... 48
3.2.2 System Control Register (SYSCR) ..... 49
3.3 Pin Functions in Each Operating Mode ..... 50
3.4 Address Map ..... 51

Section 4 Exception Handling ..... 53
4.1 Exception Handling Types and Priority ..... 53
4.2 Exception Sources and Exception Vector Table ..... 53
4.3 Reset ..... 55
4.3.1 Reset Exception Handling ..... 55
4.3.2 Interrupts after Reset ..... 57
4.3.3 State of On-Chip Peripheral Modules after Reset Release ..... 57
4.4 Traces ..... 58
4.5 Interrupts ..... 58
4.6 Trap Instruction ..... 59
4.7 Stack Status after Exception Handling ..... 60
4.8 Usage Note ..... 61

Section 5 Interrupt Controller ..... 63
5.1 Features ..... 63
5.2 Input/Output Pins ..... 65
5.3 Register Descriptions ..... 65
5.3.1 Interrupt Priority Registers A to M (IPRA to IPRM) ..... 66
5.3.2 IRQ Enable Register (IER) ..... 67
5.3.3 IRQ Sense Control Registers H and L (ISCRH, ISCRL) ..... 68
5.3.4 IRQ Status Register (ISR) ..... 70
5.4 Interrupt Sources ..... 71
5.4.1 External Interrupts ..... 71
5.4.2 Internal Interrupts ..... 72
5.5 Interrupt Exception Handling Vector Table ..... 72
5.6 Interrupt Control Modes and Interrupt Operation ..... 76
5.6.1 Interrupt Control Mode 0 ..... 76
5.6.2 Interrupt Control Mode 2 ..... 78
5.6.3 Interrupt Exception Handling Sequence ..... 79
5.6.4 Interrupt Response Times ..... 81
5.6.5 DTC Activation by Interrupt ..... 82
5.7 Usage Notes ..... 82

5.7.1 Conflict between Interrupt Generation and Disabling ..... 82
5.7.2 Instructions that Disable Interrupts ..... 83
5.7.3 When Interrupts Are Disabled ..... 83
5.7.4 Interrupts during Execution of EEPMOV Instruction ..... 84

Section 6 PC Break Controller (PBC) ..... 85
6.1 Features ..... 85
6.2 Register Descriptions ..... 86
6.2.1 Break Address Register A (BARA) ..... 86
6.2.2 Break Address Register B (BARB) ..... 87
6.2.3 Break Control Register A (BCRA) ..... 87
6.2.4 Break Control Register B (BCRB) ..... 88
6.3 Operation ..... 88
6.3.1 PC Break Interrupt Due to Instruction Fetch ..... 88
6.3.2 PC Break Interrupt Due to Data Access ..... 88
6.3.3 PC Break Operation at Consecutive Data Transfer ..... 89
6.3.4 Operation in Transitions to Power-Down Modes ..... 89
6.3.5 When Instruction Execution Is Delayed by One State ..... 90
6.4 Usage Notes ..... 91
6.4.1 Module Stop Mode Setting ..... 91
6.4.2 PC Break Interrupts ..... 91
6.4.3 CMFA and CMFB ..... 91
6.4.4 PC Break Interrupt when DTC Is Bus Master ..... 91
6.4.5 PC Break Set for Instruction Fetch at Address Following BSR, JSR, JMP, TRAPA, RTE, or RTS Instruction ..... 91
6.4.6 I Bit Set by LDC, ANDC, ORC, or XORC Instruction ..... 91
6.4.7 PC Break Set for Instruction Fetch at Address Following Bcc Instruction ..... 92
6.4.8 PC Break Set for Instruction Fetch at Branch Destination Address of Bcc Instruction ..... 92

Section 7 Bus Controller ..... 93
7.1 Basic Timing ..... 93
7.1.1 On-Chip Memory Access Timing (ROM, RAM) ..... 93
7.1.2 On-Chip Support Module Access Timing ..... 94
7.1.3 On-Chip HCAN Module Access Timing ..... 94
7.1.4 On-chip SSU Module and Realtime Input Port Data Register Access Timing ..... 95
7.2 Bus Arbitration ..... 95
7.2.1 Order of Priority of the Bus Masters ..... 95
7.2.2 Bus Transfer Timing ..... 96

Section 8 Data Transfer Controller (DTC) ..... 97
8.1 Features ..... 97
8.2 Register Descriptions ..... 99

8.2.1 DTC Mode Register A (MRA) ..... 100
8.2.2 DTC Mode Register B (MRB) ..... 101
8.2.3 DTC Source Address Register (SAR) ..... 101
8.2.4 DTC Destination Address Register (DAR) ..... 101
8.2.5 DTC Transfer Count Register A (CRA) ..... 101
8.2.6 DTC Transfer Count Register B (CRB) ..... 102
8.2.7 DTC Enable Registers (DTCER) ..... 102
8.2.8 DTC Vector Register (DTVECR) ..... 103
8.3 Activation Sources ..... 103
8.4 Location of Register Information and DTC Vector Table ..... 104
8.5 Operation ..... 107
8.5.1 Normal Mode ..... 109
8.5.2 Repeat Mode ..... 110
8.5.3 Block Transfer Mode ..... 111
8.5.4 Chain Transfer ..... 112
8.5.5 Interrupts ..... 113
8.5.6 Operation Timing ..... 113
8.5.7 Number of DTC Execution States ..... 114
8.6 Procedures for Using DTC ..... 116
8.6.1 Activation by Interrupt ..... 116
8.6.2 Activation by Software ..... 116
8.7 Examples of Use of the DTC ..... 116
8.7.1 Normal Mode ..... 116
8.7.2 Chain Transfer ..... 117
8.7.3 Software Activation ..... 118
8.8 Usage Notes ..... 118
8.8.1 Module Stop Mode Setting ..... 118
8.8.2 On-Chip RAM ..... 119
8.8.3 DTCE Bit Setting ..... 119

Section 9 I/O Ports ..... 121
9.1 Port 1 ..... 125
9.1.1 Port 1 Data Direction Register (P1DDR) ..... 125
9.1.2 Port 1 Data Register (P1DR) ..... 126
9.1.3 Port 1 Register (PORT1) ..... 126
9.1.4 Pin Functions ..... 127
9.2 Port 3 ..... 129
9.2.1 Port 3 Data Direction Register (P3DDR) ..... 129
9.2.2 Port 3 Data Register (P3DR) ..... 130
9.2.3 Port 3 Register (PORT3) ..... 130
9.2.4 Port 3 Open-Drain Control Register (P3ODR) ..... 131
9.2.5 Pin Functions ..... 131
9.3 Port 4 ..... 133

9.3.1 Port 4 Register (PORT4) ..... 133
9.4 Port 7 ..... 133
9.4.1 Port 7 Data Direction Register (P7DDR) ..... 134
9.4.2 Port 7 Data Register (P7DR) ..... 134
9.4.3 Port 7 Register (PORT7) ..... 134
9.4.4 Pin Functions ..... 135
9.5 Port 9 ..... 136
9.5.1 Port 9 Register (PORT9) ..... 136
9.6 Port A ..... 138
9.6.1 Port A Data Direction Register (PADDR) ..... 138
9.6.2 Port A Data Register (PADR) ..... 139
9.6.3 Port A Register (PORTA) ..... 139
9.6.4 Port A Pull-Up MOS Control Register (PAPCR) ..... 140
9.6.5 Port A Open-Drain Control Register (PAODR) ..... 140
9.6.6 Pin Functions ..... 141
9.7 Port B ..... 142
9.7.1 Port B Data Direction Register (PBDDR) ..... 142
9.7.2 Port B Data Register (PBDR) ..... 143
9.7.3 Port B Register (PORTB) ..... 143
9.7.4 Port B Pull-Up MOS Control Register (PBPCR) ..... 144
9.7.5 Port B Open-Drain Control Register (PBODR) ..... 144
9.7.6 Pin Functions ..... 145
9.8 Port C ..... 146
9.8.1 Port C Data Direction Register (PCDDR) ..... 147
9.8.2 Port C Data Register (PCDR) ..... 147
9.8.3 Port C Register (PORTC) ..... 148
9.8.4 Port C Pull-Up MOS Control Register (PCPCR) ..... 148
9.8.5 Port C Open-Drain Control Register (PCODR) ..... 148
9.8.6 Pin Functions ..... 149
9.9 Port D ..... 152
9.9.1 Port D Data Direction Register (PDDDR) ..... 152
9.9.2 Port D Data Register (PDDR) ..... 153
9.9.3 Port D Register (PORTD) ..... 153
9.9.4 Port D Pull-up MOS Control Register (PDPCR) ..... 153
9.9.5 Port D RealTime Input Data Register (PDRTIDR) ..... 154
9.10 Port F ..... 154
9.10.1 Port F Data Direction Register (PFDDR) ..... 155
9.10.2 Port F Data Register (PFDR) ..... 156
9.10.3 Port F Register (PORTF) ..... 156
9.10.4 Pin Functions ..... 157

Section 10 16-Bit Timer Pulse Unit (TPU) ..... 159
10.1 Features ..... 159

10.2 Input/Output Pins ..... 163
10.3 Register Descriptions ..... 164
10.3.1 Timer Control Register (TCR) ..... 166
10.3.2 Timer Mode Register (TMDR) ..... 171
10.3.3 Timer I/O Control Register (TIOR) ..... 173
10.3.4 Timer Interrupt Enable Register (TIER) ..... 190
10.3.5 Timer Status Register (TSR) ..... 192
10.3.6 Timer Counter (TCNT) ..... 195
10.3.7 Timer General Register (TGR) ..... 195
10.3.8 Timer Start Register (TSTR) ..... 195
10.3.9 Timer Synchro Register (TSYR) ..... 196
10.4 Operation ..... 196
10.4.1 Basic Functions ..... 196
10.4.2 Synchronous Operation ..... 203
10.4.3 Buffer Operation ..... 204
10.4.4 Cascaded Operation ..... 208
10.4.5 PWM Modes ..... 209
10.4.6 Phase Counting Mode ..... 214
10.5 Interrupt Sources ..... 221
10.6 DTC Activation ..... 223
10.7 A/D Converter Activation ..... 223
10.8 Operation Timing ..... 224
10.8.1 Input/Output Timing ..... 224
10.8.2 Interrupt Signal Timing ..... 228
10.9 Usage Notes ..... 231
10.9.1 Module Stop Mode Setting ..... 231
10.9.2 Input Clock Restrictions ..... 231
10.9.3 Caution on Period Setting ..... 231
10.9.4 Conflict between TCNT Write and Clear Operations ..... 232
10.9.5 Conflict between TCNT Write and Increment Operations ..... 233
10.9.6 Conflict between TGR Write and Compare Match ..... 234
10.9.7 Conflict between Buffer Register Write and Compare Match ..... 235
10.9.8 Conflict between TGR Read and Input Capture ..... 236
10.9.9 Conflict between TGR Write and Input Capture ..... 237
10.9.10 Conflict between Buffer Register Write and Input Capture ..... 238
10.9.11 Conflict between Overflow/Underflow and Counter Clearing ..... 239
10.9.12 Conflict between TCNT Write and Overflow/Underflow ..... 240
10.9.13 Multiplexing of I/O Pins ..... 240
10.9.14 Interrupts in Module Stop Mode ..... 240

Section 11 8-Bit Timers ..... 241
11.1 Features ..... 241
11.2 Input/Output Pins ..... 242

11.3 Register Descriptions ..... 243
11.3.1 Timer Counters (TCNT) ..... 244
11.3.2 Time Constant Registers A (TCORA) ..... 244
11.3.3 Time Constant Registers B (TCORB) ..... 244
11.3.4 Timer Control Registers (TCR) ..... 244
11.3.5 Timer Control/Status Registers (TCSR) ..... 247
11.4 Operation ..... 251
11.4.1 Pulse Output ..... 251
11.5 Operation Timing ..... 252
11.5.1 TCNT Incrementation Timing ..... 252
11.5.2 Timing of CMFA and CMFB Setting When a Compare-Match Occurs ..... 253
11.5.3 Timing of Timer Output When a Compare-Match Occurs ..... 254
11.5.4 Timing of Compare-Match Clear When a Compare-Match Occurs ..... 254
11.5.5 TCNT External Reset Timing ..... 254
11.5.6 Timing of Overflow Flag (OVF) Setting ..... 255
11.6 Operation with Cascaded Connection ..... 255
11.6.1 16-Bit Count Mode ..... 255
11.6.2 Compare-Match Count Mode ..... 256
11.7 Interrupt Sources ..... 256
11.7.1 Interrupt Sources and DTC Activation ..... 256
11.7.2 A/D Converter Activation ..... 257
11.8 Usage Notes ..... 258
11.8.1 Conflict between TCNT Write and Clear ..... 258
11.8.2 Conflict between TCNT Write and Increment ..... 258
11.8.3 Conflict between TCOR Write and Compare-Match ..... 259
11.8.4 Conflict between Compare-Matches A and B ..... 260
11.8.5 Switching of Internal Clocks and TCNT Operation ..... 260
11.8.6 Conflict between Interrupts and Module Stop Mode ..... 262
11.8.7 Notes on Cascaded Connection ..... 262

Section 12 Programmable Pulse Generator (PPG) ..... 263
12.1 Features ..... 263
12.2 Input/Output Pins ..... 265
12.3 Register Descriptions ..... 265
12.3.1 Next Data Enable Registers H, L (NDERH, NDERL) ..... 266
12.3.2 Output Data Registers H, L (PODRH, PODRL) ..... 267
12.3.3 Next Data Registers H, L (NDRH, NDRL) ..... 268
12.3.4 PPG Output Control Register (PCR) ..... 270
12.3.5 PPG Output Mode Register (PMR) ..... 271
12.4 Operation ..... 272
12.4.1 Overview ..... 272
12.4.2 Output Timing ..... 273
12.4.3 Sample Setup Procedure for Normal Pulse Output ..... 274

12.4.4 Example of Normal Pulse Output (Example of Five-Phase Pulse Output) ..... 275
12.4.5 Non-Overlapping Pulse Output ..... 276
12.4.6 Sample Setup Procedure for Non-Overlapping Pulse Output ..... 278
12.4.7 Example of Non-Overlapping Pulse Output (Example of Four-Phase Complementary Non-Overlapping Output) ..... 279
12.4.8 Inverted Pulse Output ..... 281
12.4.9 Pulse Output Triggered by Input Capture ..... 282
12.5 Usage Notes ..... 282
12.5.1 Module Stop Mode Setting ..... 282
12.5.2 Operation of Pulse Output Pins ..... 282

Section 13 Watchdog Timer ..... 283
13.1 Features ..... 283
13.2 Register Descriptions ..... 284
13.2.1 Timer Counter (TCNT) ..... 284
13.2.2 Timer Control/Status Register (TCSR) ..... 284
13.2.3 Reset Control/Status Register (RSTCSR) ..... 286
13.3 Operation ..... 287
13.3.1 Watchdog Timer Mode Operation ..... 287
13.3.2 Interval Timer Mode ..... 287
13.4 Interrupts ..... 288
13.5 Usage Notes ..... 288
13.5.1 Notes on Register Access ..... 288
13.5.2 Conflict between Timer Counter (TCNT) Write and Increment ..... 289
13.5.3 Changing Value of CKS2 to CKS0 ..... 290
13.5.4 Switching between Watchdog Timer Mode and Interval Timer Mode ..... 290
13.5.5 Internal Reset in Watchdog Timer Mode ..... 290
13.5.6 OVF Flag Clearing in Interval Timer Mode ..... 290

Section 14 Serial Communication Interface (SCI) ..... 291
14.1 Features ..... 291
14.2 Input/Output Pins ..... 293
14.3 Register Descriptions ..... 293
14.3.1 Receive Shift Register (RSR) ..... 294
14.3.2 Receive Data Register (RDR) ..... 294
14.3.3 Transmit Data Register (TDR) ..... 294
14.3.4 Transmit Shift Register (TSR) ..... 294
14.3.5 Serial Mode Register (SMR) ..... 295
14.3.6 Serial Control Register (SCR) ..... 299
14.3.7 Serial Status Register (SSR) ..... 302
14.3.8 Smart Card Mode Register (SCMR) ..... 307
14.3.9 Bit Rate Register (BRR) ..... 308
14.4 Operation in Asynchronous Mode ..... 315

14.4.1 Data Transfer Format ..... 315
14.4.2 Receive Data Sampling Timing and Reception Margin in Asynchronous Mode 317
14.4.3 Clock ..... 318
14.4.4 SCI Initialization (Asynchronous Mode) ..... 319
14.4.5 Data Transmission (Asynchronous Mode) ..... 320
14.4.6 Serial Data Reception (Asynchronous Mode) ..... 322
14.5 Multiprocessor Communication Function ..... 326
14.5.1 Multiprocessor Serial Data Transmission ..... 328
14.5.2 Multiprocessor Serial Data Reception ..... 329
14.6 Operation in Clocked Synchronous Mode ..... 332
14.6.1 Clock ..... 332
14.6.2 SCI Initialization (Clocked Synchronous Mode) ..... 333
14.6.3 Serial Data Transmission (Clocked Synchronous Mode) ..... 334
14.6.4 Serial Data Reception (Clocked Synchronous Mode) ..... 336
14.6.5 Simultaneous Serial Data Transmission and Reception (Clocked Synchronous Mode) ..... 338
14.7 Operation in Smart Card Interface ..... 340
14.7.1 Pin Connection Example ..... 340
14.7.2 Data Format (Except for Block Transfer Mode) ..... 341
14.7.3 Block Transfer Mode ..... 342
14.7.4 Receive Data Sampling Timing and Reception Margin in Smart Card Interface Mode ..... 343
14.7.5 Initialization ..... 344
14.7.6 Data Transmission (Except for Block Transfer Mode) ..... 344
14.7.7 Serial Data Reception (Except for Block Transfer Mode) ..... 348
14.7.8 Clock Output Control ..... 349
14.8 Interrupt Sources ..... 351
14.8.1 Interrupts in Normal Serial Communication Interface Mode ..... 351
14.8.2 Interrupts in Smart Card Interface Mode ..... 352
14.9 Usage Notes ..... 353
14.9.1 Module Stop Mode Setting ..... 353
14.9.2 Break Detection and Processing ..... 353
14.9.3 Mark State and Break Detection ..... 353
14.9.4 Receive Error Flags and Transmit Operations (Clocked Synchronous Mode Only) ..... 353

Section 15 Hitachi Controller Area Network (HCAN) ..... 355
15.1 Features ..... 355
15.2 Input/Output Pins ..... 357
15.3 Register Descriptions ..... 357
15.3.1 Master Control Register (MCR) ..... 358
15.3.2 General Status Register (GSR) ..... 359
15.3.3 Bit Configuration Register (BCR) ..... 361

15.3.4 Mailbox Configuration Register (MBCR) ..... 363
15.3.5 Transmit Wait Register (TXPR) ..... 364
15.3.6 Transmit Wait Cancel Register (TXCR) ..... 365
15.3.7 Transmit Acknowledge Register (TXACK) ..... 366
15.3.8 Abort Acknowledge Register (ABACK) ..... 367
15.3.9 Receive Complete Register (RXPR) ..... 368
15.3.10 Remote Request Register (RFPR) ..... 369
15.3.11 Interrupt Register (IRR) ..... 370
15.3.12 Mailbox Interrupt Mask Register (MBIMR) ..... 374
15.3.13 Interrupt Mask Register (IMR) ..... 375
15.3.14 Receive Error Counter (REC) ..... 376
15.3.15 Transmit Error Counter (TEC) ..... 376
15.3.16 Unread Message Status Register (UMSR) ..... 377
15.3.17 Local Acceptance Filter Masks (LAFML, LAFMH) ..... 378
15.3.18 Message Control (MC0 to MC15) ..... 380
15.3.19 Message Data (MD0 to MD15) ..... 382
15.3.20 HCAN Monitor Register (HCANMON) ..... 382
15.4 Operation ..... 384
15.4.1 Hardware and Software Resets ..... 384
15.4.2 Initialization after Hardware Reset ..... 384
15.4.3 Message Transmission ..... 390
15.4.4 Message Reception ..... 393
15.4.5 HCAN Sleep Mode ..... 396
15.4.6 HCAN Halt Mode ..... 399
15.5 Interrupt Sources ..... 400
15.6 DTC Interface ..... 401
15.7 CAN Bus Interface ..... 402
15.8 Usage Notes ..... 402
15.8.1 Module Stop Mode Setting ..... 402
15.8.2 Reset ..... 402
15.8.3 HCAN Sleep Mode ..... 403
15.8.4 Interrupts ..... 403
15.8.5 Error Counters ..... 403
15.8.6 Register Access ..... 403
15.8.7 HCAN Medium-Speed Mode ..... 403
15.8.8 Register Hold in Standby Modes ..... 403
15.8.9 Use on Bit Manipulation Instructions ..... 403
15.8.10 HCAN TXCR Operation ..... 404

Section 16 Synchronous Serial Communication Unit (SSU) ..... 405
16.1 Features ..... 405
16.2 Input/Output Pins ..... 407
16.3 Register Descriptions ..... 407

16.3.1 SS Control Register H (SSCRH) ..... 407
16.3.2 SS Control Register L (SSCRL) ..... 409
16.3.3 SS Mode Register (SSMR) ..... 410
16.3.4 SS Enable Register (SSER) ..... 411
16.3.5 SS Status Register (SSSR) ..... 412
16.3.6 SS Transmit Data Register 0 to 3 (SSTDR0 to SSTDR3) ..... 415
16.3.7 SS Receive Data Register 0 to 3 (SSRDR0 to SSRDR3) ..... 415
16.3.8 SS Shift Register (SSTRSR) ..... 415
16.4 Operation ..... 416
16.4.1 Transfer Clock ..... 416
16.4.2 Relationship of Clock Phase, Polarity, and Data ..... 416
16.4.3 Relationship between Data I/O Pins and the Shift Register ..... 416
16.4.4 Data Transmission and Data Reception ..... 417
16.4.5 $\overline{\text{SCS}}$ Pin Control and Arbitration ..... 424
16.5 Interrupt Requests ..... 425
16.6 Usage Note ..... 426
16.6.1 Setting of Module Stop Mode ..... 426

Section 17 A/D Converter ..... 427
17.1 Features ..... 427
17.2 Input/Output Pins ..... 429
17.3 Register Description ..... 430
17.3.1 A/D Data Registers A to D (ADDRA to ADDRD) ..... 430
17.3.2 A/D Control/Status Register (ADCSR) ..... 431
17.3.3 A/D Control Register (ADCR) ..... 433
17.4 Operation ..... 434
17.4.1 Single Mode ..... 434
17.4.2 Scan Mode ..... 434
17.4.3 Input Sampling and A/D Conversion Time ..... 435
17.4.4 External Trigger Input Timing ..... 437
17.5 Interrupt Source ..... 437
17.6 A/D Conversion Accuracy Definitions ..... 438
17.7 Usage Notes ..... 440
17.7.1 Module Stop Mode Setting ..... 440
17.7.2 Permissible Signal Source Impedance ..... 440
17.7.3 Influences on Absolute Accuracy ..... 440
17.7.4 Range of Analog Power Supply and Other Pin Settings ..... 441
17.7.5 Notes on Board Design ..... 441
17.7.6 Notes on Noise Countermeasures ..... 441

Section 18 RAM ..... 443

Section 19 ROM ..... 445

19.1 Features ..... 445
19.2 Mode Transitions ..... 446
19.3 Block Configuration ..... 450
19.4 Input/Output Pins ..... 451
19.5 Register Descriptions ..... 451
19.5.1 Flash Memory Control Register 1 (FLMCR1) ..... 452
19.5.2 Flash Memory Control Register 2 (FLMCR2) ..... 453
19.5.3 Erase Block Register 1 (EBR1) ..... 453
19.5.4 Erase Block Register 2 (EBR2) ..... 454
19.5.5 RAM Emulation Register (RAMER) ..... 454
19.6 On-Board Programming Modes ..... 455
19.6.1 Boot Mode ..... 456
19.6.2 Programming/Erasing in User Program Mode ..... 458
19.7 Flash Memory Emulation in RAM ..... 459
19.8 Flash Memory Programming/Erasing ..... 461
19.8.1 Program/Program-Verify ..... 461
19.8.2 Erase/Erase-Verify ..... 463
19.8.3 Interrupt Handling when Programming/Erasing Flash Memory ..... 463
19.9 Program/Erase Protection ..... 465
19.9.1 Hardware Protection ..... 465
19.9.2 Software Protection ..... 465
19.9.3 Error Protection ..... 465
19.10 Programmer Mode ..... 466
19.11 Power-Down States for Flash Memory ..... 466

Section 20 Clock Pulse Generator ..... 467
20.1 Register Descriptions ..... 468
20.1.1 System Clock Control Register (SCKCR) ..... 468
20.1.2 Low-Power Control Register (LPWRCR) ..... 469
20.2 Oscillator ..... 470
20.2.1 Connecting a Crystal Resonator ..... 470
20.2.2 External Clock Input ..... 471
20.3 PLL Circuit ..... 473
20.4 Medium-Speed Clock Divider ..... 473
20.5 Bus Master Clock Selection Circuit ..... 473
20.6 Usage Notes ..... 474
20.6.1 Note on Crystal Resonator ..... 474
20.6.2 Note on Board Design ..... 474

Section 21 Power-Down Modes ..... 477
21.1 Register Descriptions ..... 480
21.1.1 Standby Control Register (SBYCR) ..... 480
21.1.2 Module Stop Control Registers A to C (MSTPCRA to MSTPCRC) ..... 482

21.2 Medium-Speed Mode ..... 483
21.3 Sleep Mode ..... 484
21.3.1 Transition to Sleep Mode ..... 484
21.3.2 Clearing Sleep Mode ..... 484
21.4 Software Standby Mode ..... 485
21.4.1 Transition to Software Standby Mode ..... 485
21.4.2 Clearing Software Standby Mode ..... 485
21.4.3 Setting Oscillation Stabilization Time after Clearing Software Standby Mode ..... 486
21.4.4 Software Standby Mode Application Example ..... 487
21.5 Hardware Standby Mode ..... 488
21.5.1 Transition to Hardware Standby Mode ..... 488
21.5.2 Clearing Hardware Standby Mode ..... 488
21.5.3 Hardware Standby Mode Timings ..... 488
21.6 Module Stop Mode ..... 489
21.6.1 Module Stop Mode ..... 489
21.7 φ Clock Output Disabling Function ..... 490
21.8 Usage Notes ..... 490
21.8.1 I/O Port Status ..... 490
21.8.2 Current Consumption during Oscillation Stabilization Wait Period ..... 490
21.8.3 DTC Module Stop ..... 490
21.8.4 On-Chip Peripheral Module Interrupt ..... 490
21.8.5 Writing to MSTPCR ..... 491

Section 22 List of Registers ..... 493
22.1 Register Addresses (Address Order) ..... 494
22.2 Register Bits ..... 510
22.3 Register States in Each Operating Mode ..... 528

Section 23 Electrical Characteristics ..... 545
23.1 Absolute Maximum Ratings ..... 545
23.2 DC Characteristics ..... 546
23.3 AC Characteristics ..... 548
23.3.1 Clock Timing ..... 549
23.3.2 Control Signal Timing ..... 550
23.3.3 Timing of On-Chip Peripheral Modules ..... 552
23.4 A/D Conversion Characteristics ..... 560
23.5 Flash Memory Characteristics ..... 561

Appendix ..... 563
A. I/O Port States in Each Pin State ..... 563
B. Product Code Lineup ..... 564
C. Package Dimensions ..... 564

Index ..... 565

# Figures

**Section 1 Overview**
Figure 1.1 Internal Block Diagram ..... 2
Figure 1.2 Pin Arrangement ..... 3

**Section 2 CPU**
Figure 2.1 Exception Vector Table (Normal Mode) ..... 13
Figure 2.2 Stack Structure in Normal Mode ..... 13
Figure 2.3 Exception Vector Table (Advanced Mode) ..... 14
Figure 2.4 Stack Structure in Advanced Mode ..... 15
Figure 2.5 Memory Map ..... 16
Figure 2.6 CPU Registers ..... 17
Figure 2.7 Usage of General Registers ..... 18
Figure 2.8 Stack ..... 19
Figure 2.9 General Register Data Formats (1) ..... 22
Figure 2.9 General Register Data Formats (2) ..... 23
Figure 2.10 Memory Data Formats ..... 24
Figure 2.11 Instruction Formats (Examples) ..... 37
Figure 2.12 Branch Address Specification in Memory Indirect Mode ..... 41
Figure 2.13 State Transitions ..... 45

**Section 3 MCU Operating Modes**
Figure 3.1 Address Map ..... 51

**Section 4 Exception Handling**
Figure 4.1 Reset Sequence (Advanced Mode with On-chip ROM Enabled) ..... 56
Figure 4.2 Reset Sequence
(Advanced Mode with On-chip ROM Disabled: Not Available in this LSI) ..... 57
Figure 4.3 Stack Status after Exception Handling ..... 60
Figure 4.4 Operation when SP Value Is Odd ..... 61

**Section 5 Interrupt Controller**
Figure 5.1 Block Diagram of Interrupt Controller ..... 64
Figure 5.2 Block Diagram of Interrupts IRQ0 to IRQ5 ..... 71
Figure 5.3 Flowchart of Procedure up to Interrupt Acceptance in Interrupt Control Mode 0 ..... 77
Figure 5.4 Flowchart of Procedure Up to Interrupt Acceptance in Control Mode 2 ..... 79
Figure 5.5 Interrupt Exception Handling ..... 80
Figure 5.6 Conflict between Interrupt Generation and Disabling ..... 83

**Section 6 PC Break Controller (PBC)**
Figure 6.1 Block Diagram of PC Break Controller ..... 86

Figure 6.2 Operation in Power-Down Mode Transitions ..... 89

**Section 7 Bus Controller**

Figure 7.1 On-Chip Memory Access Cycle ..... 93
Figure 7.2 On-Chip Support Module Access Cycle ..... 94
Figure 7.3 On-Chip HCAN Module Access Cycle (with Wait States) ..... 94
Figure.7.4 On-chip SSU Module Access Cycle ..... 95

**Section 8 Data Transfer Controller (DTC)**

Figure 8.1 Block Diagram of DTC ..... 98
Figure 8.2 Block Diagram of DTC Activation Source Control ..... 104
Figure 8.3 Location of DTC Register Information in Address Space ..... 105
Figure 8.4 Flowchart of DTC Operation ..... 108
Figure 8.5 Memory Mapping in Normal Mode ..... 109
Figure 8.6 Memory Mapping in Repeat Mode ..... 110
Figure 8.7 Memory Mapping in Block Transfer Mode ..... 111
Figure 8.8 Chain Transfer Operation ..... 112
Figure 8.9 DTC Operation Timing (Example in Normal Mode or Repeat Mode) ..... 113
Figure 8.10 DTC Operation Timing
(Example of Block Transfer Mode, with Block Size of 2) ..... 114
Figure 8.11 DTC Operation Timing (Example of Chain Transfer) ..... 114

**Section 10 16-Bit Timer Pulse Unit (TPU)**

Figure 10.1 Block Diagram of TPU ..... 162
Figure 10.2 Example of Counter Operation Setting Procedure ..... 197
Figure 10.3 Free-Running Counter Operation ..... 198
Figure 10.4 Periodic Counter Operation ..... 199
Figure 10.5 Example of Setting Procedure for Waveform Output by Compare Match ..... 199
Figure 10.6 Example of 0 Output/1 Output Operation ..... 200
Figure 10.7 Example of Toggle Output Operation ..... 200
Figure 10.8 Example of Input Capture Operation Setting Procedure ..... 201
Figure 10.9 Example of Input Capture Operation ..... 202
Figure 10.10 Example of Synchronous Operation Setting Procedure ..... 203
Figure 10.11 Example of Synchronous Operation ..... 204
Figure 10.12 Compare Match Buffer Operation ..... 205
Figure 10.13 Input Capture Buffer Operation ..... 205
Figure 10.14 Example of Buffer Operation Setting Procedure ..... 206
Figure 10.15 Example of Buffer Operation (1) ..... 206
Figure 10.16 Example of Buffer Operation (2) ..... 207
Figure 10.17 Cascaded Operation Setting Procedure ..... 208
Figure 10.18 Example of Cascaded Operation (1) ..... 209
Figure 10.19 Example of Cascaded Operation (2) ..... 209
Figure 10.20 Example of PWM Mode Setting Procedure ..... 211

Figure 10.21 Example of PWM Mode Operation (1) ..... 212
Figure 10.22 Example of PWM Mode Operation (2) ..... 212
Figure 10.23 Example of PWM Mode Operation (3) ..... 213
Figure 10.24 Example of Phase Counting Mode Setting Procedure ..... 214
Figure 10.25 Example of Phase Counting Mode 1 Operation ..... 215
Figure 10.26 Example of Phase Counting Mode 2 Operation ..... 216
Figure 10.27 Example of Phase Counting Mode 3 Operation ..... 217
Figure 10.28 Example of Phase Counting Mode 4 Operation ..... 218
Figure 10.29 Phase Counting Mode Application Example ..... 220
Figure 10.30 Count Timing in Internal Clock Operation ..... 224
Figure 10.31 Count Timing in External Clock Operation ..... 224
Figure 10.32 Output Compare Output Timing ..... 225
Figure 10.33 Input Capture Input Signal Timing ..... 225
Figure 10.34 Counter Clear Timing (Compare Match) ..... 226
Figure 10.35 Counter Clear Timing (Input Capture) ..... 226
Figure 10.36 Buffer Operation Timing (Compare Match) ..... 227
Figure 10.37 Buffer Operation Timing (Input Capture) ..... 227
Figure 10.38 TGI Interrupt Timing (Compare Match) ..... 228
Figure 10.39 TGI Interrupt Timing (Input Capture) ..... 228
Figure 10.40 TCIV Interrupt Setting Timing ..... 229
Figure 10.41 TCIU Interrupt Setting Timing ..... 229
Figure 10.42 Timing for Status Flag Clearing by CPU ..... 230
Figure 10.43 Timing for Status Flag Clearing by DTC Activation ..... 230
Figure 10.44 Phase Difference, Overlap, and Pulse Width in Phase Counting Mode ..... 231
Figure 10.45 Conflict between TCNT Write and Clear Operations ..... 232
Figure 10.46 Conflict between TCNT Write and Increment Operations ..... 233
Figure 10.47 Conflict between TGR Write and Compare Match ..... 234
Figure 10.48 Conflict between Buffer Register Write and Compare Match ..... 235
Figure 10.49 Conflict between TGR Read and Input Capture ..... 236
Figure 10.50 Conflict between TGR Write and Input Capture ..... 237
Figure 10.51 Conflict between Buffer Register Write and Input Capture ..... 238
Figure 10.52 Conflict between Overflow and Counter Clearing ..... 239
Figure 10.53 Conflict between TCNT Write and Overflow ..... 240

**Section 11 8-Bit Timers**

Figure 11.1 Block Diagram of 8-Bit Timer Module ..... 242
Figure 11.2 Example of Pulse Output ..... 252
Figure 11.3 Count Timing for Internal Clock Input ..... 252
Figure 11.4 Count Timing for External Clock Input ..... 253
Figure 11.5 Timing of CMF Setting ..... 253
Figure 11.6 Timing of Timer Output ..... 254
Figure 11.7 Timing of Compare-Match Clear ..... 254
Figure 11.8 Timing of Clearing by External Reset Input ..... 255

Figure 11.9 Timing of OVF Setting ..... 255
Figure 11.10 Conflict between TCNT Write and Clear ..... 258
Figure 11.11 Conflict between TCNT Write and Increment ..... 259
Figure 11.12 Conflict between TCOR Write and Compare-Match ..... 259

**Section 12 Programmable Pulse Generator (PPG)**

Figure 12.1 Block Diagram of PPG ..... 264
Figure 12.2 PPG Output Operation ..... 272
Figure 12.3 Timing of Transfer and Output of NDR Contents (Example) ..... 273
Figure 12.4 Setup Procedure for Normal Pulse Output (Example) ..... 274
Figure 12.5 Normal Pulse Output Example (Five-Phase Pulse Output) ..... 275
Figure 12.6 Non-Overlapping Pulse Output ..... 276
Figure 12.7 Non-Overlapping Operation and NDR Write Timing ..... 277
Figure 12.8 Setup Procedure for Non-Overlapping Pulse Output (Example) ..... 278
Figure 12.9 Non-Overlapping Pulse Output Example (Four-Phase Complementary) ..... 279
Figure 12.10 Inverted Pulse Output (Example) ..... 281
Figure 12.11 Pulse Output Triggered by Input Capture (Example) ..... 282

**Section 13 Watchdog Timer**

Figure 13.1 Block Diagram of WDT ..... 283
Figure 13.2 Example of WDT0 Watchdog Timer Operation ..... 287
Figure 13.3 Writing to TCNT, TCSR, and RSTCSR (example for WDT0) ..... 289
Figure 13.4 Conflict between TCNT Write and Increment ..... 289

**Section 14 Serial Communication Interface (SCI)**

Figure 14.1 Block Diagram of SCI ..... 292
Figure 14.2 Data Format in Asynchronous Communication (Example with 8-Bit Data, Parity, Two Stop Bits) ..... 315
Figure 14.3 Receive Data Sampling Timing in Asynchronous Mode ..... 317
Figure 14.4 Relationship between Output Clock and Transfer Data Phase (Asynchronous Mode) ..... 318
Figure 14.5 Sample SCI Initialization Flowchart ..... 319
Figure 14.6 Example of Operation in Transmission in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit) ..... 320
Figure 14.7 Sample Serial Transmission Flowchart ..... 321
Figure 14.8 Example of SCI Operation in Reception (Example with 8-Bit Data, Parity, One Stop Bit) ..... 322
Figure 14.9 Sample Serial Reception Data Flowchart (1) ..... 324
Figure 14.9 Sample Serial Reception Data Flowchart (2) ..... 325
Figure 14.10 Example of Communication Using Multiprocessor Format (Transmission of Data H′AA to Receiving Station A) ..... 327
Figure 14.11 Sample Multiprocessor Serial Transmission Flowchart ..... 328
Figure 14.12 Example of SCI Operation in Reception

(Example with 8-Bit Data, Multiprocessor Bit, One Stop Bit) ..... 329
Figure 14.13 Sample Multiprocessor Serial Reception Flowchart (1) ..... 330
Figure 14.13 Sample Multiprocessor Serial Reception Flowchart (2) ..... 331
Figure 14.14 Data Format in Synchronous Communication (For LSB-First) ..... 332
Figure 14.15 Sample SCI Initialization Flowchart ..... 333
Figure 14.16 Sample SCI Transmission Operation in Clocked Synchronous Mode ..... 334
Figure 14.17 Sample Serial Transmission Flowchart ..... 335
Figure 14.18 Example of SCI Operation in Reception ..... 336
Figure 14.19 Sample Serial Reception Flowchart ..... 337
Figure 14.20 Sample Flowchart of Simultaneous Serial Transmit and Receive Operations ..... 339
Figure 14.21 Schematic Diagram of Smart Card Interface Pin Connections ..... 340
Figure 14.22 Normal Smart Card Interface Data Format ..... 341
Figure 14.23 Direct Convention (SDIR = SINV = O/$\overline{E}$ = 0) ..... 341
Figure 14.24 Inverse Convention (SDIR = SINV = O/$\overline{E}$ = 1) ..... 342
Figure 14.25 Receive Data Sampling Timing in Smart Card Mode
(Using Clock of 372 Times the Transfer Rate) ..... 343
Figure 14.26 Retransfer Operation in SCI Transmit Mode ..... 345
Figure 14.27 TEND Flag Generation Timing in Transmission Operation ..... 346
Figure 14.28 Example of Transmission Processing Flow ..... 347
Figure 14.29 Retransfer Operation in SCI Receive Mode ..... 348
Figure 14.30 Example of Reception Processing Flow ..... 349
Figure 14.31 Timing for Fixing Clock Output Level ..... 349
Figure 14.32 Clock Halt and Restart Procedure ..... 350

**Section 15 Hitachi Controller Area Network (HCAN)**

Figure 15.1 HCAN Block Diagram ..... 356
Figure 15.2 Message Control Register Configuration ..... 380
Figure 15.3 Standard Format ..... 380
Figure 15.4 Extended Format ..... 380
Figure 15.5 Message Data Configuration ..... 382
Figure 15.6 Hardware Reset Flowchart ..... 385
Figure 15.7 Software Reset Flowchart ..... 386
Figure 15.8 Detailed Description of One Bit ..... 387
Figure 15.9 Transmission Flowchart ..... 390
Figure 15.10 Transmit Message Cancellation Flowchart ..... 392
Figure 15.11 Reception Flowchart ..... 394
Figure 15.12 Unread Message Overwrite Flowchart ..... 396
Figure 15.13 HCAN Sleep Mode Flowchart ..... 398
Figure 15.14 HCAN Halt Mode Flowchart ..... 400
Figure 15.15 DTC Transfer Flowchart ..... 402
Figure 15.16 High-Speed Interface Using PCA82C250 ..... 403

**Section 16 Synchronous Serial Communication Unit (SSU)**

Figure 16.1 Block Diagram of SSU ..... 406
Figure 16.2 Relationship of Clock Phase, Polarity, and Data ..... 416
Figure 16.3 Relationship between Data I/O Pins and the Shift Register ..... 417
Figure 16.4 Example of SSU Initialization ..... 418
Figure 16.5 Example of Transmission Operation ..... 419
Figure 16.6 Example of Data Transmission Flowchart ..... 420
Figure 16.7 Example of Reception Operation ..... 422
Figure 16.8 Example of Data Reception Flowchart ..... 423
Figure 16.9 Example of Simultaneous Transmission/Reception Flowchart ..... 424
Figure 16.10 Arbitration Detection Timing (Before Transfer) ..... 425
Figure 16.11 Arbitration Detection Timing (After Transfer End) ..... 425

**Section 17 A/D Converter**

Figure 17.1 Block Diagram of A/D Converter ..... 428
Figure 17.2 A/D Conversion Timing ..... 435
Figure 17.3 External Trigger Input Timing ..... 437
Figure 17.4 A/D Conversion Accuracy Definitions ..... 439
Figure 17.5 A/D Conversion Accuracy Definitions ..... 439
Figure 17.6 Example of Analog Input Circuit ..... 440
Figure 17.7 Example of Analog Input Protection Circuit ..... 442
Figure 17.8 Analog Input Pin Equivalent Circuit ..... 442

**Section 19 ROM**

Figure 19.1 Block Diagram of Flash Memory ..... 446
Figure 19.2 Flash Memory State Transitions ..... 447
Figure 19.3 Boot Mode ..... 448
Figure 19.4 User Program Mode ..... 449
Figure 19.5 Flash Memory Block Configuration ..... 450
Figure 19.6 Programming/Erasing Flowchart Example in User Program Mode ..... 458
Figure 19.7 Flowchart for Flash Memory Emulation in RAM ..... 459
Figure 19.8 Example of RAM Overlap Operation ..... 460
Figure 19.9 Program/Program-Verify Flowchart ..... 462
Figure 19.10 Erase/Erase-Verify Flowchart ..... 464

**Section 20 Clock Pulse Generator**

Figure 20.1 Block Diagram of Clock Pulse Generator ..... 467
Figure 20.2 Connection of Crystal Resonator (Example) ..... 470
Figure 20.3 Crystal Resonator Equivalent Circuit ..... 470
Figure 20.4 External Clock Input (Examples) ..... 471
Figure 20.5 External Clock Input Timing ..... 472
Figure 20.6 Note on Board Design of Oscillator Circuit ..... 474
Figure 20.7 External Circuitry Recommended for PLL Circuit ..... 475

**Section 21 Power-Down Modes**

Figure 21.1 Mode Transition Diagram ..... 478
Figure 21.2 Medium-Speed Mode Transition and Clearance Timing ..... 484
Figure 21.3 Software Standby Mode Application Example ..... 487
Figure 21.4 Timing of Transition to Hardware Standby Mode ..... 488
Figure 21.5 Timing of Recovery from Hardware Standby Mode ..... 489

**Section 23 Electrical Characteristics**

Figure 23.1 Output Load Circuit ..... 548
Figure 23.2 System Clock Timing ..... 549
Figure 23.3 Oscillation Settling Timing ..... 550
Figure 23.4 Reset Input Timing ..... 551
Figure 23.5 Interrupt Input Timing ..... 551
Figure 23.6 I/O Port Input/Output Timing ..... 555
Figure 23.7 Realtime Input Port Data Input Timing ..... 555
Figure 23.8 TPU Input/Output Timing ..... 555
Figure 23.9 TPU Clock Input Timing ..... 556
Figure 23.10 SCK Clock Input Timing ..... 556
Figure 23.11 SCI Input/Output Timing (Clocked Synchronous Mode) ..... 556
Figure 23.12 A/D Converter External Trigger Input Timing ..... 556
Figure 23.13 HCAN Input/Output Timing ..... 557
Figure 23.14 PPG Output Timing ..... 557
Figure 23.15 SSU Timing (Master, CPHS = 1) ..... 557
Figure 23.16 SSU Timing (Master, CPHS = 0) ..... 558
Figure 23.17 SSU Timing (Slave, CPHS = 1) ..... 558
Figure 23.18 SSU Timing (Slave, CPHS = 0) ..... 559

**Appendix**

Figure C.1 FP-100M Package Dimensions ..... 564

# Tables

**Section 2 CPU**
Table 2.1 Instruction Classification ..... 25
Table 2.2 Operation Notation ..... 26
Table 2.3 Data Transfer Instructions ..... 27
Table 2.4 Arithmetic Operations Instructions (1) ..... 28
Table 2.4 Arithmetic Operations Instructions (2) ..... 29
Table 2.5 Logic Operations Instructions ..... 30
Table 2.6 Shift Instructions ..... 31
Table 2.7 Bit Manipulation Instructions (1) ..... 32
Table 2.7 Bit Manipulation Instructions (2) ..... 33
Table 2.8 Branch Instructions ..... 34
Table 2.9 System Control Instructions ..... 35
Table 2.10 Block Data Transfer Instructions ..... 36
Table 2.11 Addressing Modes ..... 38
Table 2.12 Absolute Address Access Ranges ..... 39

**Section 3 MCU Operating Modes**
Table 3.1 MCU Operating Mode Selection ..... 47

**Section 4 Exception Handling**
Table 4.1 Exception Types and Priority ..... 53
Table 4.2 Exception Handling Vector Table ..... 54
Table 4.3 Statuses of CCR and EXR after Trace Exception Handling ..... 58
Table 4.4 Statuses of CCR and EXR after Trap Instruction Exception Handling ..... 59

**Section 5 Interrupt Controller**
Table 5.1 Pin Configuration ..... 65
Table 5.2 Interrupt Sources, Vector Addresses, and Interrupt Priorities ..... 73
Table 5.3 Interrupt Control Modes ..... 76
Table 5.4 Interrupt Response Times ..... 81
Table 5.5 Number of States in Interrupt Handling Routine Execution Status ..... 82

**Section 8 Data Transfer Controller (DTC)**
Table 8.1 Interrupt Sources, DTC Vector Addresses, and Corresponding DTCEs ..... 106
Table 8.2 Register Information in Normal Mode ..... 109
Table 8.3 Register Information in Repeat Mode ..... 110
Table 8.4 Register Information in Block Transfer Mode ..... 111
Table 8.5 DTC Execution Status ..... 115
Table 8.6 Number of States Required for Each Execution Status ..... 115

**Section 9 I/O Ports**


Table 9.1 Port Functions ..... 122
Table 9.2 P17 Pin Function ..... 127
Table 9.3 P16 Pin Function ..... 127
Table 9.4 P15 Pin Function ..... 127
Table 9.5 P14 Pin Function ..... 128
Table 9.6 P13 Pin Function ..... 128
Table 9.7 P12 Pin Function ..... 128
Table 9.8 P11 Pin Function ..... 129
Table 9.9 P10 Pin Function ..... 129
Table 9.10 P37 Pin Function ..... 131
Table 9.11 P36 Pin Function ..... 132
Table 9.12 P35 Pin Function ..... 132
Table 9.13 P34 Pin Function ..... 132
Table 9.14 P33 Pin Function ..... 132
Table 9.15 P32 Pin Function ..... 132
Table 9.16 P31 Pin Function ..... 132
Table 9.17 P30 Pin Function ..... 133
Table 9.18 P77 Pin Function ..... 135
Table 9.19 P76 Pin Function ..... 135
Table 9.20 P75 Pin Function ..... 135
Table 9.21 P74 Pin Function ..... 135
Table 9.22 P73 Pin Function ..... 136
Table 9.23 P72 Pin Function ..... 136
Table 9.24 P71 Pin Function ..... 136
Table 9.25 P70 Pin Function ..... 136
Table 9.26 PA3 Pin Function ..... 141
Table 9.27 PA2 Pin Function ..... 141
Table 9.28 PA1 Pin Function ..... 141
Table 9.29 PA0 Pin Function ..... 141
Table 9.30 PB7 Pin Function ..... 145
Table 9.31 PB6 Pin Function ..... 145
Table 9.32 PB5 Pin Function ..... 145
Table 9.33 PB4 Pin Function ..... 145
Table 9.34 PB3 Pin Function ..... 146
Table 9.35 PB2 Pin Function ..... 146
Table 9.36 PB1 Pin Function ..... 146
Table 9.37 PB0 Pin Function ..... 146
Table 9.38 PC7 Pin Function ..... 149
Table 9.39 PC6 Pin Function ..... 149
Table 9.40 PC5 Pin Function ..... 150
Table 9.41 PC4 Pin Function ..... 150
Table 9.42 PC3 Pin Function ..... 150

Table 9.43 PC2 Pin Function ..... 150
Table 9.44 PC1 Pin Function ..... 151
Table 9.45 PC 0Pin Function ..... 151
Table 9.46 PF7 Pin Function ..... 157
Table 9.47 PF6 Pin Function ..... 157
Table 9.48 PF5 Pin Function ..... 157
Table 9.49 PF4 Pin Function ..... 157
Table 9.50 PF3 Pin Function ..... 157
Table 9.51 PF2 Pin Function ..... 157
Table 9.52 PF1 Pin Function ..... 158
Table 9.53 PF0 Pin Function ..... 158

**Section 10 16-Bit Timer Pulse Unit (TPU)**

Table 10.1 TPU Functions ..... 160
Table 10.2 TPU Pins ..... 163
Table 10.3 CCLR0 to CCLR2 (Channels 0 and 3) ..... 167
Table 10.4 CCLR0 to CCLR2 (Channels 1, 2, 4, and 5) ..... 167
Table 10.5 TPSC0 to TPSC2 (Channel 0) ..... 168
Table 10.6 TPSC0 to TPSC2 (Channel 1) ..... 168
Table 10.7 TPSC0 to TPSC2 (Channels 2) ..... 169
Table 10.8 TPSC0 to TPSC2 (Channel 3) ..... 169
Table 10.9 TPSC0 to TPSC2 (Channel 4) ..... 170
Table 10.10 TPSC0 to TPSC2 (Channel 5) ..... 170
Table 10.11 MD0 to MD3 ..... 172
Table 10.12 TIORH_0 (Channel 0) ..... 174
Table 10.13 TIORL_0 (Channel 0) ..... 175
Table 10.14 TIOR_1 (Channel 1) ..... 176
Table 10.15 TIOR_2 (Channel 2) ..... 177
Table 10.16 TIORH_3 (Channel 3) ..... 178
Table 10.17 TIORL_3 (Channel 3) ..... 179
Table 10.18 TIOR_4 (Channel 4) ..... 180
Table 10.19 TIOR_5 (Channel 5) ..... 181
Table 10.20 TIORH_0 (Channel 0) ..... 182
Table 10.21 TIORL_0 (Channel 0) ..... 183
Table 10.22 TIOR_1 (Channel 1) ..... 184
Table 10.23 TIOR_2 (Channel 2) ..... 185
Table 10.24 TIORH_3 (Channel 3) ..... 186
Table 10.25 TIORL_3 (Channel 3) ..... 187
Table 10.26 TIOR_4 (Channel 4) ..... 188
Table 10.27 TIOR_5 (Channel 5) ..... 189
Table 10.28 Register Combinations in Buffer Operation ..... 205
Table 10.29 Cascaded Combinations ..... 208
Table 10.30 PWM Output Registers and Output Pins ..... 210

Table 10.30 PWM Output Registers and Output Pins (cont) ..... 211
Table 10.31 Phase Counting Mode Clock Input Pins ..... 214
Table 10.32 Up/Down-Count Conditions in Phase Counting Mode 1 ..... 215
Table 10.33 Up/Down-Count Conditions in Phase Counting Mode 2 ..... 216
Table 10.34 Up/Down-Count Conditions in Phase Counting Mode 3 ..... 217
Table 10.35 Up/Down-Count Conditions in Phase Counting Mode 4 ..... 218
Table 10.36 TPU Interrupts ..... 222

**Section 11 8-Bit Timers**
Table 11.1 Pin Configuration ..... 243
Table 11.2 8-Bit Timer Interrupt Sources ..... 257
Table 11.3 Timer Output Priorities ..... 260
Table 11.4 Switching of Internal Clock and TCNT Operation ..... 261

**Section 12 Programmable Pulse Generator (PPG)**
Table 12.1 Pin Configuration ..... 265

**Section 13 Watchdog Timer**
Table 13.1 WDT Interrupt Source ..... 288

**Section 14 Serial Communication Interface (SCI)**
Table 14.1 Pin Configuration ..... 293
Table 14.2 The Relationships between The N Setting in BRR and Bit Rate B ..... 308
Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (1) ..... 309
Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (2) ..... 310
Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (3) ..... 311
Table 14.4 Maximum Bit Rate for Each Frequency (Asynchronous Mode) ..... 311
Table 14.5 Maximum Bit Rate with External Clock Input (Asynchronous Mode) ..... 312
Table 14.6 BRR Settings for Various Bit Rates (Clocked Synchronous Mode) ..... 313
Table 14.7 Maximum Bit Rate with External Clock Input (Clocked Synchronous Mode) ..... 313
Table 14.8 Examples of Bit Rate for Various BRR Settings (Smart Card Interface Mode) (When n = 0 and S = 372) ..... 314
Table 14.9 Maximum Bit Rate at Various Frequencies (Smart Card Interface Mode) (when S = 372) ..... 314
Table 14.10 Serial Transfer Formats (Asynchronous Mode) ..... 316
Table 14.11 SSR Status Flags and Receive Data Handling ..... 323
Table 14.12 SCI Interrupt Sources ..... 351
Table 14.13 SCI Interrupt Sources ..... 352

**Section 15 Hitachi Controller Area Network (HCAN)**
Table 15.1 HCAN Pins ..... 357
Table 15.2 Limits for the Settable Value ..... 387
Table 15.3 Setting Range for TSEG1 and TSEG2 in BCR ..... 388

Table 15.4 HCAN Interrupt Sources ..... 400

**Section 16 Synchronous Serial Communication Unit (SSU)**
Table 16.1 Pin Configuration ..... 407
Table 16.2 Interrupt Souses ..... 426

**Section 17 A/D Converter**
Table 17.1 Pin Configuration ..... 429
Table 17.2 Analog Input Channels and Corresponding ADDR Registers ..... 430
Table 17.3 A/D Conversion Time (Single Mode) ..... 436
Table 17.4 A/D Conversion Time (Scan Mode) ..... 436
Table 17.5 A/D Converter Interrupt Source ..... 437
Table 17.6 Analog Pin Specifications ..... 442

**Section 19 ROM**
Table 19.1 Differences between Boot Mode and User Program Mode ..... 447
Table 19.2 Pin Configuration ..... 451
Table 19.3 Setting On-Board Programming Modes ..... 455
Table 19.4 Boot Mode Operation ..... 457
Table 19.5 System Clock Frequencies for which Automatic Adjustment of LSI Bit Rate is Possible ..... 457
Table 19.6 Flash Memory Operating States ..... 466

**Section 20 Clock Pulse Generator**
Table 20.1 Damping Resistance Value ..... 470
Table 20.2 Crystal Resonator Characteristics ..... 470
Table 20.3 External Clock Input Conditions ..... 472

**Section 21 Power-Down Modes**
Table 21.1 Low Power Consumption Mode Transition Conditions ..... 478
Table 21.2 LSI Internal States in Each Mode ..... 479
Table 21.3 Oscillation Stabilization Time Settings ..... 486
Table 21.4 $\phi$ Pin State in Each Processing State ..... 490

**Section 23 Electrical Characteristics**
Table 23.1 Absolute Maximum Ratings ..... 545
Table 23.2 DC Characteristics ..... 546
Table 23.3 Permissible Output Currents ..... 548
Table 23.4 Clock Timing ..... 549
Table 23.5 Control Signal Timing ..... 550
Table 23.6 Timing of On-Chip Peripheral Modules ..... 552
Table 23.7 Timing of SSU ..... 553
Table 23.8 A/D Conversion Characteristics ..... 560

Table 23.9 Flash Memory Characteristics ..... 561

# Section 1 Overview

## 1.1 Overview

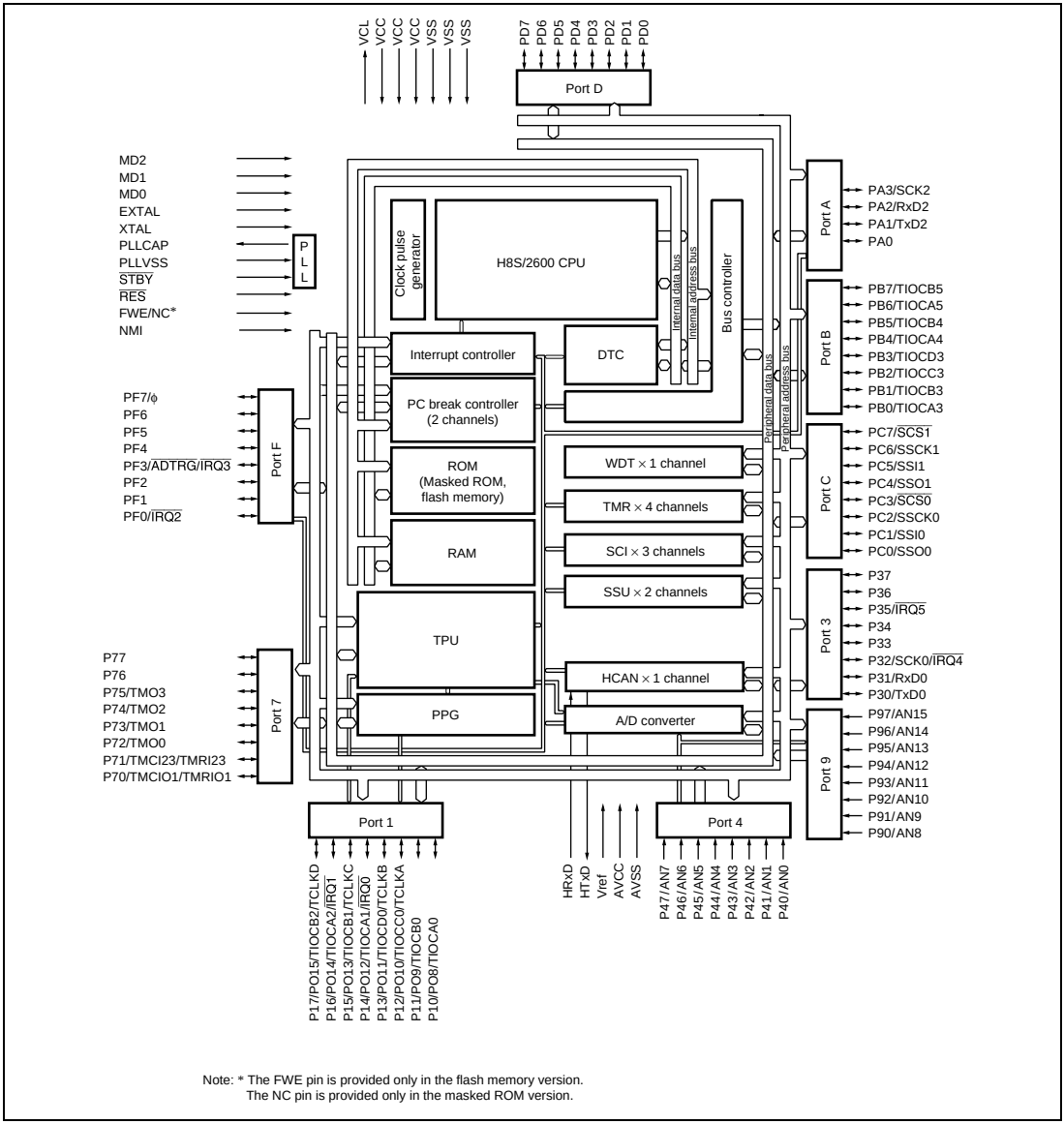
- High-speed H8S/2600 central processing unit with an internal 16-bit architecture
  - Upward-compatible with H8/300 and H8/300H CPUs on an object level
  - Sixteen 16-bit general registers
  - 69 basic instructions
- Various peripheral functions
  - PC break controller
  - Data transfer controller
  - 16-bit timer-pulse unit (TPU)
  - 8-bit timer (TMR)
  - Programmable pulse generator (PPG)
  - Watchdog timer
  - Asynchronous or clocked synchronous serial communication interface (SCI)
  - Hitachi controller area network (HCAN)
  - Synchronous serial communication unit (SSU)
  - 10-bit A/D converter
  - Clock pulse generator
- On-chip memory

| ROM | Model | ROM | RAM | Remarks |
|---|---|---|---|---|
| F-ZTAT Version | HD64F2628 | 128 kbytes | 8 kbytes | |
| Masked ROM Version | HD6432628 | 128 kbytes | 8 kbytes | Under development |
| | HD6432627 | 96 kbytes | 6 kbytes | Under development |

- General I/O ports
  - I/O pins: 59
  - Input-only pins: 17
- Supports various power-down states
- Compact package

| Package | (Code) | Body Size | Pin Pitch |
|---|---|---|---|
| QFP-100 | FP-100M | 14.0 × 14.0 mm | 0.5 mm |

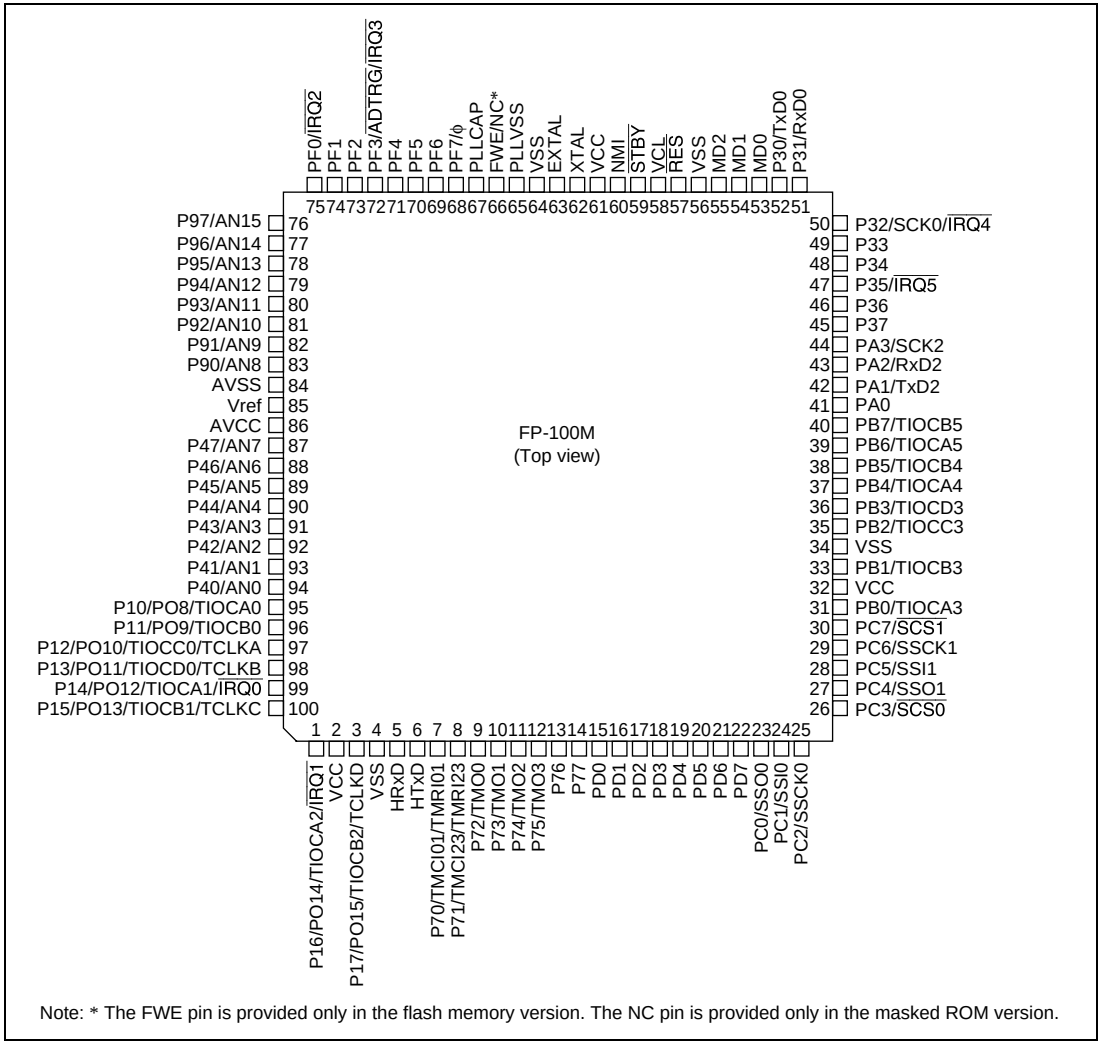
## 1.2 Internal Block Diagram




Note: * The FWE pin is provided only in the flash memory version.
The NC pin is provided only in the masked ROM version.

**Figure 1.1 Internal Block Diagram**

## 1.3 Pin Arrangement




Note: * The FWE pin is provided only in the flash memory version. The NC pin is provided only in the masked ROM version.

**Figure 1.2 Pin Arrangement**

## 1.4 Pin Functions

| Type | Symbol | Pin NO. | I/O | Function |
|---|---|---|---|---|
| Power Supply | VCC | 2<br>32<br>61 | Input | Power supply pins. Connect all these pins to the system power supply. |
| | VSS | 4<br>34<br>56<br>64 | Input | Ground pins. Connect all these pins to the system power supply (0 V). |
| | VCL | 58 | Output | External capacitance pin for internal power-down power supply. Connect this pin to VSS via a 0.1-µF capacitor (placed close to the pins). |
| Clock | PLLVSS | 65 | Input | On-chip PLL oscillator ground pin. |
| | PLLCAP | 67 | Output | External capacitance pin for an on-chip PLL oscillator. |
| | XTAL | 62 | Input | For connection to a crystal resonator. For examples of crystal resonator connection and external clock input, see section 20, Clock Pulse Generator. |
| | EXTAL | 63 | Input | For connection to a crystal resonator. (An external clock can be supplied from the EXTAL pin.) For examples of crystal resonator connection and external clock input, see section 20, Clock Pulse Generator. |
| | $\phi$ | 68 | Output | Supplies the system clock to external devices. |
| Operating mode control | MD2<br>MD1<br>MD0 | 55<br>54<br>53 | Input | Set the operating mode. Inputs at these pins should not be changed during operation. |
| System control | $\overline{\text{RES}}$ | 57 | Input | Reset input pin. When this pin is low, the chip is reset. |
| | $\overline{\text{STBY}}$ | 59 | Input | When this pin is low, a transition is made to hardware standby mode. |
| | FWE | 66 | Input | Pin for use by flash memory. This pin is only used in the flash memory version. |

| Type | Symbol | Pin NO. | I/O | Function |
|---|---|---|---|---|
| Interrupts | NMI | 60 | Input | Nonmaskable interrupt pin. If this pin is not used, it should be fixed high. |
| | $\overline{IRQ5}$<br>$\overline{IRQ4}$<br>$\overline{IRQ3}$<br>$\overline{IRQ2}$<br>$\overline{IRQ1}$<br>$\overline{IRQ0}$ | 47<br>50<br>72<br>75<br>1<br>99 | Input | These pins request a maskable interrupt. |
| 16-bit timer-pulse unit | TCLKA<br>TCLKB<br>TCLKC<br>TCLKD | 97<br>98<br>100<br>3 | Input | These pins input an external clock. |
| | TIOCA0<br>TIOCB0<br>TIOCC0<br>TIOCD0 | 95<br>96<br>97<br>98 | Input/ Output | TGRA_0 to TGRD_0 input capture input/output compare output/PWM output pins. |
| | TIOCA1<br>TIOCB1 | 99<br>100 | Input/ Output | TGRA_1 to TGRB_1 input capture input/output compare output/PWM output pins. |
| | TIOCA2<br>TIOCB2 | 1<br>3 | Input/ Output | TGRA_2 to TGRB_2 input capture input/output compare output/PWM output pins. |
| | TIOCA3<br>TIOCB3<br>TIOCC3<br>TIOCD3 | 31<br>33<br>35<br>36 | Input/ Output | TGRA_3 to TGRD_3 input capture input/output compare output/PWM output pins. |
| | TIOCA4<br>TIOCB4 | 37<br>38 | Input/ Output | TGRA_4 to TGRB_4 input capture input/output compare output/PWM output pins. |
| | TIOCA5<br>TIOCB5 | 39<br>40 | Input/ Output | TGRA_5 to TGRB_5 input capture input/output compare output/PWM output pins. |
| Program-mable pulse generator (PPG) | PO15<br>PO14<br>PO13<br>PO12<br>PO11<br>PO10<br>PO9<br>PO8 | 3<br>1<br>100<br>99<br>98<br>97<br>96<br>95 | Output | Pulse output pins. |
| 8-bit timer (TMR) | TMO3<br>TMO2<br>TMO1<br>TMO0 | 12<br>11<br>10<br>9 | Output | Compare-match output pins |
| | TMCI23<br>TMCI01 | 8<br>7 | Input | Input pins of external clocks input to the counter |

| Type | Symbol | Pin NO. | I/O | Function |
|---|---|---|---|---|
| 8-bit timer (TMR) | TMRI23<br>TMRI01 | 8<br>7 | Input | Counter reset input pins |
| Serial communi-cation Interface (SCI)/ smart card interface | TxD2<br>TxD0 | 42<br>52 | Output | Data output pins |
| | RxD2<br>RxD0 | 43<br>51 | Input | Data input pins |
| | SCK2<br>SCK0 | 44<br>50 | Input/ Output | Clock input/output pins |
| HCAN | HTxD | 6 | Output | CAN bus transmission pin |
| | HRxD | 5 | Input | CAN bus reception pin |
| Synchro-nous serial communi-cation unit (SSU) | SSO1<br>SSO0 | 27<br>23 | Input/ Output | Data input/output pins |
| | SSI1<br>SSI0 | 28<br>24 | Input/ Output | Data input/output pins |
| | SSCK1<br>SSCK0 | 29<br>25 | Input/ Output | Clock input/output pins |
| | $\overline{\text{SCS1}}$<br>$\overline{\text{SCS0}}$ | 30<br>26 | Input/ Output | Chip select input/output pins |
| A/D converter | AN15<br>AN14<br>AN13<br>AN12<br>AN11<br>AN10<br>AN9<br>AN8<br>AN7<br>AN6<br>AN5<br>AN4<br>AN3<br>AN2<br>AN1<br>AN0 | 76<br>77<br>78<br>79<br>80<br>81<br>82<br>83<br>87<br>88<br>89<br>90<br>91<br>92<br>93<br>94 | Input | Analog input pins |
| | $\overline{\text{ADTRG}}$ | 72 | Input | Pin for input of an external trigger to start A/D conversion |
| | AVCC | 86 | Input | Power supply pin for the A/D converter. When the A/D converter is not used, connect this pin to the system power supply (+5 V). |
| | AVSS | 84 | Input | The ground pin for the A/D converter. Connect this pin to the system power supply (0 V). |

| Type | Symbol | Pin NO. | I/O | Function |
|---|---|---|---|---|
| A/D converter | Vref | 85 | Input | The reference voltage input pin for the A/D converter. When the A/D converter is not used, connect this pin to the system power supply (+5 V). |
| I/O ports | P17<br>P16<br>P15<br>P14<br>P13<br>P12<br>P11<br>P10 | 3<br>1<br>100<br>99<br>98<br>97<br>96<br>95 | Input/ Output | Eight input/output pins |
| | P37<br>P36<br>P35<br>P34<br>P33<br>P32<br>P31<br>P30 | 45<br>46<br>47<br>48<br>49<br>50<br>51<br>52 | Input/ Output | Eight input/output pins |
| | P47<br>P46<br>P45<br>P44<br>P43<br>P42<br>P41<br>P40 | 87<br>88<br>89<br>90<br>91<br>92<br>93<br>94 | Input | Eight input pins |
| | P77<br>P76<br>P75<br>P74<br>P73<br>P72<br>P71<br>P70 | 14<br>13<br>12<br>11<br>10<br>9<br>8<br>7 | Input/ Output | Eight input/output pins |
| | P97<br>P96<br>P95<br>P94<br>P93<br>P92<br>P91<br>P90 | 76<br>77<br>78<br>79<br>80<br>81<br>82<br>83 | Input | Eight input pins |

| Type | Symbol | Pin NO. | I/O | Function |
|---|---|---|---|---|
| I/O ports | PA3 | 44 | Input/ | Four input/output pins |
| | PA2 | 43 | Output | |
| | PA1 | 42 | | |
| | PA0 | 41 | | |
| | PB7 | 40 | Input/ | Eight input/output pins |
| | PB6 | 39 | Output | |
| | PB5 | 38 | | |
| | PB4 | 37 | | |
| | PB3 | 36 | | |
| | PB2 | 35 | | |
| | PB1 | 33 | | |
| | PB0 | 31 | | |
| | PC7 | 30 | Input/ | Eight input/output pins |
| | PC6 | 29 | Output | |
| | PC5 | 28 | | |
| | PC4 | 27 | | |
| | PC3 | 26 | | |
| | PC2 | 25 | | |
| | PC1 | 24 | | |
| | PC0 | 23 | | |
| | PD7 | 22 | Input/ | Eight input/output pins |
| | PD6 | 21 | Output | |
| | PD5 | 20 | | |
| | PD4 | 19 | | |
| | PD3 | 18 | | |
| | PD2 | 17 | | |
| | PD1 | 16 | | |
| | PD0 | 15 | | |
| | PF7 | 68 | Input/ | Eight input/output pins |
| | PF6 | 69 | Output | |
| | PF5 | 70 | | |
| | PF4 | 71 | | |
| | PF3 | 72 | | |
| | PF2 | 73 | | |
| | PF1 | 74 | | |
| | PF0 | 75 | | |

# Section 2 CPU

The H8S/2600 CPU is a high-speed central processing unit with an internal 32-bit architecture that is upward-compatible with the H8/300 and H8/300H CPUs. The H8S/2600 CPU has sixteen 16-bit general registers, can address a 16-Mbyte linear address space, and is ideal for realtime control. This section describes the H8S/2600 CPU. The usable modes and address spaces differ depending on the product. For details on each product, refer to section 3, MCU Operating Modes.

## 2.1 Features

- Upward-compatible with H8/300 and H8/300H CPUs
  - Can execute H8/300 and H8/300H CPUs object programs
- General-register architecture
  - Sixteen 16-bit general registers also usable as sixteen 8-bit registers or eight 32-bit registers
- Sixty-nine basic instructions
  - 8/16/32-bit arithmetic and logic instructions
  - Multiply and divide instructions
  - Powerful bit-manipulation instructions
  - Multiply-and-accumulate instruction
- Eight addressing modes
  - Register direct [Rn]
  - Register indirect [@ERn]
  - Register indirect with displacement [@(d:16,ERn) or @(d:32,ERn)]
  - Register indirect with post-increment or pre-decrement [@ERn+ or @–ERn]
  - Absolute address [@aa:8, @aa:16, @aa:24, or @aa:32]
  - Immediate [#xx:8, #xx:16, or #xx:32]
  - Program-counter relative [@(d:8,PC) or @(d:16,PC)]
  - Memory indirect [@@aa:8]
- 16-Mbyte address space
  - Program: 16 Mbytes
  - Data: 16 Mbytes
- High-speed operation
  - All frequently-used instructions execute in one or two states
  - 8/16/32-bit register-register add/subtract : 1 state
  - 8 × 8-bit register-register multiply : 3 states
  - 16 ÷ 8-bit register-register divide : 12 states
  - 16 × 16-bit register-register multiply : 4 states
  - 32 ÷ 16-bit register-register divide : 20 states

- Two CPU operating modes
  - Normal mode*
  - Advanced mode
- Power-down state
  - Transition to power-down state by the SLEEP instruction
  - CPU clock speed selection

Note:* Normal mode is not available in this LSI.

### 2.1.1 Differences between H8S/2600 CPU and H8S/2000 CPU

The differences between the H8S/2600 CPU and the H8S/2000 CPU are shown below.

- Register configuration

  The MAC register is supported by the H8S/2600 CPU only.
- Basic instructions

  The four instructions MAC, CLRMAC, LDMAC, and STMAC are supported by the H8S/2600 CPU only.
- The number of execution states of the MULXU and MULXS instructions;

| | | Execution States | |
|---|---|---|---|
| **Instruction** | **Mnemonic** | **H8S/2600** | **H8S/2000** |
| MULXU | MULXU.B Rs, Rd | 3 | 12 |
| | MULXU.W Rs, ERd | 4 | 20 |
| MULXS | MULXS.B Rs, Rd | 4 | 13 |
| | MULXS.W Rs, ERd | 5 | 21 |

In addition, there are differences in address space, CCR and EXR register functions, and power-down modes, etc., depending on the model.

### 2.1.2 Differences from H8/300 CPU

In comparison to the H8/300 CPU, the H8S/2600 CPU has the following enhancements:

- More general registers and control registers
  - Eight 16-bit extended registers, and one 8-bit and two 32-bit control registers, have been added.
- Expanded address space
  - Normal mode supports the same 64-kbyte address space as the H8/300 CPU.
  - Advanced mode supports a maximum 16-Mbyte address space.
- Enhanced addressing
  - The addressing modes have been enhanced to make effective use of the 16-Mbyte address space.
- Enhanced instructions
  - Addressing modes of bit-manipulation instructions have been enhanced.
  - Signed multiply and divide instructions have been added.
  - A multiply-and-accumulate instruction has been added.
  - Two-bit shift instructions have been added.
  - Instructions for saving and restoring multiple registers have been added.
  - A test and set instruction has been added.
- Higher speed
  - Basic instructions execute twice as fast.

### 2.1.3 Differences from H8/300H CPU

In comparison to the H8/300H CPU, the H8S/2600 CPU has the following enhancements:

- More control registers
  - One 8-bit and two 32-bit control registers have been added.
- Enhanced instructions
  - Addressing modes of bit-manipulation instructions have been enhanced.
  - A multiply-and-accumulate instruction has been added.
  - Two-bit shift instructions have been added.
  - Instructions for saving and restoring multiple registers have been added.
  - A test and set instruction has been added.
- Higher speed
  - Basic instructions execute twice as fast.

## 2.2 CPU Operating Modes

The H8S/2600 CPU has two operating modes: normal and advanced. Normal mode supports a maximum 64-kbyte address space. Advanced mode supports a maximum 16-Mbyte total address space. The mode is selected by the mode pins.

### 2.2.1 Normal Mode

The exception vector table and stack have the same structure as in the H8/300 CPU.

- Address Space

  Linear access to a 64-kbyte maximum address space is provided.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers, or as the upper 16-bit segments of 32-bit registers. When En is used as a 16-bit register it can contain any value, even when the corresponding general register (Rn) is used as an address register. If the general register is referenced in the register indirect addressing mode with pre-decrement (@–Rn) or post-increment (@Rn+) and a carry or borrow occurs, however, the value in the corresponding extended register (En) will be affected.
- Instruction Set

  All instructions and addressing modes can be used. Only the lower 16 bits of effective addresses (EA) are valid.
- Exception Vector Table and Memory Indirect Branch Addresses

  In normal mode the top area starting at H′0000 is allocated to the exception vector table. One branch address is stored per 16 bits. The exception vector table structure in normal mode is shown in figure 2.1. For details of the exception vector table, see section 4, Exception Handling.

  The memory indirect addressing mode (@@aa:8) employed in the JMP and JSR instructions uses an 8-bit absolute address included in the instruction code to specify a memory operand that contains a branch address. In normal mode the operand is a 16-bit word operand, providing a 16-bit branch address. Branch addresses can be stored in the area from H′0000 to H′00FF. Note that the first part of this range is also used for the exception vector table.
- Stack Structure

  When the program counter (PC) is pushed onto the stack in a subroutine call, and the PC, condition-code register (CCR), and extended control register (EXR) are pushed onto the stack in exception handling, they are stored as shown in figure 2.2. EXR is not pushed onto the stack in interrupt control mode 0. For details, see section 4, Exception Handling.

Note: Normal mode is not available in this LSI.

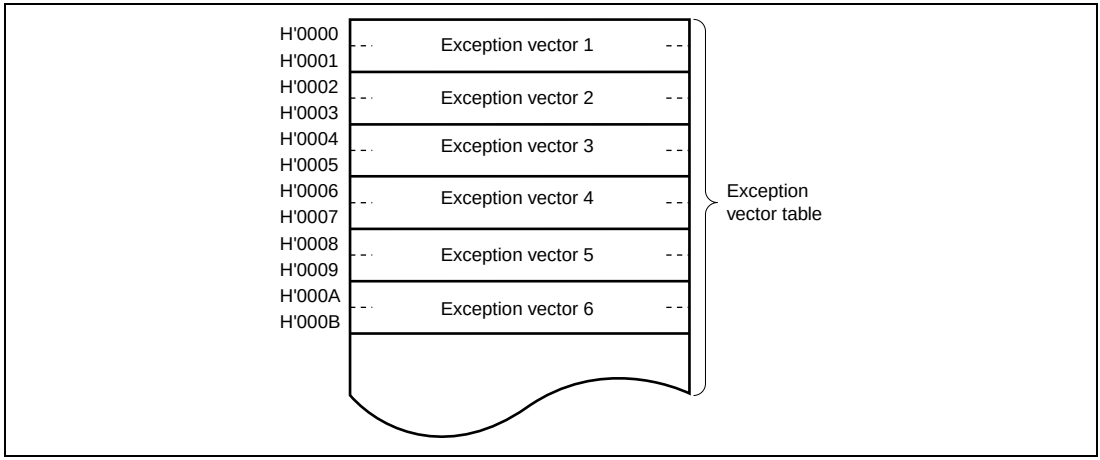



**Figure 2.1 Exception Vector Table (Normal Mode)**

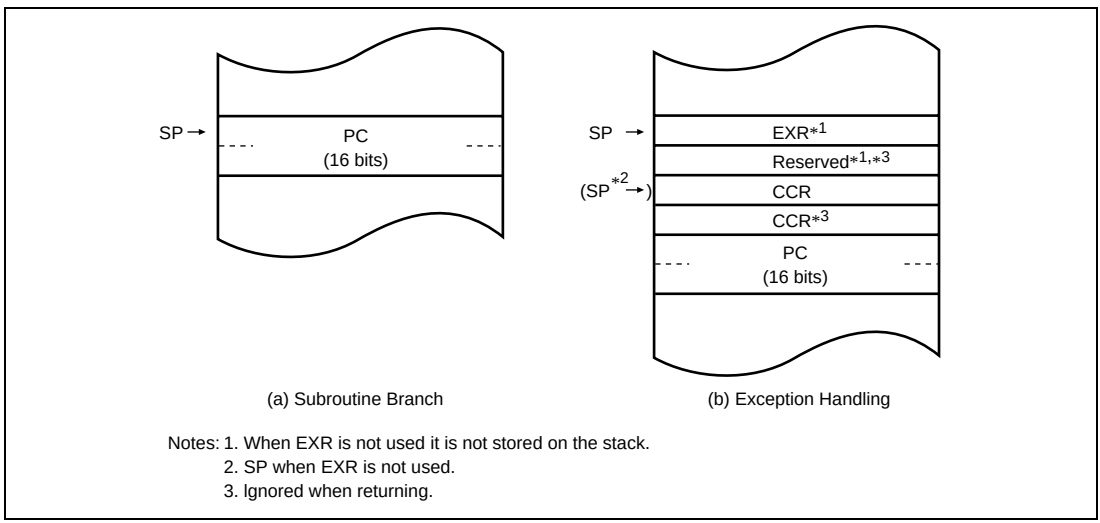



**Figure 2.2 Stack Structure in Normal Mode**

### 2.2.2 Advanced Mode

- Address Space

  Linear access to a 16-Mbyte maximum address space is provided.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers, or as the upper 16-bit segments of 32-bit registers or address registers.
- Instruction Set

  All instructions and addressing modes can be used.

- Exception Vector Table and Memory Indirect Branch Addresses

  In advanced mode, the top area starting at H′00000000 is allocated to the exception vector table in units of 32 bits. In each 32 bits, the upper 8 bits are ignored and a branch address is stored in the lower 24 bits (figure 2.3). For details of the exception vector table, see section 4, Exception Handling.

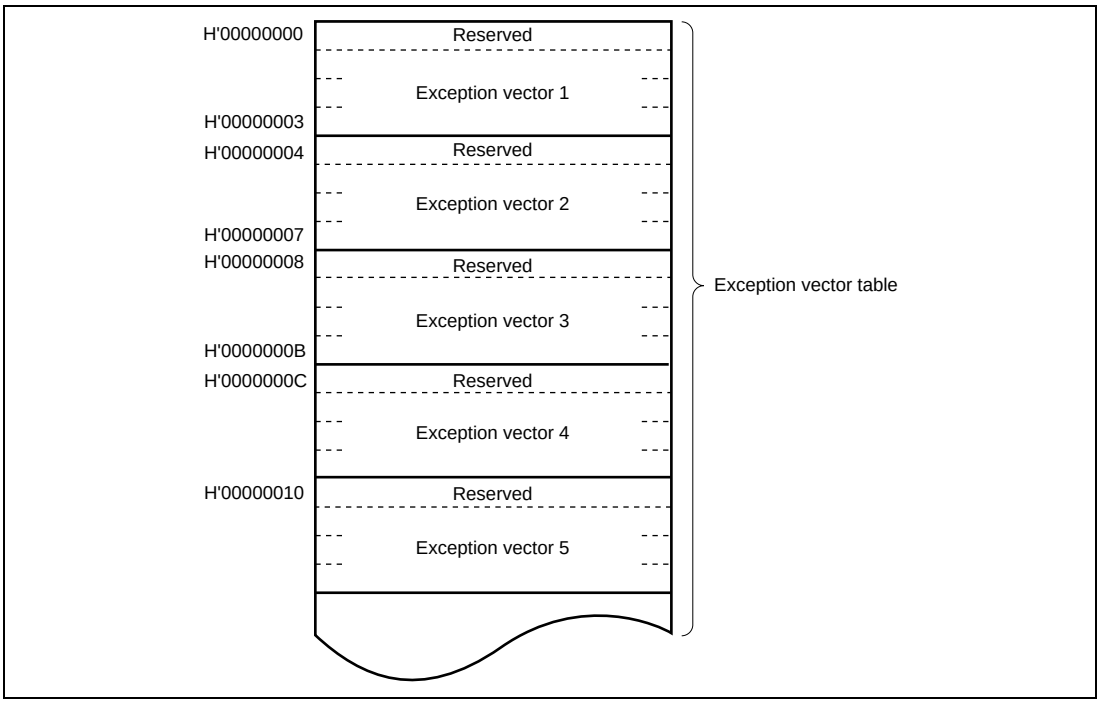
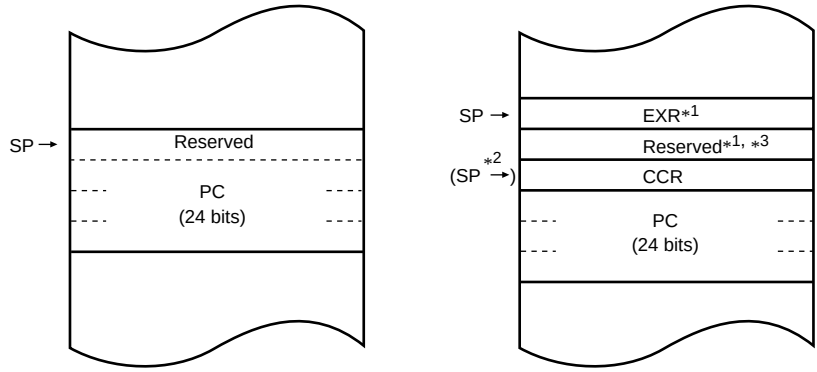



**Figure 2.3 Exception Vector Table (Advanced Mode)**

The memory indirect addressing mode (@@aa:8) employed in the JMP and JSR instructions uses an 8-bit absolute address included in the instruction code to specify a memory operand that contains a branch address. In advanced mode the operand is a 32-bit longword operand, providing a 32-bit branch address. The upper 8 bits of these 32 bits is a reserved area that is regarded as H′00. Branch addresses can be stored in the area from H′00000000 to H′000000FF. Note that the first part of this range is also used for the exception vector table.

- Stack Structure

  In advanced mode, when the program counter (PC) is pushed onto the stack in a subroutine call, and the PC, condition-code register (CCR), and extended control register (EXR) are pushed onto the stack in exception handling, they are stored as shown in figure 2.4. When EXR is not pushed onto the stack in interrupt control mode 0. For details, see section 4, Exception Handling.




(a) Subroutine Branch

(b) Exception Handling

Notes: 1. When EXR is not used it is not stored on the stack.
2. SP when EXR is not used.
3. Ignored when returning.

**Figure 2.4 Stack Structure in Advanced Mode**

## 2.3 Address Space

Figure 2.5 shows a memory map for the H8S/2600 CPU. The H8S/2600 CPU provides linear access to a maximum 64-kbyte address space in normal mode, and a maximum 16-Mbyte (architecturally 4-Gbyte) address space in advanced mode. The usable modes and address spaces differ depending on the product. For details on each product, refer to section 3, MCU Operating Modes.

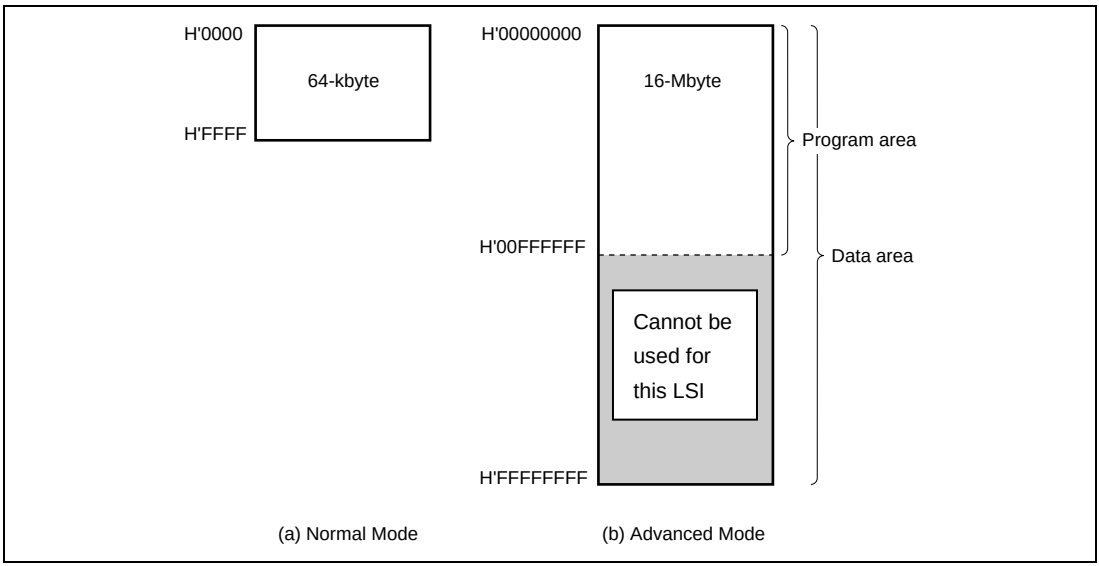



**Figure 2.5 Memory Map**

## 2.4 Registers

The H8S/2600 CPU has the internal registers shown in figure 2.6. There are two types of registers; general registers and control registers. The control registers are a 24-bit program counter (PC), an 8-bit extended control register (EXR), an 8-bit condition code register (CCR), and a 64-bit multiply-accumulate register (MAC).

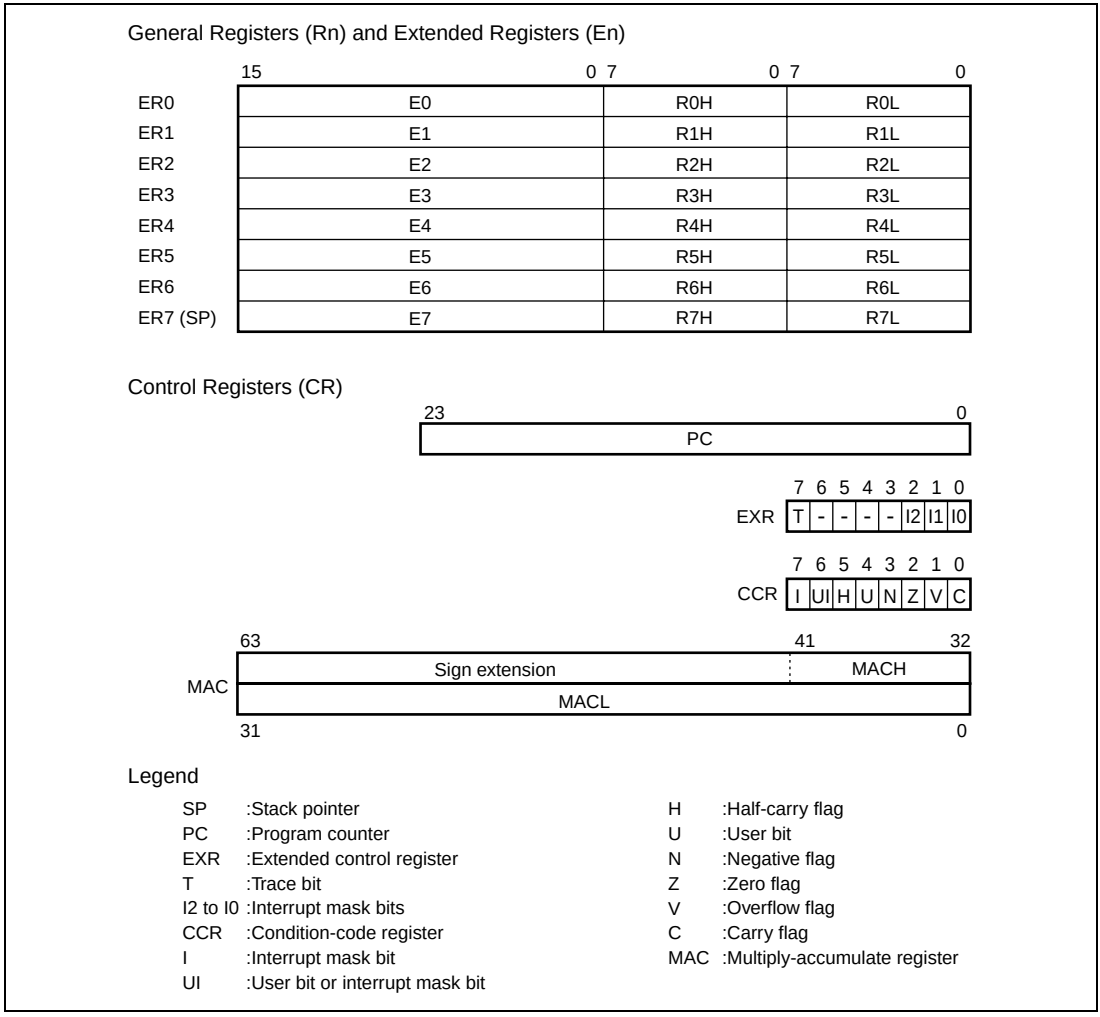



**Figure 2.6 CPU Registers**

### 2.4.1 General Registers

The H8S/2600 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 2.7 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 general 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 2.8 shows the stack.

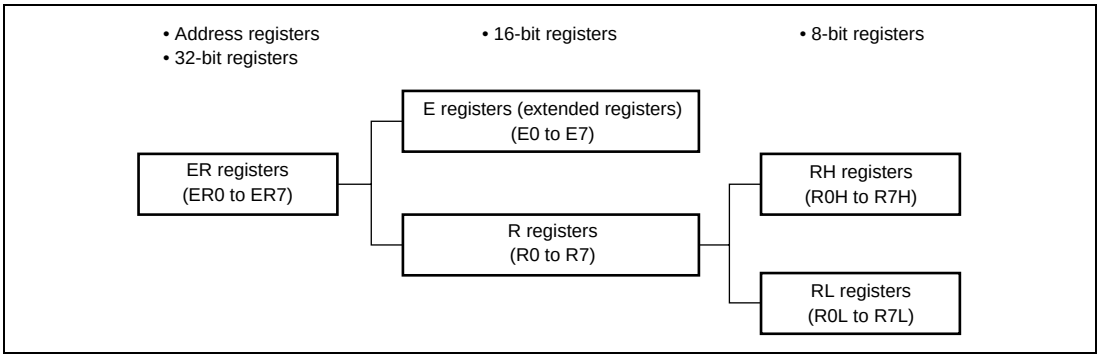



**Figure 2.7 Usage of General Registers**

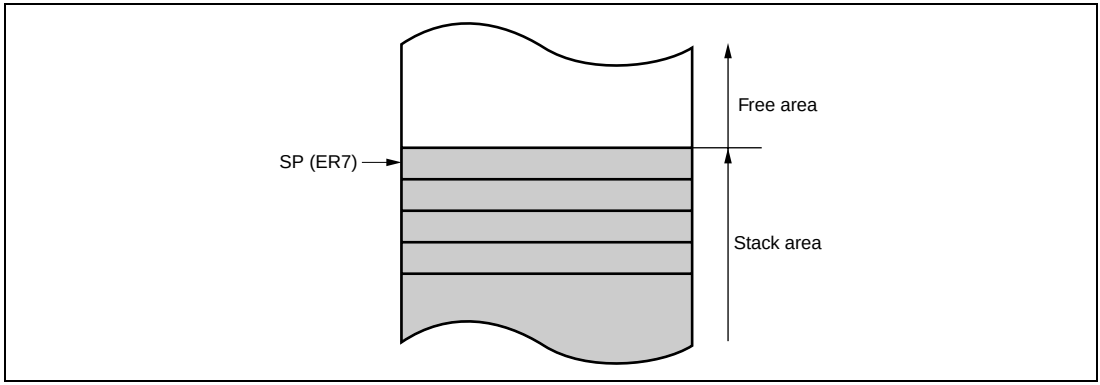



**Figure 2.8 Stack**

### 2.4.2 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).

### 2.4.3 Extended Control Register (EXR)

EXR is an 8-bit register that manipulates the LDC, STC, ANDC, ORC, and XORC instructions. When these instructions, except for the STC instruction, are executed, all interrupts including NMI will be masked for three states after execution is completed.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | T | 0 | R/W | Trace Bit<br>When this bit is set to 1, a trace exception is generated each time an instruction is executed. When this bit is cleared to 0, instructions are executed in sequence. |
| 6 to 3 | — | All 1 | — | Reserved<br>Theese bits are always read as 1. |
| 2 | I2 | 1 | R/W | These bits designate the interrupt mask level (0 to 7). For details, refer to section 5, Interrupt Controller. |
| 1 | I1 | 1 | R/W | |
| 0 | I0 | 1 | R/W | |

### 2.4.4 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.

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.

| 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. For details, refer to section 5, Interrupt Controller. |
| 6 | UI | undefined | R/W | User Bit or Interrupt Mask Bit<br>Can be read or written by software using the LDC, STC, ANDC, ORC, and XORC instructions. This bit cannot be used as an interrupt mask bit in this LSI. |
| 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 read or written 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. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 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. |

### 2.4.5 Multiply-Accumulate Register (MAC)

This 64-bit register stores the results of multiply-and-accumulate operations. It consists of two 32-bit registers denoted MACH and MACL. The lower 10 bits of MACH are valid; the upper bits are a sign extension.

### 2.4.6 Initial Values of CPU Registers

Reset exception handling loads the CPU's program counter (PC) from the vector table, clears the trace bit in EXR to 0, and sets the interrupt mask bits in CCR and EXR to 1. The other CCR bits and the general registers are not initialized. In particular, the stack pointer (ER7) is not initialized. The stack pointer should therefore be initialized by an MOV.L instruction executed immediately after a reset.

## 2.5 Data Formats

The H8S/2600 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.

### 2.5.1 General Register Data Formats

Figure 2.9 shows the data formats in general registers.

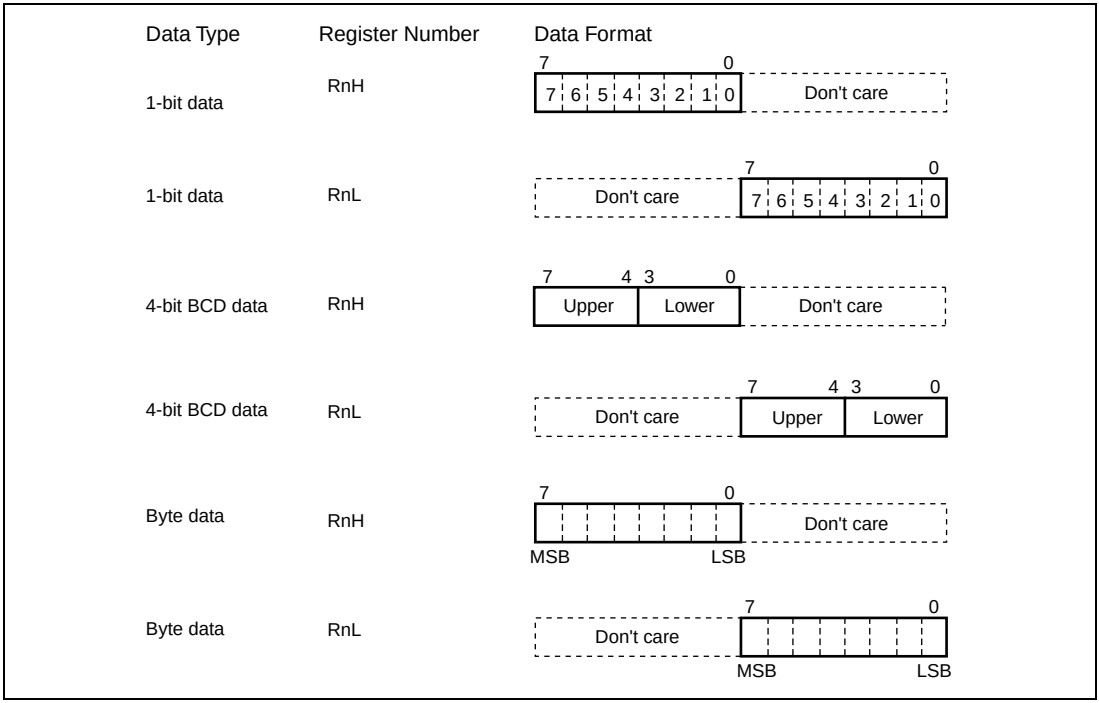



**Figure 2.9 General Register Data Formats (1)**

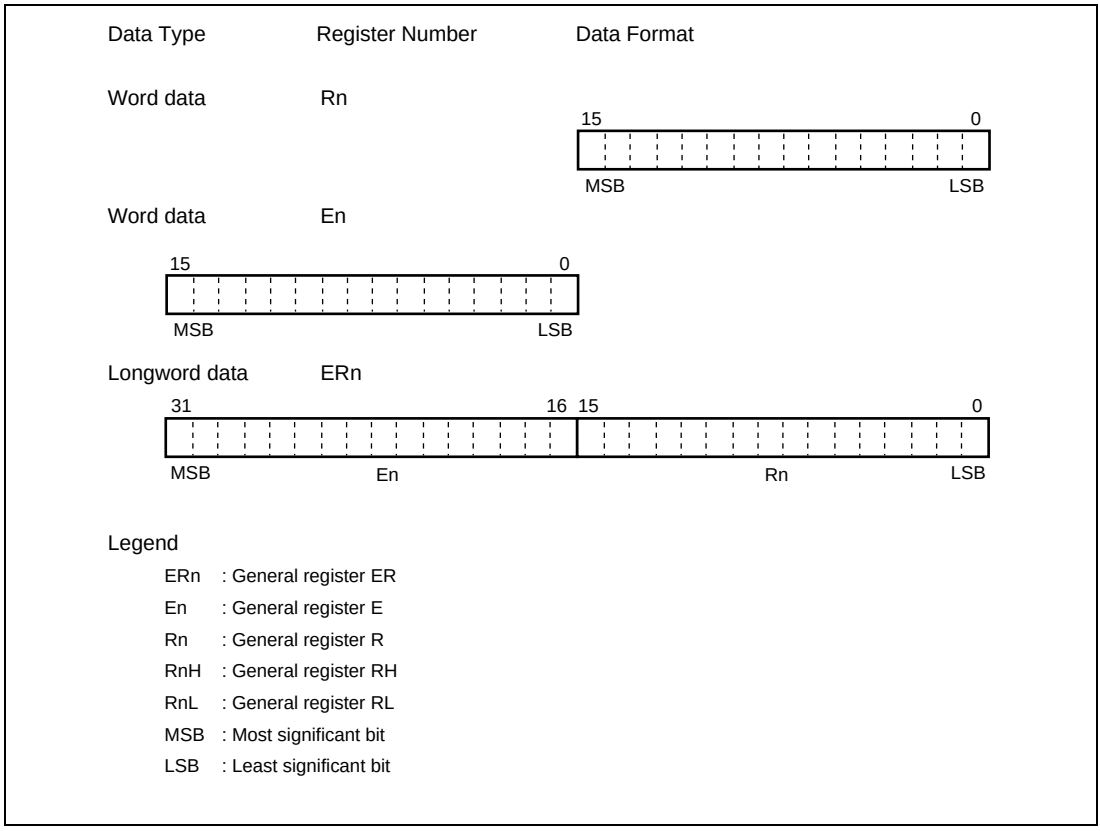



**Figure 2.9 General Register Data Formats (2)**

### 2.5.2 Memory Data Formats

Figure 2.10 shows the data formats in memory. The H8S/2600 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 is used as an address register to access the stack, the operand size should be word or longword.

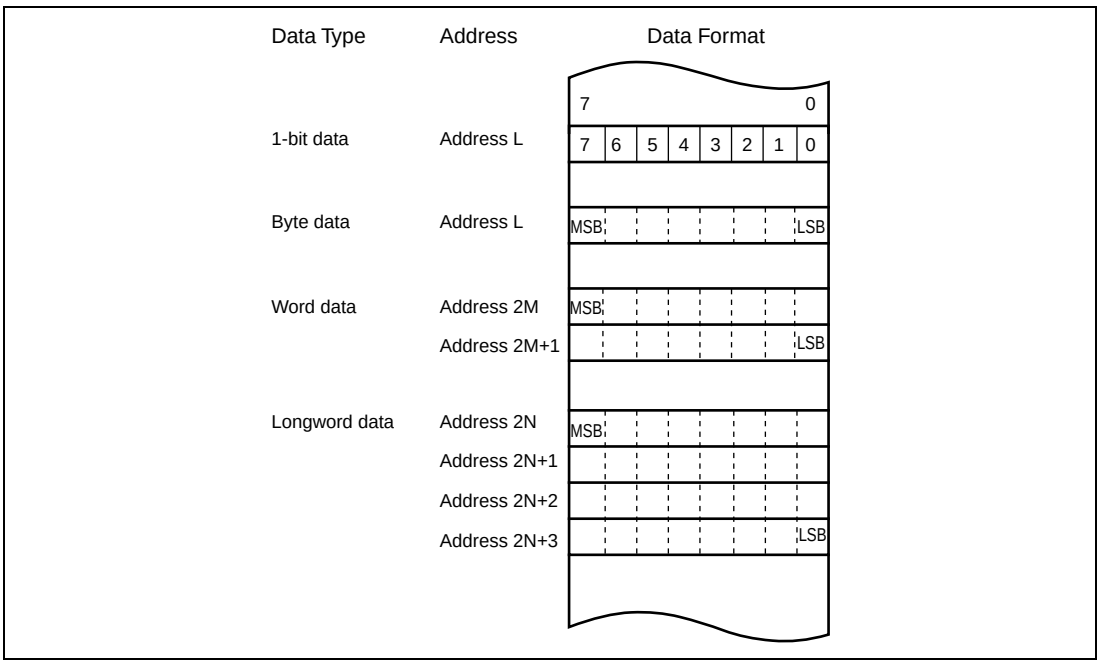



**Figure 2.10 Memory Data Formats**

## 2.6 Instruction Set

The H8S/2600 CPU has 69 instructions. The instructions are classified by function in table 2.1.

**Table 2.1 Instruction Classification**

| Function | Instructions | Size | Types |
|---|---|---|---|
| Data transfer | MOV | B/W/L | 5 |
| | POP*[1], PUSH*[1] | W/L | |
| | LDM, STM | L | |
| | MOVFPE*[3], MOVTPE*[3] | B | |
| Arithmetic operations | ADD, SUB, CMP, NEG | B/W/L | 23 |
| | ADDX, SUBX, DAA, DAS | B | |
| | INC, DEC | B/W/L | |
| | ADDS, SUBS | L | |
| | MULXU, DIVXU, MULXS, DIVXS | B/W | |
| | EXTU, EXTS | W/L | |
| | TAS*[4] | B | |
| | MAC, LDMAC, STMAC, CLRMAC | — | |
| Logic operations | AND, OR, XOR, NOT | B/W/L | 4 |
| Shift | SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR | B/W/L | 8 |
| Bit manipulation | BSET, BCLR, BNOT, BTST, BLD, BILD, BST, BIST, BAND, BIAND, BOR, BIOR, BXOR, BIXOR | B | 14 |
| Branch | Bcc*[2], JMP, BSR, JSR, RTS | — | 5 |
| System control | TRAPA, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP | — | 9 |
| Block data transfer | EEPMOV | — | 1 |
| | | | Total: 69 |

Notes: B-byte; W-word; L-longword.

1. POP.W Rn and PUSH.W Rn are identical to MOV.W @SP+,Rn and MOV.W Rn,@-SP. POP.L ERn and PUSH.L ERn are identical to MOV.L @SP+,ERn and MOV.L ERn,@-SP.
2. Bcc is the general name for conditional branch instructions.
3. Cannot be used in this LSI.
4. Only register ER0, ER1, ER4, or ER5 should be used when using the TAS instruction.

### 2.6.1 Table of Instructions Classified by Function

Tables 2.3 to 2.10 summarize the instructions in each functional category. The notation used in tables 2.3 to 2.10 is defined below.

**Table 2.2 Operation Notation**

| Symbol | Description |
|---|---|
| Rd | General register (destination)* |
| Rs | General register (source)* |
| Rn | General register* |
| ERn | General register (32-bit register) |
| MAC | Multiply-accumulate register (32-bit register) |
| (EAd) | Destination operand |
| (EAs) | Source operand |
| EXR | Extended control register |
| 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) |
| :8/:16/:24/:32 | 8-, 16-, 24-, or 32-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 (ER0 to ER7).

**Table 2.3 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 | Cannot be used in this LSI. |
| MOVTPE | B | 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. |
| LDM | L | @SP+ → Rn (register list)<br>Pops two or more general registers from the stack. |
| STM | L | Rn (register list) → @–SP<br>Pushes two or more general registers onto the stack. |

Note:* Refers to the operand size.

B: Byte

W: Word

L: Longword

**Table 2.4 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 2.4 Arithmetic Operations Instructions (2)**

| Instruction | Size*[1] | 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. |
| TAS*[2] | B | @ERd – 0, 1 → (<bit 7> of @ERd)<br>Tests memory contents, and sets the most significant bit (bit 7) to 1. |
| MAC | — | (EAs) × (EAd) + MAC → MAC<br>Performs signed multiplication on memory contents and adds the result to the multiply-accumulate register. The following operations can be performed:<br>16 bits × 16 bits + 32 bits → 32 bits, saturating<br>16 bits × 16 bits + 42 bits → 42 bits, non-saturating |
| CLRMAC | — | 0 → MAC<br>Clears the multiply-accumulate register to zero. |
| LDMAC<br>STMAC | L | Rs → MAC, MAC → Rd<br>Transfers data between a general register and a multiply-accumulate register. |

Note: 1. Refers to the operand size.
B: Byte
W: Word
L: Longword
2. Only register ER0, ER1, ER4, or ER5 should be used when using the TAS instruction.

**Table 2.5 Logic Operations Instructions**

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

Note:* Refers to the operand size.

B: Byte

W: Word

L: Longword

**Table 2.6 Shift Instructions**

| Instruction | Size* | Function |
|---|---|---|
| SHAL<br>SHAR | B/W/L | Rd (shift) → Rd<br>Performs an arithmetic shift on general register contents.<br>1-bit or 2-bit shifts are possible. |
| SHLL<br>SHLR | B/W/L | Rd (shift) → Rd<br>Performs a logical shift on general register contents.<br>1-bit or 2-bit shifts are possible. |
| ROTL<br>ROTR | B/W/L | Rd (rotate) → Rd<br>Rotates general register contents.<br>1-bit or 2-bit rotations are possible. |
| ROTXL<br>ROTXR | B/W/L | Rd (rotate) → Rd<br>Rotates general register contents through the carry flag.<br>1-bit or 2-bit rotations are possible. |

Note:* Refers to the operand size.

B: Byte

W: Word

L: Longword

**Table 2.7 Bit Manipulation Instructions (1)**

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

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

**Table 2.7 Bit Manipulation Instructions (2)**

| Instruction | Size*[1] | Function |
|---|---|---|
| BXOR | B | C ⊕ (<bit-No.> of <EAd>) → C<br>XORs the carry flag with a specified bit in a general register or memory operand and stores the result in the carry flag. |
| BIXOR | B | C ⊕ [~(<bit-No.> of <EAd>)] → C<br>XORs the carry flag with the inverse of a specified bit in a general register or memory operand and stores the result in the carry flag.<br>The bit number is specified by 3-bit immediate data. |
| BLD | B | (<bit-No.> of <EAd>) → C<br>Transfers a specified bit in a general register or memory operand to the carry flag. |
| BILD | B | ~(<bit-No.> of <EAd>) → C<br>Transfers the inverse of a specified bit in a general register or memory operand to the carry flag.<br>The bit number is specified by 3-bit immediate data. |
| BST | B | C → (<bit-No.> of <EAd>)<br>Transfers the carry flag value to a specified bit in a general register or memory operand. |
| BIST | B | ~C → (<bit-No.> of <EAd>)<br>Transfers the inverse of the carry flag value to a specified bit in a general register or memory operand.<br>The bit number is specified by 3-bit immediate data. |

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

**Table 2.8 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 |

**Table 2.9 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, (EAs) → EXR<br>Moves general register or memory contents or immediate data to CCR or EXR. Although CCR and EXR are 8-bit registers, word-size transfers are performed between them and memory. The upper 8 bits are valid. |
| STC | B/W | CCR → (EAd), EXR → (EAd)<br>Transfers CCR or EXR contents to a general register or memory. Although CCR and EXR are 8-bit registers, word-size transfers are performed between them and memory. The upper 8 bits are valid. |
| ANDC | B | CCR ∧ #IMM → CCR, EXR ∧ #IMM → EXR<br>Logically ANDs the CCR or EXR contents with immediate data. |
| ORC | B | CCR ∨ #IMM → CCR, EXR ∨ #IMM → EXR<br>Logically ORs the CCR or EXR contents with immediate data. |
| XORC | B | CCR ⊕ #IMM → CCR, EXR ⊕ #IMM → EXR<br>Logically XORs the CCR or EXR contents with immediate data. |
| NOP | — | PC + 2 → PC<br>Only increments the program counter. |

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

**Table 2.10 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; |
| | | 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. |

### 2.6.2 Basic Instruction Formats

The H8S/2600 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 2.11 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.

- Condition Field

  Specifies the branching condition of Bcc instructions.

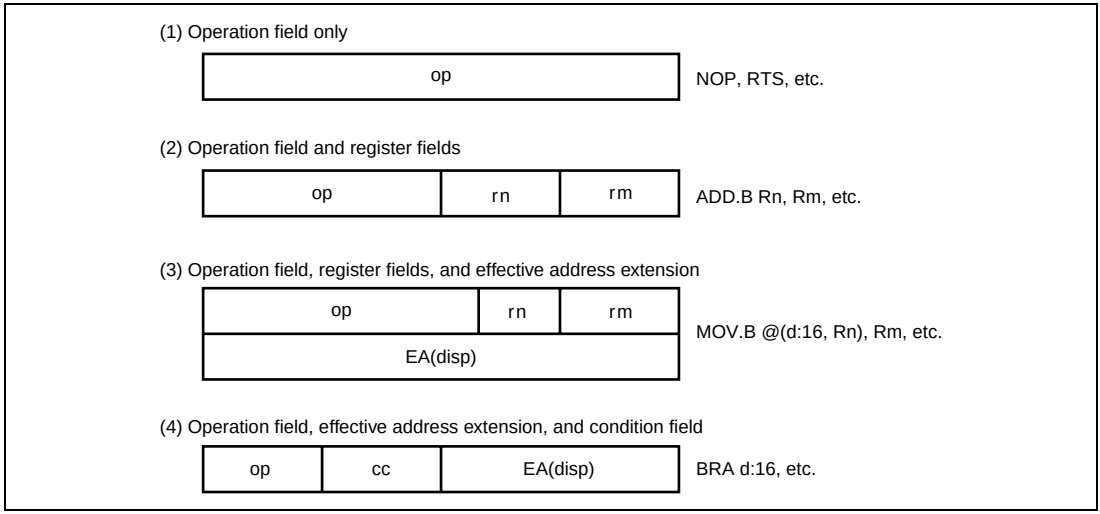



**Figure 2.11 Instruction Formats (Examples)**

## 2.7 Addressing Modes and Effective Address Calculation

The H8S/2600 CPU supports the eight addressing modes listed in table 2.11. Each instruction uses a subset of these 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 2.11 Addressing Modes**

| No. | Addressing Mode | Symbol |
|---|---|---|
| 1 | Register direct | Rn |
| 2 | Register indirect | @ERn |
| 3 | Register indirect with displacement | @(d:16,ERn)/@(d:32,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/@aa:32 |
| 6 | Immediate | #xx:8/#xx:16/#xx:32 |
| 7 | Program-counter relative | @(d:8,PC)/@(d:16,PC) |
| 8 | Memory indirect | @@aa:8 |

### 2.7.1 Register Direct—Rn

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

### 2.7.2 Register Indirect—@ERn

The register field of the instruction code specifies an address register (ERn) which contains the address of the operand on memory. If the address is a program instruction address, the lower 24 bits are valid and the upper 8 bits are all assumed to be 0 (H′00).

### 2.7.3 Register Indirect with Displacement—@(d:16, ERn) or @(d:32, ERn)

A 16-bit or 32-bit displacement contained in the instruction is added to an address register (ERn) specified by the register field of the instruction, and the sum gives the address of a memory operand. A 16-bit displacement is sign-extended when added.

### 2.7.4 Register Indirect with Post-Increment or Pre-Decrement—@ERn+ or @-ERn

**Register indirect with post-increment—@ERn+:** The register field of the instruction code specifies an address register (ERn) which contains the address of a memory operand. After the operand is accessed, 1, 2, or 4 is added to the address register contents and the sum is stored in the address register. The value added is 1 for byte access, 2 for word transfer instruction, or 4 for longword transfer instruction. For the word or longword transfer instructions, 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 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 transfer instruction, or 4 for longword transfer instruction. For the word or longword transfer instructions, the register value should be even.

### 2.7.5 Absolute Address—@aa:8, @aa:16, @aa:24, or @aa:32

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), or 32 bits long (@aa:32). Table 2.12 indicates the accessible absolute address ranges.

To access data, the absolute address should be 8 bits (@aa:8), 16 bits (@aa:16), or 32 bits (@aa:32) long. For an 8-bit absolute address, the upper 24 bits are all assumed to be 1 (H′FFFF). For a 16-bit absolute address the upper 16 bits are a sign extension. A 32-bit absolute address can access the entire address space.

A 24-bit absolute address (@aa:24) indicates the address of a program instruction. The upper 8 bits are all assumed to be 0 (H′00).

**Table 2.12 Absolute Address Access Ranges**

| Absolute Address | | Normal Mode* | Advanced Mode |
|---|---|---|---|
| Data address | 8 bits (@aa:8) | H′FF00 to H′FFFF | H′FFFF00 to H′FFFFFF |
| | 16 bits (@aa:16) | H′0000 to H′FFFF | H′000000 to H′007FFF, H′FF8000 to H′FFFFFF |
| | 32 bits (@aa:32) | | H′000000 to H′FFFFFF |
| Program instruction address | 24 bits (@aa:24) | | |

Note: Normal mode is not available in this LSI.

### 2.7.6 Immediate—#xx:8, #xx:16, or #xx:32

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

The ADDS, SUBS, INC, and DEC instructions contain immediate data implicitly. Some bit manipulation instructions contain 3-bit immediate data in the instruction code, specifying a bit number. The TRAPA instruction contains 2-bit immediate data in its instruction code, specifying a vector address.

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

This mode is used in the Bcc and BSR instructions. 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. Only the lower 24 bits of this branch address are valid; the upper 8 bits are all assumed to be 0 (H′00). 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.

### 2.7.8 Memory Indirect—@@aa:8

This mode can be used by the JMP and JSR instructions. The instruction code contains an 8-bit absolute address specifying a memory operand. This memory operand contains a branch address. The 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 in normal mode, H′000000 to H′0000FF in advanced mode). In normal mode, the memory operand is a word operand and the branch address is 16 bits long. In advanced mode, the memory operand is a longword operand, the first byte of which is assumed to be 0 (H′00).

Note that the first part of the address range is also the exception vector area. For further details, refer to section 4, Exception Handling.

If an odd address is specified in word or longword memory access, or as a branch address, the least significant bit is regarded as 0, causing data to be accessed or instruction code to be fetched at the address preceding the specified address. (For further information, see section 2.5.2, Memory Data Formats.)

Note: Normal mode is not available in this LSI.

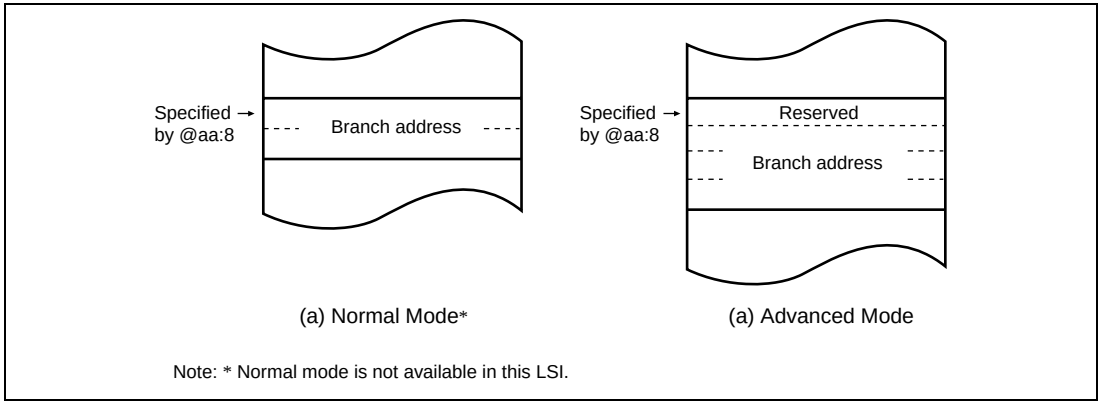


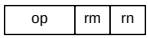
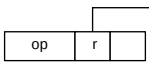
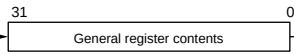
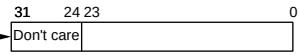
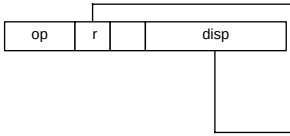
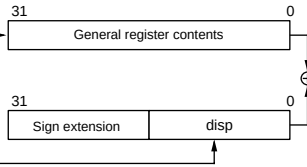
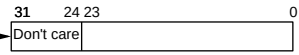
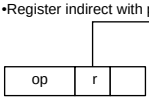
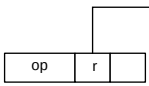
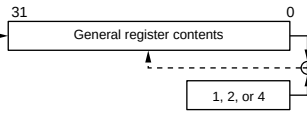
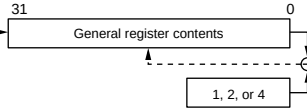
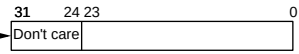
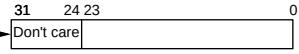

**Figure 2.12 Branch Address Specification in Memory Indirect Mode**

### 2.7.9 Effective Address Calculation

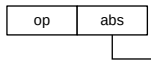
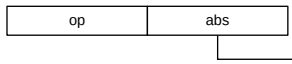
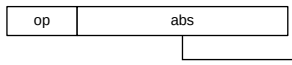
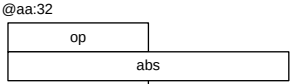
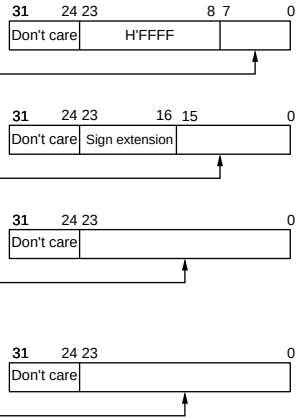
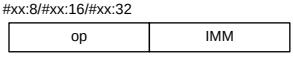
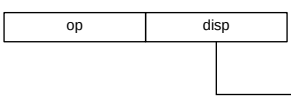
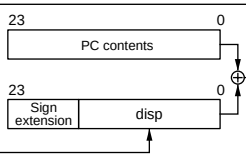
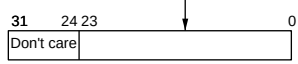
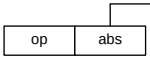
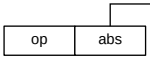
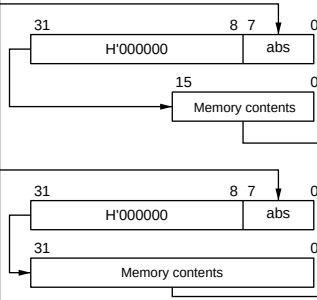
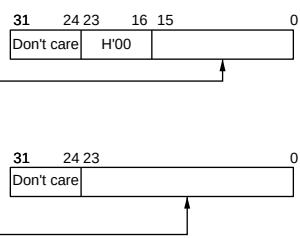
Table 2.13 indicates how effective addresses are calculated in each addressing mode. In normal mode the upper 8 bits of the effective address are ignored in order to generate a 16-bit address.

Note : Normal mode is not available in this LSI.

## Table 2.13 Effective Address Calculation (1)

| No | Addressing Mode and Instruction Format | Effective Address Calculation | Effective Address (EA) |
|---|---|---|---|
| 1 | Register direct(Rn)<br>  | | Operand is general register contents. |
| 2 | Register indirect(@ERn)<br>  |  |  |
| 3 | Register indirect with displacement<br>@(d:16,ERn) or @(d:32,ERn)<br>  |  |  |
| 4 | Register indirect with post-increment or<br>pre-decrement<br>•Register indirect with post-increment @ERn+<br>  <br>•Register indirect with pre-decrement @-ERn<br>  |  <br>  <br>Operand Size / Offset<br>Byte 1<br>Word 2<br>Longword 4 |  <br>  |

## Table 2.13 Effective Address Calculation (2)

| No | Addressing Mode and Instruction Format | Effective Address Calculation | Effective Address (EA) |
|---|---|---|---|
| 5 | Absolute address<br>@aa:8<br>  <br>@aa:16<br>  <br>@aa:24<br>  <br>@aa:32<br>  | |  |
| 6 | Immediate<br>#xx:8/#xx:16/#xx:32<br>  | | Operand is immediate data. |
| 7 | Program-counter relative<br>@(d:8,PC) @(d:16,PC)<br>  |  |  |
| 8 | Memory indirect @@aa:8<br>• Normal mode*<br>  <br>• Advanced mode<br>  |  |  |

Note: * Normal mode is not available in this LSI.

## 2.8 Processing States

The H8S/2600 CPU has five main processing states: the reset state, exception handling state, program execution state, bus-released state, and power-down state. Figure 2.13 indicates the state transitions.

- Reset State

  In this state, the CPU and all on-chip peripheral modules are initialized and not operating. When the $\overline{\text{RES}}$ input goes low, all current processing stops and the CPU enters the reset state. All interrupts are masked in the reset state. Reset exception handling starts when the $\overline{\text{RES}}$ signal changes from low to high. For details, refer to section 4, Exception Handling.

  The reset state can also be entered by a watchdog timer overflow.

- Exception-Handling State

  The exception-handling state is a transient state that occurs when the CPU alters the normal processing flow due to an exception source, such as a reset, trace, interrupt, or trap instruction. The CPU fetches a start address (vector) from the exception vector table and branches to that address. For further details, refer to section 4, Exception Handling.

- Program Execution State

  In this state, the CPU executes program instructions in sequence.

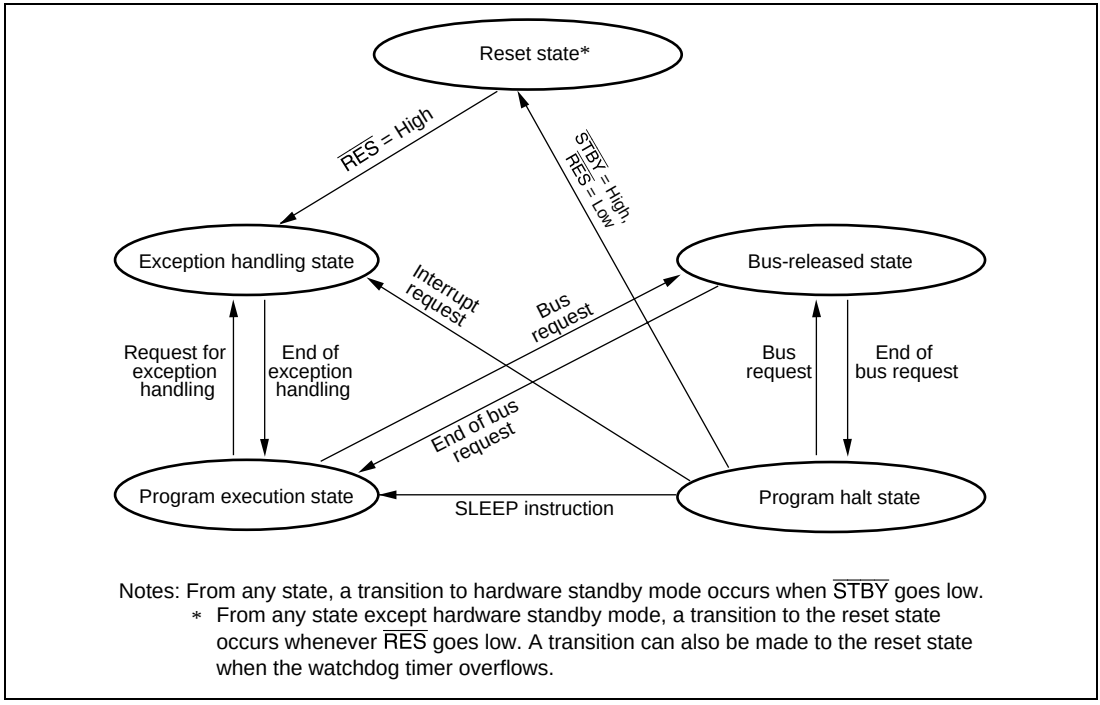
- Bus-Released State

  The bus has been released in response to a bus request from a bus master other than the CPU.

  While the bus is released, the CPU halts operations.

- Program stop state

  This is a power-down state in which the CPU stops operating. The program stop state occurs when a SLEEP instruction is executed or the CPU enters hardware standby mode. For further details, refer to section 21, Power-Down Modes.




Notes: From any state, a transition to hardware standby mode occurs when $\overline{STBY}$ goes low.

* From any state except hardware standby mode, a transition to the reset state occurs whenever $\overline{RES}$ goes low. A transition can also be made to the reset state when the watchdog timer overflows.

**Figure 2.13 State Transitions**

## 2.9 Usage Note

### 2.9.1 Notes on Using the Bit Operation Instruction

Instructions BSET, BCLR, BNOT, BST, and BIST read data in byte units, and write data in byte units after bit operation. Therefore, attention must be paid when these instructions are used for ports or registers including write-only bits.

Instruction BCLR can be used to clear the flag in the internal I/O register to 0. If it is obvious that the flag has been set to 1 by the interrupt processing routine, it is unnecessary to read the flag beforehand.

# Section 3 MCU Operating Modes

## 3.1 Operating Mode Selection

This LSI supports only operating mode 7, that is, the advanced single-chip mode. The operating mode is determined by the setting of the mode pins (MD2 to MD0). Only mode 7 can be used in this LSI. Therefore, all mode pins must be fixed high, as shown in table 3.1. Do not change the mode pin settings during operation.

**Table 3.1 MCU Operating Mode Selection**

| MCU Operating Mode | MD2 | MD1 | MD0 | CPU Operating Mode | Description | On-Chip ROM | External Data Bus: Initial Width | External Data Bus: Max. Width |
|---|---|---|---|---|---|---|---|---|
| 7 | 1 | 1 | 1 | Advanced mode | Single-chip mode | Enabled | — | — |

## 3.2 Register Descriptions

The following registers are related to the operating mode.

- Mode control register(MDCR)
- System control register(SYSCR)

### 3.2.1 Mode Control Register (MDCR)

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 7 | — | 1 | R/W | Reserved<br>Only 1 should be written to this bit. |
| 6 to 3 | — | All 0 | — | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 2<br>1<br>0 | MDS2<br>MDS1<br>MDS0 | —<br>—<br>— | R<br>R<br>R | Mode select 2 to 0<br>These bits indicate the input levels at pins MD2 to MD0 (the current operating mode). Bits MDS2 to MDS0 correspond to MD2 to MD0. MDS2 to MDS0 are read-only bits and they cannot be written to. The mode pin (MD2 to MD0) input levels are latched into these bits when MDCR is read. These latches are canceled by a reset. These latches are canceled by a reset. |

### 3.2.2 System Control Register (SYSCR)

SYSCR is an 8-bit readable/writable register that selects saturating or non-saturating calculation for the MAC instruction, selects the interrupt control mode and the detected edge for NMI, and enables or disables on-chip RAM.

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 7 | MACS | 0 | R/W | MAC Saturation<br>Selects either saturating or non-saturating calculation for the MAC instruction.<br>0: Non-saturating calculation for the MAC instruction<br>1: Saturating calculation for the MAC instruction |
| 6 | — | 0 | — | Reserved<br>This bit is always read as 0 and cannot be modified. |
| 5<br>4 | INTM1<br>INTM0 | 0<br>0 | R/W<br>R/W | These bits select the control mode of the interrupt controller. For details of the interrupt control modes, see section 5.6, Interrupt Control Modes and Interrupt Operation.<br>00: Interrupt control mode 0<br>01: Setting prohibited<br>10: Interrupt control mode 2<br>11: Setting prohibited |
| 3 | NMIEG | 0 | R/W | NMI Edge Select<br>Selects the valid edge of the NMI interrupt input.<br>0: An interrupt is requested at the falling edge of NMI input<br>1: An interrupt is requested at the rising edge of NMI input |
| 2,<br>1 | — | All 0 | — | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 0 | RAME | 1 | R/W | RAM Enable<br>Enables or disables on-chip RAM. The RAME bit is initialized when the reset status is released.<br>0: On-chip RAM is disabled<br>1: On-chip RAM is enabled |

## 3.3 Pin Functions in Each Operating Mode

The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is enabled, however external addresses cannot be accessed.

All I/O ports are available for use as input-output ports.

## 3.4 Address Map

Figure 3.1 shows the address map in each operating mode.

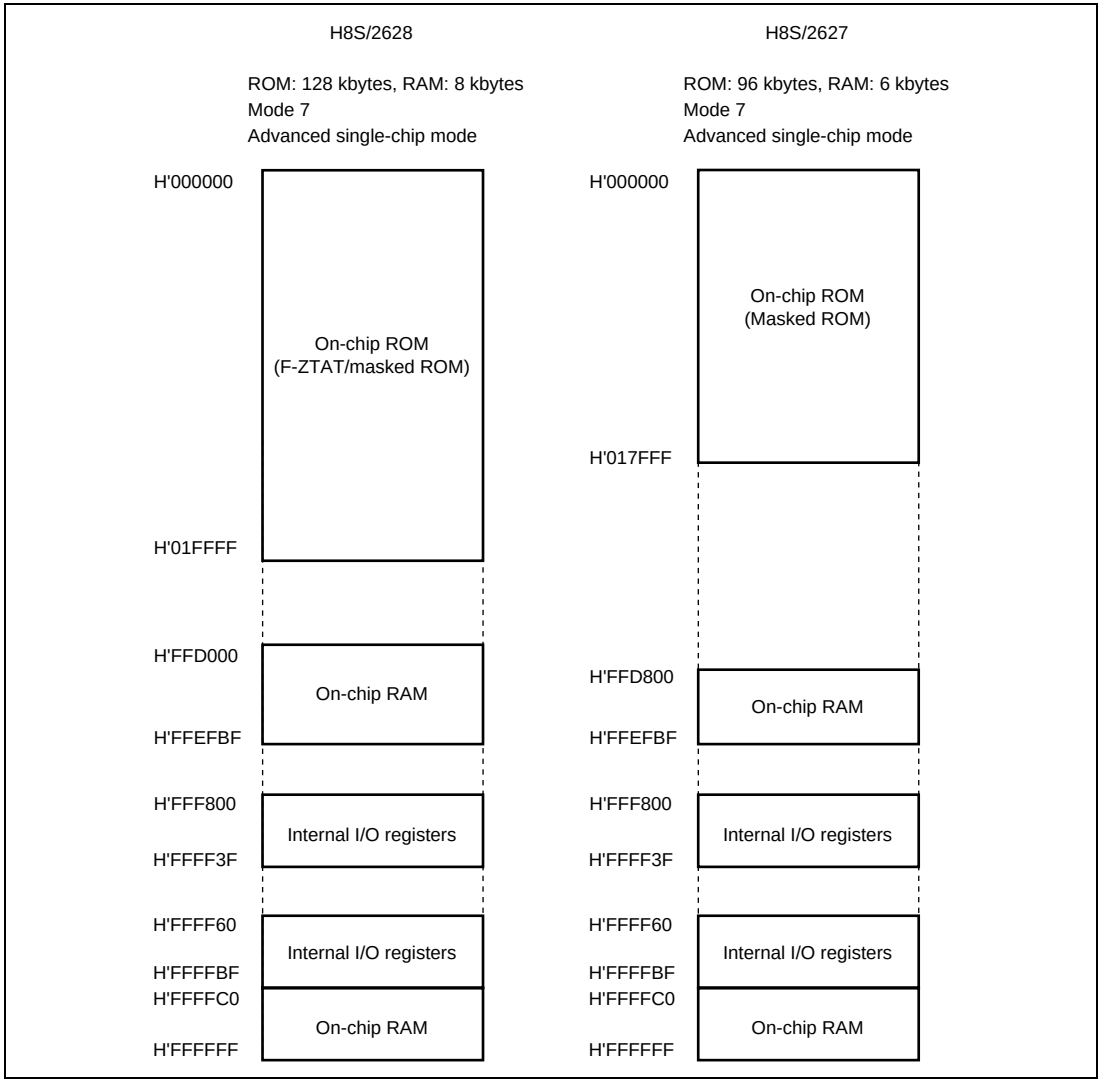



**Figure 3.1 Address Map**

# Section 4 Exception Handling

## 4.1 Exception Handling Types and Priority

As shown in table 4.1, exception handling may be caused by a reset, trace, interrupt, or trap instruction. Exception handling is prioritized as shown in table 4.1. If two or more exceptions occur simultaneously, they are accepted and processed in order of priority. Exception sources, the stack structure, and operation of the CPU vary depending on the interrupt control mode. For details on the interrupt control mode, refer to section 5, Interrupt Controller.

**Table 4.1 Exception Types and Priority**

| Priority | Exception Type | Start of Exception Handling |
|---|---|---|
| High ↑ | Reset | Starts immediately after a low-to-high transition at the $\overline{\text{RES}}$ pin, or when the watchdog timer overflows. The CPU enters the reset state when the $\overline{\text{RES}}$ pin is low. |
| | Trace*[1] | Starts when execution of the current instruction or exception handling ends, if the trace (T) bit in EXR is set to 1. |
| | Direct transition | Starts when a direction transition occurs as the result of SLEEP instruction execution. |
| | Interrupt | Starts when execution of the current instruction or exception handling ends, if an interrupt request has been issued.*[2] |
| Low | Trap instruction *[3] | Started by execution of a trap instruction (TRAPA). |

Notes: 1. Traces are enabled only in interrupt control mode 2. Trace exception handling is not executed after execution of an RTE instruction.
2. Interrupt detection is not performed on completion of ANDC, ORC, XORC, or LDC instruction execution, or on completion of reset exception handling.
3. Trap instruction exception handling requests are accepted at all times in program execution state.

## 4.2 Exception Sources and Exception Vector Table

Different vector addresses are assigned to different exception sources. Table 4.2 lists the exception sources and their vector addresses. Since the usable modes differ depending on the product, for details on each product, refer to section 3, MCU Operating Modes.

**Table 4.2 Exception Handling Vector Table**

| Exception Source | | Vector Number | Vector Address*[1] Normal Mode | Vector Address*[1] Advanced Mode |
|---|---|---|---|---|
| Power-on reset | | 0 | H′0000 to H′0001 | H′0000 to H′0003 |
| Manual reset *[2] | | 1 | H′0002 to H′0003 | H′0004 to H′0007 |
| Reserved for system use | | 2 | H′0004 to H′0005 | H′0008 to H′000B |
| | | 3 | H′0006 to H′0007 | H′000C to H′000F |
| | | 4 | H′0008 to H′0019 | H′0010 to H′0013 |
| Trace | | 5 | H′000A to H′000B | H′0014 to H′0017 |
| Interrupt (direct transitions)*[2] | | 6 | H′000C to H′000D | H′0018 to H′001B |
| Interrupt (NMI) | | 7 | H′000E to H′000F | H′001C to H′001F |
| Trap instruction (#0) | | 8 | H′0010 to H′0011 | H′0020 to H′0023 |
| (#1) | | 9 | H′0012 to H′0013 | H′0024 to H′0027 |
| (#2) | | 10 | H′0014 to H′0015 | H′0028 to H′002B |
| (#3) | | 11 | H′0016 to H′0017 | H′002C to H′002F |
| Reserved for system use | | 12 | H′0018 to H′0019 | H′0030 to H′0033 |
| | | 13 | H′001A to H′001B | H′0034 to H′0037 |
| | | 14 | H′001C to H′001D | H′0038 to H′003B |
| | | 15 | H′001E to H′001F | H′003C to H′003F |
| External interrupt | IRQ0 | 16 | H′0020 to H′0021 | H′0040 to H′0043 |
| | IRQ1 | 17 | H′0022 to H′0023 | H′0044 to H′0047 |
| | IRQ2 | 18 | H′0024 to H′0025 | H′0048 to H′004B |
| | IRQ3 | 19 | H′0026 to H′0027 | H′004C to H′004F |
| | IRQ4 | 20 | H′0028 to H′0029 | H′0050 to H′0053 |
| | IRQ5 | 21 | H′002A to H′002B | H′0054 to H′0057 |
| Reserved for system use | | 22 | H′002C to H′002D | H′0058 to H′005B |
| | | 23 | H′002E to H′002F | H′005C to H′005F |
| Internal interrupt*[3] | | 24<br>⎮<br>127 | H′0030 to H′0031<br>⎮<br>H′00FE to H′00FF | H′0060 to H′0063<br>⎮<br>H′01FC to H′01FF |

Notes: 1. Lower 16 bits of the address.
2. Not available in this LSI.
3. For details of internal interrupt vectors, see section 5.5, Interrupt Exception Handling Vector Table.

## 4.3 Reset

A reset has the highest exception priority.

When the $\overline{\text{RES}}$ pin goes low, all processing halts and this LSI enters the reset state. To ensure that this LSI is reset, hold the $\overline{\text{RES}}$ pin low for at least 20 ms at power-up. To reset the chip during operation, hold the $\overline{\text{RES}}$ pin low for at least 20 states. A reset initializes the internal state of the CPU and the registers of on-chip peripheral modules.

The chip can also be reset by overflow of the watchdog timer. For details, see section 13, Watchdog Timer.

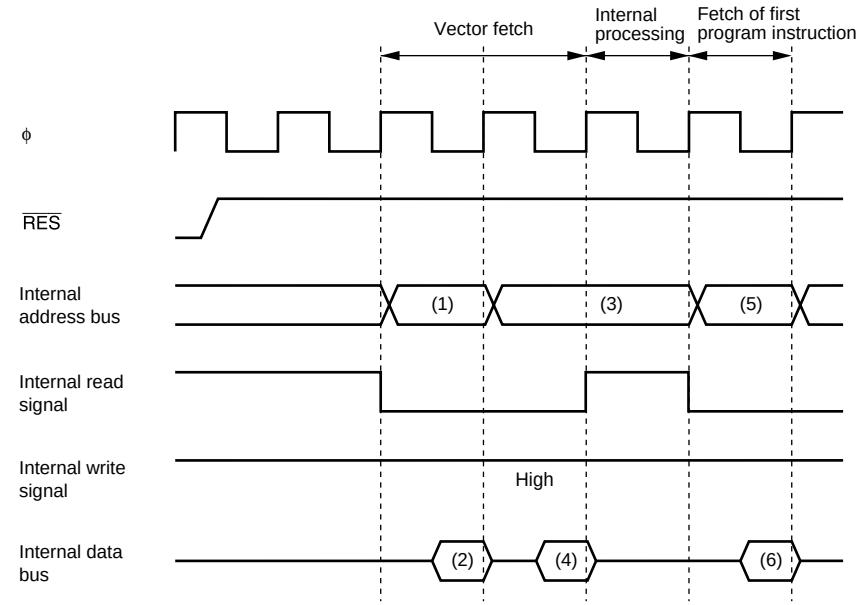
The interrupt control mode is 0 immediately after reset.

### 4.3.1 Reset Exception Handling

When the $\overline{\text{RES}}$ pin goes high after being held low for the necessary period, this LSI starts reset exception handling as follows:

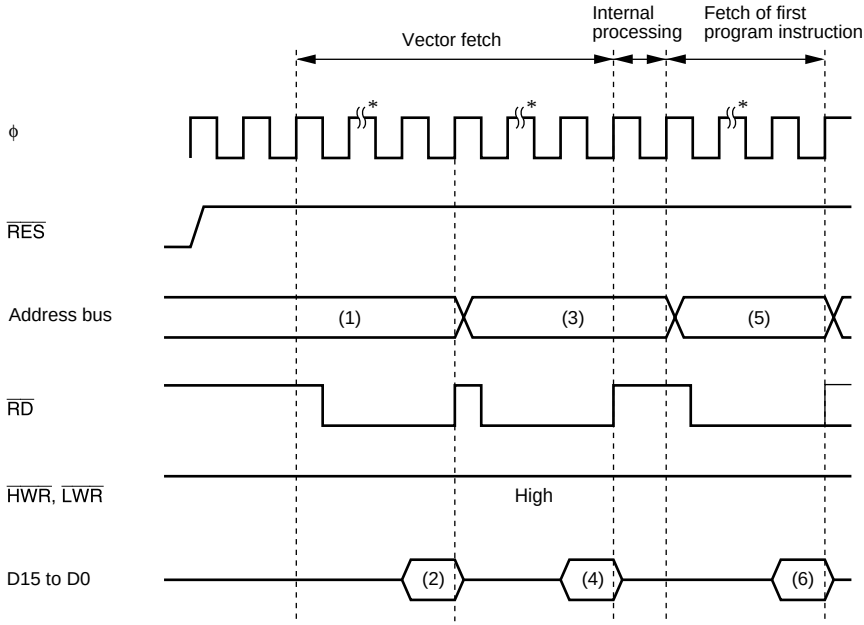
1. The internal state of the CPU and the registers of the on-chip peripheral modules are initialized, the T bit in EXR is cleared to 0, and the I bit in EXR and CCR is set to 1.
2. The reset exception handling vector address is read and transferred to the PC, and program execution starts from the address indicated by the PC.

Figures 4.1 and 4.2 show examples of the reset sequence.




(1)(3) Reset exception handling vector address (when reset, (1)=H'000000, (3)=H'000002)
(2)(4) Start address (contents of reset exception handling vector address)
(5) Start address ((5)=(2)(4))
(6) First program instruction

**Figure 4.1 Reset Sequence (Advanced Mode with On-chip ROM Enabled)**




(1)(3) Reset exception handling vector address (when reset, (1)=H'000000, (3)=H'000002)
(2)(4) Start address (contents of reset exception handling vector address)
(5) Start address ((5)=(2)(4))
(6) First program instruction

Note:* Three program wait states are inserted.

**Figure 4.2 Reset Sequence (Advanced Mode with On-chip ROM Disabled: Not Available in this LSI)**

### 4.3.2 Interrupts after Reset

If an interrupt is accepted immediately 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 exception handling is executed. Since the first instruction of a program is always executed immediately after the reset, make sure that this instruction initializes the stack pointer (example: MOV.L #xx: 32, SP).

### 4.3.3 State of On-Chip Peripheral Modules after Reset Release

After reset release, MSTPCRA to MSTPCRC are initialized to H′3F, H′FF, and H′FF, respectively, and all modules except the DTC enter module stop mode. Consequently, on-chip peripheral module registers cannot be read or written to. Register reading and writing is enabled when the module stop mode is cancelled.

## 4.4 Traces

Traces are enabled in interrupt control mode 2. Trace mode is not activated in interrupt control mode 0, irrespective of the state of the T bit. For details of interrupt control modes, see section 5, Interrupt Controller.

If the T bit in EXR is set to 1, trace mode is activated. In trace mode, a trace exception occurs on completion of each instruction. Trace mode is not affected by interrupt mask bit in CCR. Table 4.3 shows the states of CCR and EXR after execution of trace exception handling. Trace mode is cancelled by clearing the T bit in EXR to 0 with the trace exception handling. The T bit saved on the stack retains its value of 1, and when control is returned from the trace exception handling routine by the RTE instruction, trace mode resumes. Trace exception handling is not carried out after execution of the RTE instruction.

Interrupts are accepted even within the trace exception handling routine.

**Table 4.3 Statuses of CCR and EXR after Trace Exception Handling**

| | CCR | | EXR | |
|---|---|---|---|---|
| **Interrupt Control Mode** | **I** | **UI** | **I2 to I0** | **T** |
| 0 | Trace exception handling cannot be used. | | | |
| 2 | 1 | — | — | 0 |

**Legend**

1: Set to 1

0: Cleared to 0

—: Retains value prior to execution

## 4.5 Interrupts

Interrupts are controlled by the interrupt controller. The interrupt controller has two interrupt control modes and can assign interrupts other than NMI to eight priority/mask levels to enable multiplexed interrupt control. The source to start interrupt exception handling and the vector address differ depending on the product. For details, refer to section 5, Interrupt Controller.

Interrupt exception handling is conducted as follows:

1. The values in the program counter (PC), condition code register (CCR), and extended control register (EXR) are saved to the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.
3. A vector address corresponding to the interrupt source is generated, the start address is loaded from the vector table to the PC, and program execution begins from that address.

## 4.6 Trap Instruction

Trap instruction exception handling starts when a TRAPA instruction is executed. Trap instruction exception handling can be executed at all times in the program execution state.

Trap instruction exception handling is conducted as follows:

1. The values in the program counter (PC), condition code register (CCR), and extended control register (EXR) are saved to the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.
3. A vector address corresponding to the interrupt source is generated, the start address is loaded from the vector table to the PC, and program execution starts from that address.

The TRAPA instruction fetches a start address from a vector table entry corresponding to a vector number from 0 to 3, as specified in the instruction code.

Table 4.4 shows the statuses of CCR and EXR after execution of trap instruction exception handling.

**Table 4.4 Statuses of CCR and EXR after Trap Instruction Exception Handling**

| | CCR | | EXR | |
|---|---|---|---|---|
| **Interrupt Control Mode** | **I** | **UI** | **I2 to I0** | **T** |
| 0 | 1 | — | — | — |
| 2 | 1 | — | — | 0 |

**Legend**

1: Set to 1

0: Cleared to 0

—: Retains value prior to execution

## 4.7 Stack Status after Exception Handling

Figures 4.3 shows the stack after completion of trap instruction exception handling and interrupt exception handling.

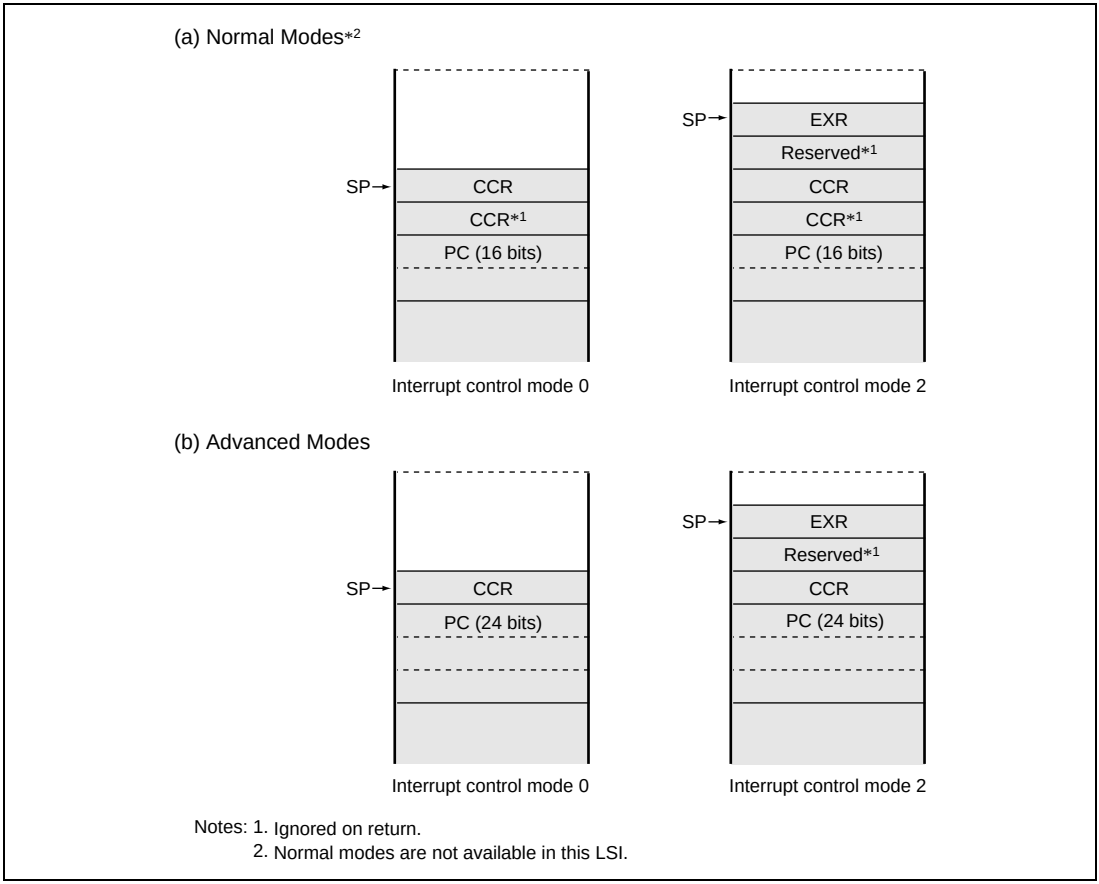



**Figure 4.3 Stack Status after Exception Handling**

## 4.8 Usage Note

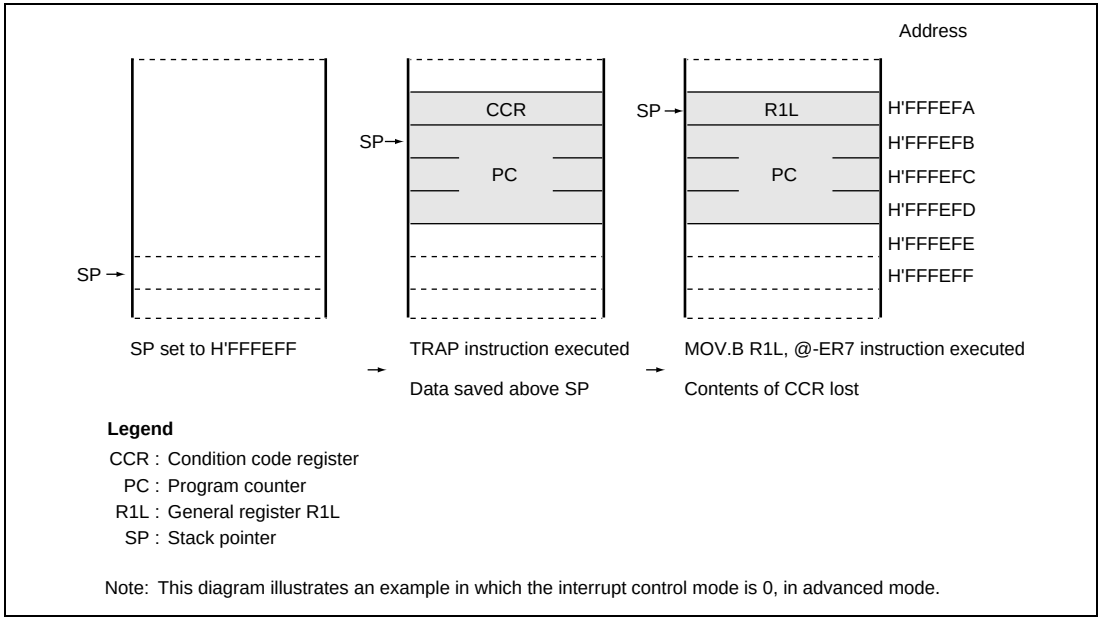
When accessing word data or longword data, this LSI assumes that the lowest address bit is 0. The stack should always be accessed by word transfer instruction or longword transfer instruction, and the value of the stack pointer (SP: ER7) should always be kept even. Use the following instructions to save registers:

```
PUSH.W   Rn    (or MOV.W Rn, @-SP)
PUSH.L   ERn   (or MOV.L ERn, @-SP)
```

Use the following instructions to restore registers:

```
POP.W    Rn    (or MOV.W @SP+, Rn)
POP.L    ERn   (or MOV.L @SP+, ERn)
```

Setting SP to an odd value may lead to a malfunction. Figure 4.4 shows an example of what happens when the SP value is odd.




Note: This diagram illustrates an example in which the interrupt control mode is 0, in advanced mode.

**Figure 4.4 Operation when SP Value Is Odd**

# Section 5 Interrupt Controller

## 5.1 Features

- Two interrupt control modes
  - Any of two interrupt control modes can be set by means of the INTM1 and INTM0 bits in the system control register (SYSCR).
- Priorities settable with IPR
  - An interrupt priority register (IPR) is provided for setting interrupt priorities. Eight priority levels can be set for each module for all interrupts except NMI. NMI is assigned the highest priority level of 8, and can be accepted at all times.
- Independent vector addresses
  - All interrupt sources are assigned independent vector addresses, making it unnecessary for the source to be identified in the interrupt handling routine.
- Seven external interrupts
  - NMI is the highest-priority interrupt, and is accepted at all times. Rising edge or falling edge can be selected for NMI. Falling edge, rising edge, or both edge detection, or level sensing, can be selected for IRQ0 to IRQ5.
- DTC control
  - The DTC can be activated by an interrupt request.

A block diagram of the interrupt controller is shown in figure 5.1.

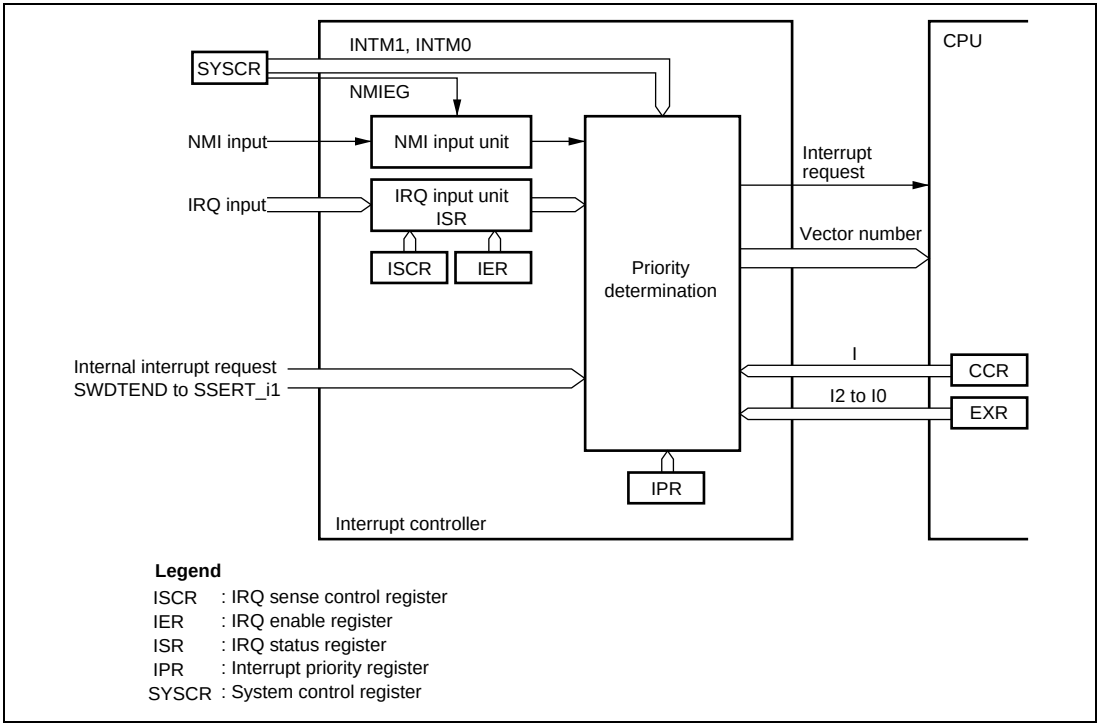



**Figure 5.1 Block Diagram of Interrupt Controller**

## 5.2 Input/Output Pins

Table 5.1 summarizes the pins of the interrupt controller.

**Table 5.1 Pin Configuration**

| Name | I/O | Function |
|---|---|---|
| NMI | Input | Nonmaskable external interrupt<br>Rising or falling edge can be selected |
| $\overline{\text{IRQ5}}$ | Input | Maskable external interrupts |
| $\overline{\text{IRQ4}}$ | Input | Rising, falling, or both edges, or level sensing, can be selected |
| $\overline{\text{IRQ3}}$ | Input | |
| $\overline{\text{IRQ2}}$ | Input | |
| $\overline{\text{IRQ1}}$ | Input | |
| $\overline{\text{IRQ0}}$ | Input | |

## 5.3 Register Descriptions

The interrupt controller has the following registers. For the system control register (SYSCR), refer to section 3.2.2, System Control Register (SYSCR).

- System control register (SYSCR)
- IRQ sense control register H (ISCRH)
- IRQ sense control register L (ISCRL)
- IRQ enable register (IER)
- IRQ status register (ISR)
- Interrupt priority register A (IPRA)
- Interrupt priority register B (IPRB)
- Interrupt priority register C (IPRC)
- Interrupt priority register D (IPRD)
- Interrupt priority register E (IPRE)
- Interrupt priority register F (IPRF)
- Interrupt priority register G (IPRG)
- Interrupt priority register H (IPRH)
- Interrupt priority register I (IPRI)
- Interrupt priority register J (IPRJ)
- Interrupt priority register K (IPRK)
- Interrupt priority register L (IPRL)
- Interrupt priority register M (IPRM)

### 5.3.1 Interrupt Priority Registers A to M (IPRA to IPRM)

The IPR registers are thirteen 8-bit readable/writable registers that set priorities (levels 7 to 0) for interrupts other than NMI.

The correspondence between interrupt sources and IPR settings is shown in table 5.2. Setting a value in the range from H′0 to H′7 in the 3-bit groups of bits 0 to 2 and 4 to 6 sets the priority of the corresponding interrupt.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved<br>These bits are always read as 0. |
| 6<br>5<br>4 | IPR6<br>IPR5<br>IPR4 | 1<br>1<br>1 | R/W<br>R/W<br>R/W | Sets the priority of the corresponding interrupt source.<br>000: Priority level 0 (Lowest)<br>001: Priority level 1<br>010: Priority level 2<br>011: Priority level 3<br>100: Priority level 4<br>101: Priority level 5<br>110: Priority level 6<br>111: Priority level 7 (Highest) |
| 3 | — | 0 | — | Reserved<br>These bits are always read as 0. |
| 2<br>1<br>0 | IPR2<br>IPR1<br>IPR0 | 1<br>1<br>1 | R/W<br>R/W<br>R/W | Sets the priority of the corresponding interrupt source.<br>000: Priority level 0 (Lowest)<br>001: Priority level 1<br>010: Priority level 2<br>011: Priority level 3<br>100: Priority level 4<br>101: Priority level 5<br>110: Priority level 6<br>111: Priority level 7 (Highest) |

### 5.3.2 IRQ Enable Register (IER)

IER is an 8-bit readable/writable register that controls the enabling and disabling of interrupt requests IRQ0 to IRQ5.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | R/W | Reserved<br>Only 0 should be written to these bits. |
| 5 | IRQ5E | 0 | R/W | IRQ5 Enable<br>The IRQ5 interrupt request is enabled when this bit is 1. |
| 4 | IRQ4E | 0 | R/W | IRQ4 Enable<br>The IRQ4 interrupt request is enabled when this bit is 1. |
| 3 | IRQ3E | 0 | R/W | IRQ3 Enable<br>The IRQ3 interrupt request is enabled when this bit is 1. |
| 2 | IRQ2E | 0 | R/W | IRQ2 Enable<br>The IRQ2 interrupt request is enabled when this bit is 1. |
| 1 | IRQ1E | 0 | R/W | IRQ1 Enable<br>The IRQ1 interrupt request is enabled when this bit is 1. |
| 0 | IRQ0E | 0 | R/W | IRQ0 Enable<br>The IRQ0 interrupt request is enabled when this bit is 1. |

### 5.3.3 IRQ Sense Control Registers H and L (ISCRH, ISCRL)

The ISCR registers are 16-bit readable/writable registers that select the source that generates an interrupt request at pins $\overline{IRQ0}$ to $\overline{IRQ5}$.

- ISCRH

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 12 | — | All 0 | R/W | Reserved<br>Only 0 should be written to these bits. |
| 11 | IRQ5SCB | 0 | R/W | IRQ5 Sense Control B |
| 10 | IRQ5SCA | 0 | R/W | IRQ5 Sense Control A<br>00: Interrupt request generated at $\overline{IRQ5}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{IRQ5}$ input<br>10: Interrupt request generated at rising edge of $\overline{IRQ5}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{IRQ5}$ input |
| 9 | IRQ4SCB | 0 | R/W | IRQ4 Sense Control B |
| 8 | IRQ4SCA | 0 | R/W | IRQ4 Sense Control A<br>00: Interrupt request generated at $\overline{IRQ4}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{IRQ4}$ input<br>10: Interrupt request generated at rising edge of $\overline{IRQ4}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{IRQ4}$ input |

- ISCRL

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | IRQ3SCB | 0 | R/W | IRQ3 Sense Control B |
| 6 | IRQ3SCA | 0 | R/W | IRQ3 Sense Control A<br>00: Interrupt request generated at $\overline{\text{IRQ3}}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{\text{IRQ3}}$ input<br>10: Interrupt request generated at rising edge of $\overline{\text{IRQ3}}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ3}}$ input |
| 5 | IRQ2SCB | 0 | R/W | IRQ2 Sense Control B |
| 4 | IRQ2SCA | 0 | R/W | IRQ2 Sense Control A<br>00: Interrupt request generated at $\overline{\text{IRQ2}}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{\text{IRQ2}}$ input<br>10: Interrupt request generated at rising edge of $\overline{\text{IRQ2}}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ2}}$ input |
| 3 | IRQ1SCB | 0 | R/W | IRQ1 Sense Control B |
| 2 | IRQ1SCA | 0 | R/W | IRQ1 Sense Control A<br>00: Interrupt request generated at $\overline{\text{IRQ1}}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{\text{IRQ1}}$ input<br>10: Interrupt request generated at rising edge of $\overline{\text{IRQ1}}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ1}}$ input |
| 1 | IRQ0SCB | 0 | R/W | IRQ0 Sense Control B |
| 0 | IRQ0SCA | 0 | R/W | IRQ0 Sense Control A<br>00: Interrupt request generated at $\overline{\text{IRQ0}}$ input level low<br>01: Interrupt request generated at falling edge of $\overline{\text{IRQ0}}$ input<br>10: Interrupt request generated at rising edge of $\overline{\text{IRQ0}}$ input<br>11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ0}}$ input |

### 5.3.4 IRQ Status Register (ISR)

ISR is an 8-bit readable/writable register that indicates the status of IRQ0 to IRQ5 interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | R/W | Reserved<br>Only 0 should be written to these bits. |
| 5 | IRQ5F | 0 | R/W | [Setting conditions]<br>When the interrupt source selected by the ISCR registers occurs<br>[Clearing conditions]<br>• Cleared by reading IRQnF flag when IRQnF = 1, then writing 0 to IRQnF flag<br>• When interrupt exception handling is executed when low-level detection is set and $\overline{\text{IRQn}}$ input is high<br>• When IRQn interrupt exception handling is executed when falling, rising, or both-edge detection is set<br>• When the DTC is activated by an IRQn interrupt, and the DISEL bit in MRB of the DTC is cleared to 0 |
| 4 | IRQ4F | 0 | R/W | |
| 3 | IRQ3F | 0 | R/W | |
| 2 | IRQ2F | 0 | R/W | |
| 1 | IRQ1F | 0 | R/W | |
| 0 | IRQ0F | 0 | R/W | |

## 5.4 Interrupt Sources

### 5.4.1 External Interrupts

There are seven external interrupts: NMI and IRQ0 to IRQ5. These interrupts can be used to restore this LSI from software standby mode.

**NMI Interrupt:** NMI is the highest-priority interrupt, and is always accepted by the CPU regardless of the interrupt control mode or the status of the CPU interrupt mask bits. The NMIEG bit in SYSCR can be used to select whether an interrupt is requested at a rising edge or a falling edge on the NMI pin.

**IRQ0 to IRQ5 Interrupts:** Interrupts IRQ0 to IRQ5 are requested by an input signal at pins $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ5}}$. Interrupts IRQ0 to IRQ5 have the following features:

- Using ISCR, it is possible to select whether an interrupt is generated by a low level, falling edge, rising edge, or both edges, at pins $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ5}}$.
- Enabling or disabling of interrupt requests IRQ0 to IRQ5 can be selected with IER.
- The interrupt priority level can be set with IPR.
- The status of interrupt requests IRQ0 to IRQ5 is indicated in ISR. ISR flags can be cleared to 0 by software.

The detection of IRQ0 to IRQ5 interrupts does not depend on whether the relevant pin has been set for input or output. However, when a pin is used as an external interrupt input pin, do not clear the corresponding DDR to 0; and use the pin as an I/O pin for another function.

A block diagram of interrupts IRQ0 to IRQ5 is shown in figure 5.2.

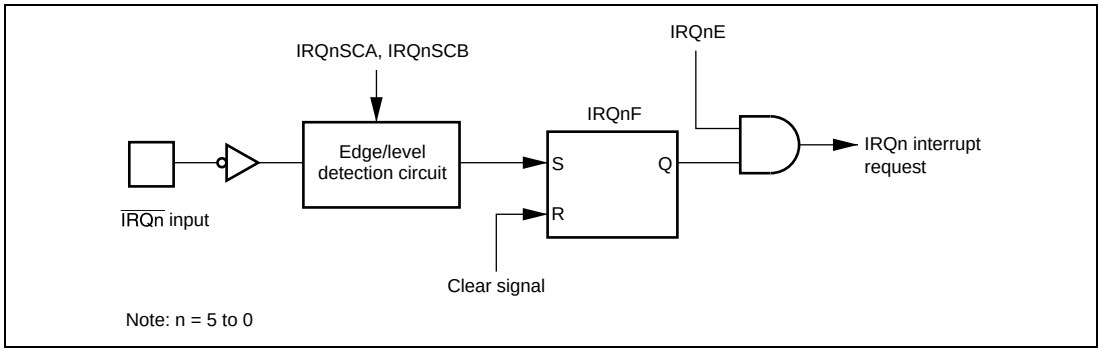



**Figure 5.2 Block Diagram of Interrupts IRQ0 to IRQ5**

### 5.4.2 Internal Interrupts

The sources for internal interrupts from on-chip peripheral modules have the following features:

- For each on-chip peripheral module there are flags that indicate the interrupt request status, and enable bits that select enabling or disabling of these interrupts. If both of these are set to 1 for a particular interrupt source, an interrupt request is issued to the interrupt controller.
- The interrupt priority level can be set by means of IPR.
- The DTC can be activated by a TPU, SCI, or other interrupt request.
- When the DTC is activated by an interrupt request, it is not affected by the interrupt control mode or CPU interrupt mask bit.

## 5.5 Interrupt Exception Handling Vector Table

Table 5.2 shows interrupt exception handling sources, vector addresses, and interrupt priorities.

For default priorities, the lower the vector number, the higher the priority. Priorities among modules can be set by means of IPR. Modules set at the same priority will conform to their default priorities. Priorities within a module are fixed.

**Table 5.2 Interrupt Sources, Vector Addresses, and Interrupt Priorities**

| Interrupt Source | Origin of Interrupt Source | Vector Number | Vector Address* Advanced Mode | IPR | Priority |
|---|---|---|---|---|---|
| External pin | NMI | 7 | H′001C | | High |
| | IRQ0 | 16 | H′0040 | IPRA6 to IPRA4 | |
| | IRQ1 | 17 | H′0044 | IPRA2 to IPRA0 | |
| | IRQ2 | 18 | H′0048 | IPRB6 to IPRB4 | |
| | IRQ3 | 19 | H′004C | | |
| | IRQ4 | 20 | H′0050 | IPRB2 to IPRB0 | |
| | IRQ5 | 21 | H′0054 | | |
| — | Reserved for system use | 22 | H′0058 | | |
| | Reserved for system use | 23 | H′005C | | |
| DTC | SWDTEND | 24 | H′0060 | IPRC2 to IPRC0 | |
| Watchdog timer 0 | WOVI0 | 25 | H′0064 | IPRD6 to IPRD4 | |
| PC break control | PC break | 27 | H′006C | IPRE6 to IPRE4 | |
| A/D | ADI | 28 | H′0070 | IPRE2 to IPRE0 | |
| TPU channel 0 | TGIA_0 | 32 | H′0080 | IPRF6 to IPRF4 | |
| | TGIB_0 | 33 | H′0084 | | |
| | TGIC_0 | 34 | H′0088 | | |
| | TGID_0 | 35 | H′008C | | |
| | TCIV_0 | 36 | H′0090 | | |
| TPU channel 1 | TGIA_1 | 40 | H′00A0 | IPRF2 to IPRF0 | |
| | TGIB_1 | 41 | H′00A4 | | |
| | TCIV_1 | 42 | H′00A8 | | |
| | TCIU_1 | 43 | H′00AC | | |
| TPU channel 2 | TGIA_2 | 44 | H′00B0 | IPRG6 to IPRG4 | |
| | TGIB_2 | 45 | H′00B4 | | |
| | TCIV_2 | 46 | H′00B8 | | |
| | TCIU_2 | 47 | H′00BC | | Low |

| Interrupt Source | Origin of Interrupt Source | Vector Number | Vector Address* Advanced Mode | IPR | Priority |
|---|---|---|---|---|---|
| TPU | TGIA_3 | 48 | H′00C0 | IPRG2 to IPRG0 | High |
| channel 3 | TGIB_3 | 49 | H′00C4 | | |
| | TGIC_3 | 50 | H′00C8 | | |
| | TGID_3 | 51 | H′00CC | | |
| | TCIV_3 | 52 | H′00D0 | | |
| TPU | TGIA_4 | 56 | H′00E0 | IPRH6 to IPRH4 | |
| channel 4 | TGIB_4 | 57 | H′00E4 | | |
| | TCIV_4 | 58 | H′00E8 | | |
| | TCIU_4 | 59 | H′00EC | | |
| TPU | TGIA_5 | 60 | H′00F0 | IPRH2 to IPRH0 | |
| channel 5 | TGIB_5 | 61 | H′00F4 | | |
| | TCIV_5 | 62 | H′00F8 | | |
| | TCIU_5 | 63 | H′00FC | | |
| 8-bit timer channel 0 | CMIA_0 | 64 | H′0100 | IPRI6 to IPRI4 | |
| | CMIB_0 | 65 | H′0104 | | |
| | OVI_0 | 66 | H′0108 | | |
| 8-bit timer channel 1 | CMIA_1 | 68 | H′0110 | IPRI2 to IPRI0 | |
| | CMIB_1 | 69 | H′0114 | | |
| | OVI_1 | 70 | H′0118 | | |
| SCI | ERI_0 | 80 | H′0140 | IPRJ2 to IPRJ0 | |
| channel 0 | RXI_0 | 81 | H′0144 | | |
| | TXI_0 | 82 | H′0148 | | |
| | TEI_0 | 83 | H′014C | | |
| SCI | ERI_2 | 88 | H′0160 | IPRK2 to IPRK0 | |
| channel 2 | RXI_2 | 89 | H′0164 | | |
| | TXI_2 | 90 | H′0168 | | |
| | TEI_2 | 91 | H′016C | | Low |

| Interrupt Source | Origin of Interrupt Source | Vector Number | Vector Address* Advanced Mode | IPR | Priority |
|---|---|---|---|---|---|
| 8-bit timer channel 2 | CMIA_2 | 92 | H′0170 | IPRL6 to IPRL4 | High |
| | CMIB_2 | 93 | H′0174 | | |
| | OVI_2 | 94 | H′0178 | | |
| 8-bit timer channel 3 | CMIA_3 | 96 | H′0180 | | |
| | CMIB_3 | 97 | H′0184 | | |
| | OVI_3 | 98 | H′0188 | | |
| HCAN | ERS0, OVR0 | 104 | H′01A0 | IPRM6 to IPRM4 | |
| | RM0 | 105 | H′01A4 | | |
| | RM1 | 106 | H′01A8 | | |
| | SLE0 | 107 | H′01AC | | |
| SSU channel 0 | SSEr_i0 | 108 | H′01B0 | IPRM2 to IPRM0 | |
| | SSRx_i0 | 109 | H′01B4 | | |
| | SSTx_i0 | 110 | H′01B8 | | |
| SSU channel 1 | SSERT_i1 | 111 | H′01BC | | Low |

Note: * Lower 16 bits of the start address.

## 5.6 Interrupt Control Modes and Interrupt Operation

The interrupt controller has two modes: interrupt control mode 0 and interrupt control mode 2. Interrupt operations differ depending on the interrupt control mode. The interrupt control mode is selected by SYSCR. Table 5.3 shows the differences between interrupt control mode 0 and interrupt control mode 2.

**Table 5.3 Interrupt Control Modes**

| Interrupt Control Mode | Priority Setting Registers | Interrupt Mask Bits | Description |
|---|---|---|---|
| 0 | Default | I | The priorities of interrupt sources are fixed at the default settings.<br>Interrupt sources, except for NMI, are masked by the I bit. |
| 2 | IPR | I2 to I0 | 8 priority levels other than NMI can be set with IPR.<br>8-level interrupt mask control is performed by bits I2 to I0. |

### 5.6.1 Interrupt Control Mode 0

In interrupt control mode 0, interrupt requests other than for NMI are masked by the I bit in CCR in the CPU. Figure 5.3 shows a flowchart of the interrupt acceptance operation in this case.

1. If an interrupt source occurs when the corresponding interrupt enable bit is set to 1, an interrupt request is sent to the interrupt controller.
2. If the I bit in CCR is set to 1, only an NMI interrupt is accepted, and other interrupt requests are held pending. If the I bit is cleared, an interrupt request is accepted.
3. When interrupt requests are sent to the interrupt controller, the interrupt with the highest priority according to the interrupt priority levels is selected and other interrupt requests are held pending.
4. When the CPU accepts an interrupt request, it starts interrupt exception handling after execution of the current instruction has been completed.
5. The PC and CCR are saved to the stack area by interrupt exception handling. The PC saved on the stack shows the address of the first instruction to be executed after returning from the interrupt handling routine.
6. Next, the I bit in CCR is set to 1. This masks all interrupts except NMI.
7. The CPU generates a vector address for the accepted interrupt and starts execution of the interrupt handling routine at the address indicated by the contents of the vector address in the vector table.

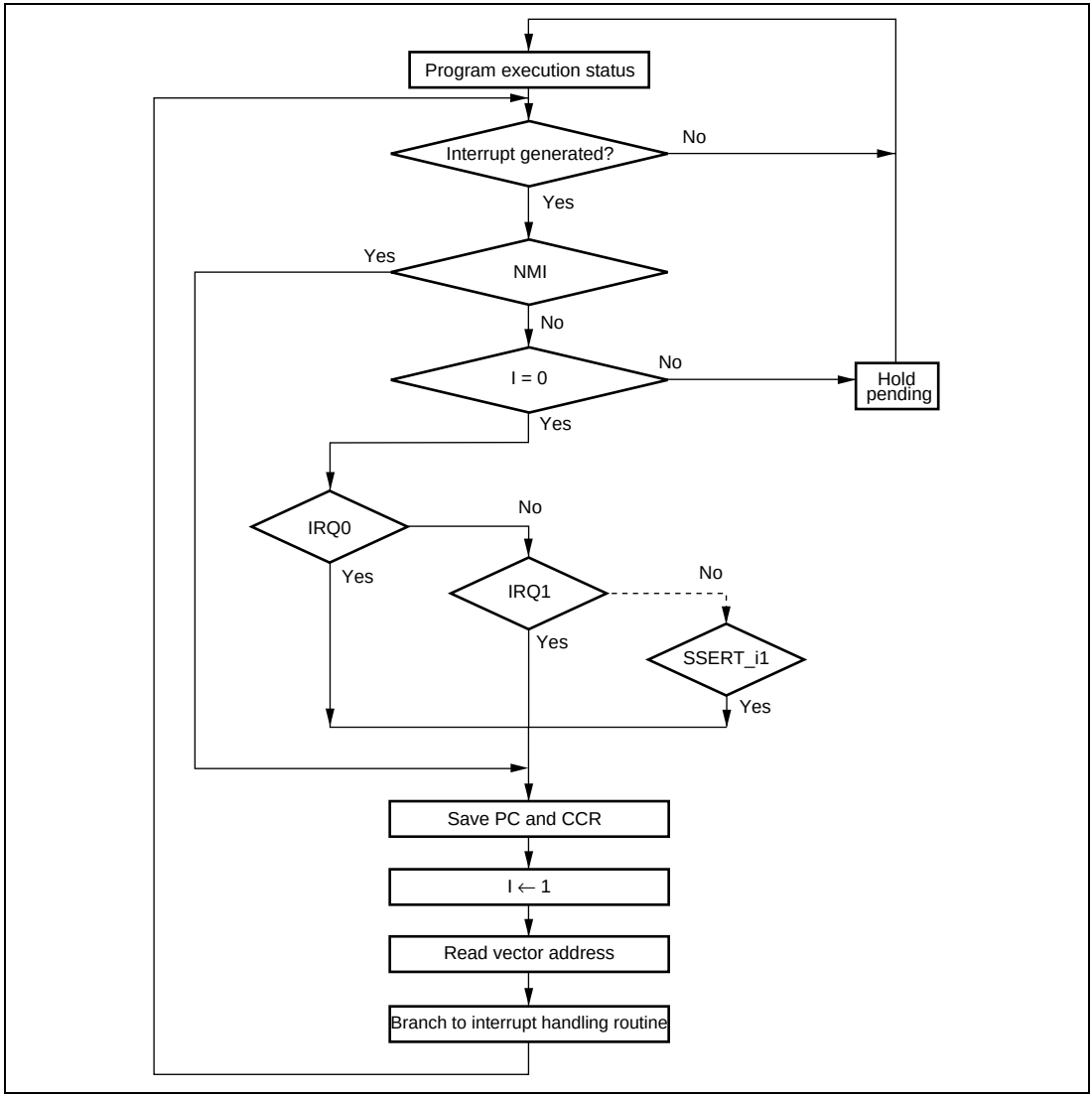



**Figure 5.3 Flowchart of Procedure up to Interrupt Acceptance in Interrupt Control Mode 0**

### 5.6.2 Interrupt Control Mode 2

In interrupt control mode 2, mask control is applied to eight levels for interrupt requests other than NMI by comparing the EXR interrupt mask level (I2 to I0 bits) in the CPU and the IPR setting. Figure 5.4 shows a flowchart of the interrupt acceptance operation in this case.

1. If an interrupt source occurs when the corresponding interrupt enable bit is set to 1, an interrupt request is sent to the interrupt controller.
2. When interrupt requests are sent to the interrupt controller, the interrupt with the highest priority according to the interrupt priority levels set in IPR is selected, and lower-priority interrupt requests are held pending. If a number of interrupt requests with the same priority are generated at the same time, the interrupt request with the highest priority according to the priority system shown in table 5.2 is selected.
3. Next, the priority of the selected interrupt request is compared with the interrupt mask level set in EXR. An interrupt request with a priority no higher than the mask level set at that time is held pending, and only an interrupt request with a priority higher than the interrupt mask level is accepted.
4. When the CPU accepts an interrupt request, it starts interrupt exception handling after execution of the current instruction has been completed.
5. The PC, CCR, and EXR are saved to the stack area by interrupt exception handling. The PC saved on the stack shows the address of the first instruction to be executed after returning from the interrupt handling routine.
6. The T bit in EXR is cleared to 0. The interrupt mask level is rewritten with the priority level of the accepted interrupt. If the accepted interrupt is NMI, the interrupt mask level is set to H′7.
7. The CPU generates a vector address for the accepted interrupt and starts execution of the interrupt handling routine at the address indicated by the contents of the vector address in the vector table.

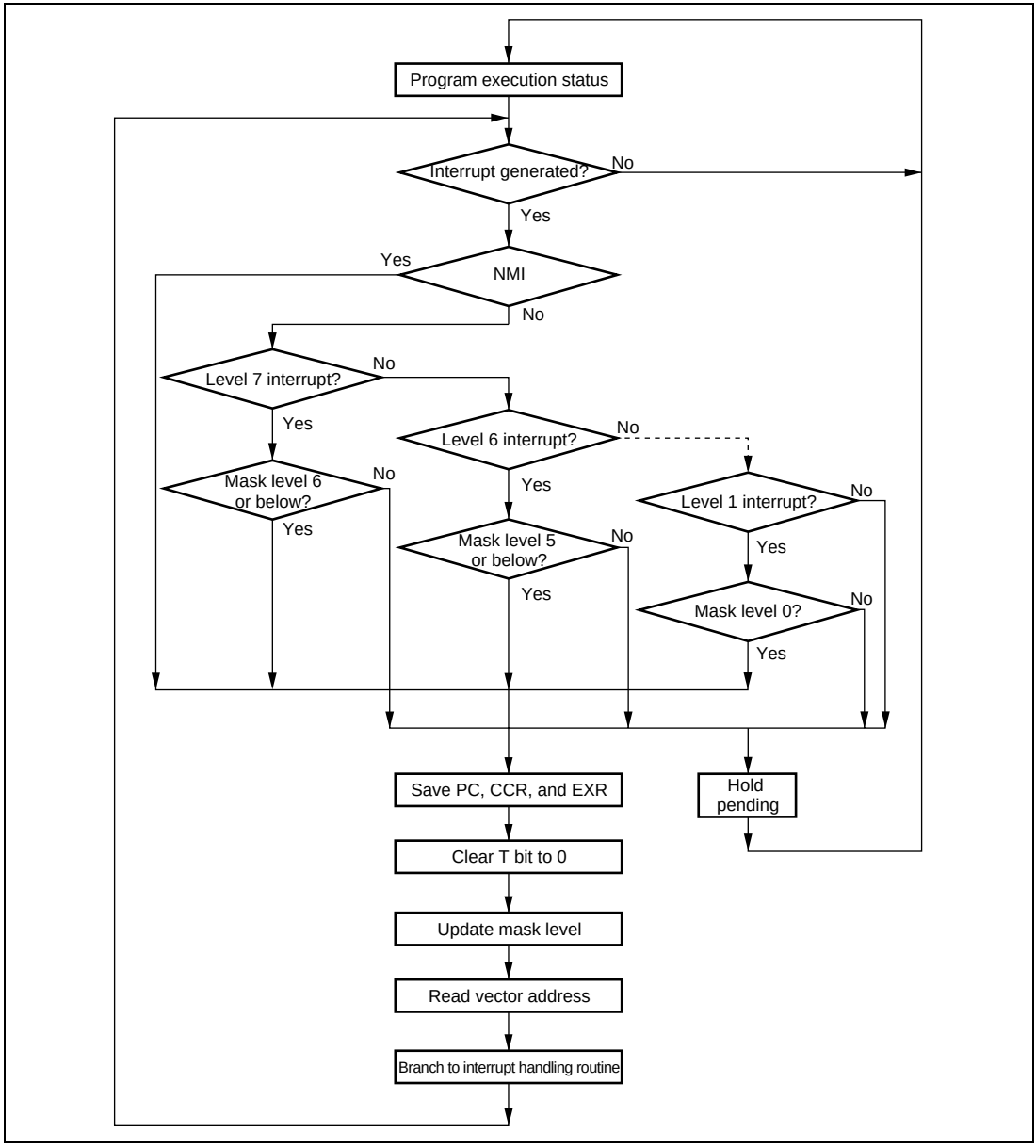



**Figure 5.4 Flowchart of Procedure Up to Interrupt Acceptance in Control Mode 2**

### 5.6.3 Interrupt Exception Handling Sequence

Figure 5.5 shows the interrupt exception handling sequence. The example shown is for the case where interrupt control mode 0 is set in advanced mode, and the program area and stack area are in on-chip memory.

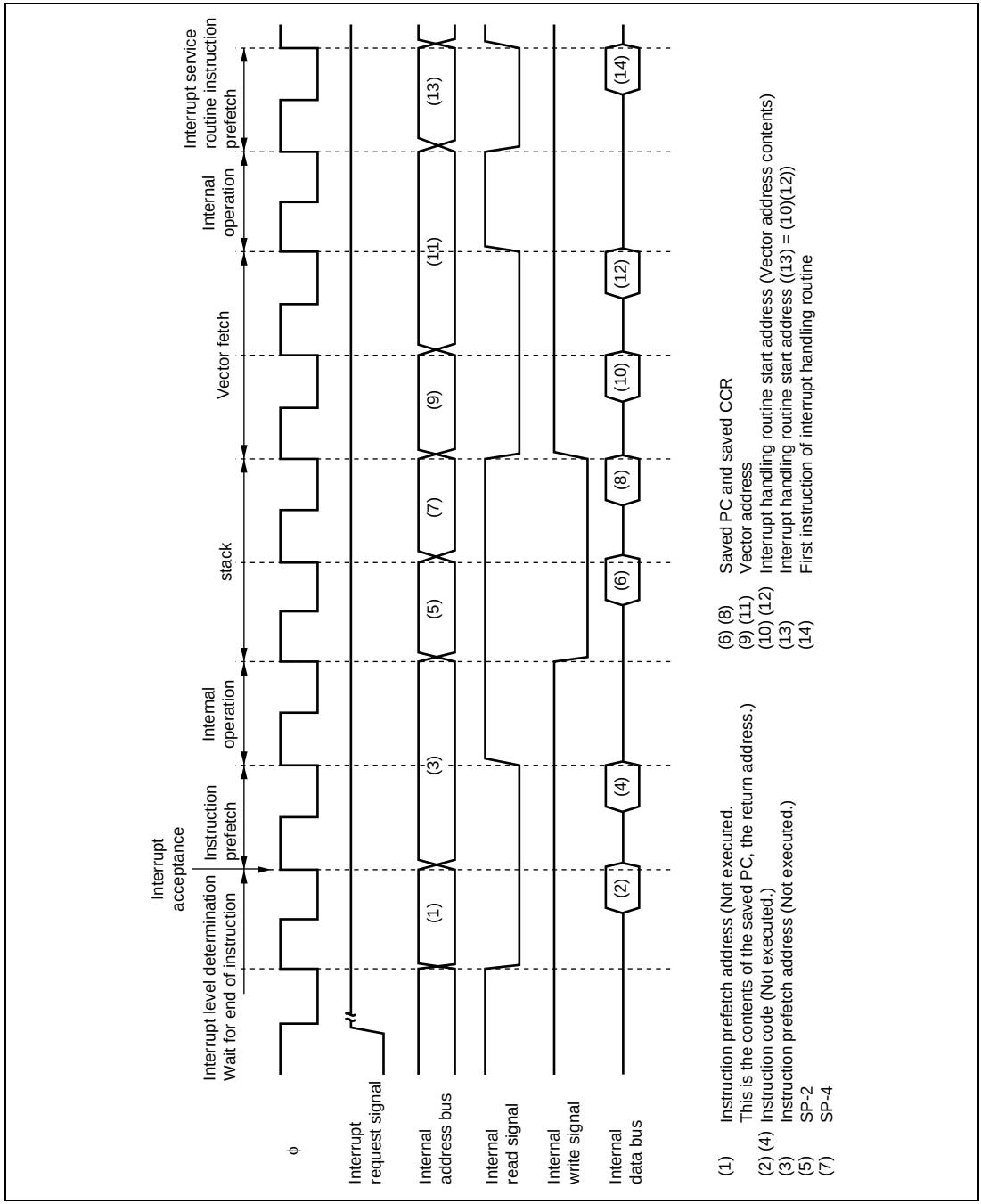



**Figure 5.5 Interrupt Exception Handling**

### 5.6.4 Interrupt Response Times

Table 5.4 shows interrupt response times - the interval between generation of an interrupt request and execution of the first instruction in the interrupt handling routine. The execution status symbols used in table 5.4 are explained in table 5.5.

This LSI is capable of fast word transfer to on-chip memory, has the program area in on-chip ROM and the stack area in on-chip RAM, enabling high-speed processing.

**Table 5.4 Interrupt Response Times**

| | | Normal Mode*[5] | | Advanced Mode | |
|---|---|---|---|---|---|
| No. | Execution Status | Interrupt control mode 0 | Interrupt control mode 2 | Interrupt control mode 0 | Interrupt control mode 2 |
| 1 | Interrupt priority determination*[1] | 3 | 3 | 3 | 3 |
| 2 | Number of wait states until executing instruction ends*[2] | 1 to 19 +2·$S_I$ | 1 to 19+2·$S_I$ | 1 to 19+2·$S_I$ | 1 to 19+2·$S_I$ |
| 3 | PC, CCR, EXR stack save | 2·$S_K$ | 3·$S_K$ | 2·$S_K$ | 3·$S_K$ |
| 4 | Vector fetch | $S_I$ | $S_I$ | 2·$S_I$ | 2·$S_I$ |
| 5 | Instruction fetch*[3] | 2·$S_I$ | 2·$S_I$ | 2·$S_I$ | 2·$S_I$ |
| 6 | Internal processing*[4] | 2 | 2 | 2 | 2 |
| Total (using on-chip memory) | | 11 to 31 | 12 to 32 | 12 to 32 | 13 to 33 |

Notes: 1. Two states in case of internal interrupt.
2. Refers to MULXS and DIVXS instructions.
3. Prefetch after interrupt acceptance and interrupt handling routine prefetch.
4. Internal processing after interrupt acceptance and internal processing after vector fetch.
5. Not available in this LSI.

**Table 5.5 Number of States in Interrupt Handling Routine Execution Status**

| | | Object of Access | | | | |
|---|---|---|---|---|---|---|
| | | | External Device * | | | |
| | | | 8-Bit Bus | | 16-Bit Bus | |
| Symbol | | Internal Memory | 2-State Access | 3-State Access | 2-State Access | 3-State Access |
| Instruction fetch | $S_I$ | 1 | 4 | 6+2m | 2 | 3+m |
| Branch address read | $S_J$ | | | | | |
| Stack manipulation | $S_K$ | | | | | |

**Legend**

m : Number of wait states in an external device access.

Note:* Not available in this LSI.

### 5.6.5 DTC Activation by Interrupt

The DTC can be activated by an interrupt. For details, see section 8, Data Transfer Controller (DTC).

## 5.7 Usage Notes

### 5.7.1 Conflict between Interrupt Generation and Disabling

When an interrupt enable bit is cleared to 0 to disable interrupts, the disabling becomes effective after execution of the instruction.

When an interrupt enable bit is cleared to 0 by an instruction such as BCLR or MOV, and if an interrupt is generated during execution of the instruction, the interrupt concerned will still be enabled on completion of the instruction, and so interrupt exception handling for that interrupt will be executed on completion of the instruction. However, if there is an interrupt request of higher priority than that interrupt, interrupt exception handling will be executed for the higher-priority interrupt, and the lower-priority interrupt will be ignored.

The same also applies when an interrupt source flag is cleared to 0.

Figure 5.6 shows an example in which the TCIEV bit in TIER_0 of the TPU is cleared to 0.

The above conflict will not occur if an enable bit or interrupt source flag is cleared to 0 while the interrupt is masked.

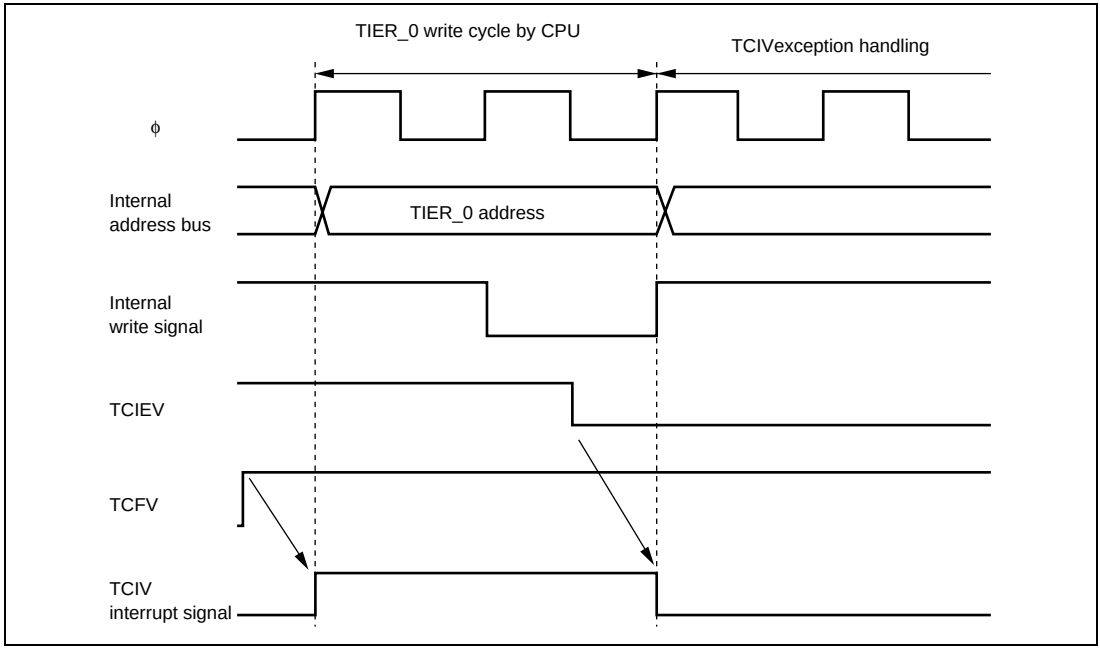



**Figure 5.6 Conflict between Interrupt Generation and Disabling**

### 5.7.2 Instructions that Disable Interrupts

The instructions that disable interrupts are LDC, ANDC, ORC, and XORC. After any of these instructions are executed, all interrupts including NMI are disabled and the next instruction is always executed. When the I bit is set by one of these instructions, the new value becomes valid two states after execution of the instruction ends.

### 5.7.3 When Interrupts Are Disabled

There are times when interrupt acceptance is disabled by the interrupt controller.

The interrupt controller disables interrupt acceptance for a 3-state period after the CPU has updated the mask level with an LDC, ANDC, ORC, or XORC instruction.

### 5.7.4 Interrupts during Execution of EEPMOV Instruction

Interrupt operation differs between the EEPMOV.B instruction and the EEPMOV.W instruction.

With the EEPMOV.B instruction, an interrupt request (including NMI) issued during the transfer is not accepted until the transfer is completed.

With the EEPMOV.W instruction, if an interrupt request is issued during the transfer, interrupt exception handling starts at a break in the transfer cycle. The PC value saved on the stack in this case is the address of the next instruction.

Therefore, if an interrupt is generated during execution of an EEPMOV.W instruction, the following coding should be used.

```
L1:   EEPMOV.W
      MOV.W     R4,R4
      BNE       L1
```

# Section 6 PC Break Controller (PBC)

The PC break controller (PBC) provides functions that simplify program debugging. Using these functions, it is easy to create a self-monitoring debugger, enabling programs to be debugged with the chip alone, without using an in-circuit emulator. A block diagram of the PC break controller is shown in figure 6.1.

## 6.1 Features

- Two break channels (A and B)
- 24-bit break address
  - Bit masking possible
- Four types of break compare conditions
  - Instruction fetch
  - data read
  - data write
  - data read/write
- Bus master
  - Either CPU or CPU/DTC can be selected
- The timing of PC break exception handling after the occurrence of a break condition is as follows:
  - Immediately before execution of the instruction fetched at the set address (instruction fetch)
  - Immediately after execution of the instruction that accesses data at the set address (data access)
- Module stop mode can be set

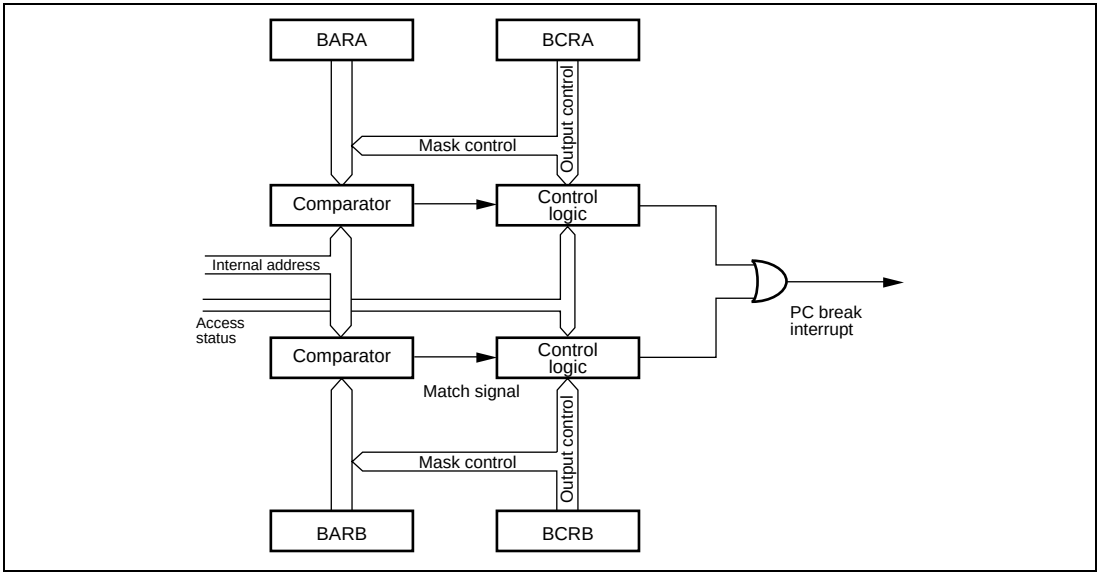



**Figure 6.1 Block Diagram of PC Break Controller**

## 6.2 Register Descriptions

The PC break controller has the following registers.

- Break address register A (BARA)
- Break address register B (BARB)
- Break control register A (BCRA)
- Break control register B (BCRB)

### 6.2.1 Break Address Register A (BARA)

BARA is a 32-bit readable/writable register that specifies the channel A break address.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | Undefined | — | Reserved<br>These bits are read as an undefined value and cannot be modified. |
| 23 to 0 | BAA23 to BAA0 | H′000000 | R/W | These bits set the channel A PC break address. |

### 6.2.2 Break Address Register B (BARB)

BARB is the channel B break address register. The bit configuration is the same as for BARA.

### 6.2.3 Break Control Register A (BCRA)

BCRA controls channel A PC breaks. BCRA also contains a condition match flag.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMFA | 0 | R/W | Condition Match Flag A<br>[Setting condition]<br>When a condition set for channel A is satisfied<br>[Clearing condition]<br>When 0 is written to CMFA after reading CMFA = 1 |
| 6 | CDA | 0 | R/W | CPU Cycle/DTC Cycle Select A<br>Selects the channel A break condition bus master.<br>0: CPU<br>1: CPU or DTC |
| 5<br>4<br>3 | BAMRA2<br>BAMRA1<br>BAMRA0 | 0<br>0<br>0 | R/W<br>R/W<br>R/W | Break Address Mask Register A2 to A0<br>These bits specify which bits of the break address set in BARA are to be masked.<br>000: BAA23 to BAA0 (All bits are unmasked)<br>001: BAA23 to BAA1 (Lowest bit is masked)<br>010: BAA23 to BAA2 (Lower 2 bits are masked)<br>011: BAA23 to BAA3 (Lower 3 bits are masked)<br>100: BAA23 to BAA4 (Lower 4 bits are masked)<br>101: BAA23 to BAA8 (Lower 8 bits are masked)<br>110: BAA23 to BAA12 (Lower 12 bits are masked)<br>111: BAA23 to BAA16 (Lower 16 bits are masked) |
| 2<br>1 | CSELA1<br>CSELA0 | 0<br>0 | R/W<br>R/W | Break Condition Select A<br>Selects break condition of channel A.<br>00: Instruction fetch is used as break condition<br>01: Data read cycle is used as break condition<br>10: Data write cycle is used as break condition<br>11: Data read/write cycle is used as break condition |
| 0 | BIEA | 0 | R/W | Break Interrupt Enable A<br>When this bit is 1, the PC break interrupt request of channel A is enabled. |

### 6.2.4 Break Control Register B (BCRB)

BCRB is the channel B break control register. The bit configuration is the same as for BCRA.

## 6.3 Operation

The operation flow from break condition setting to PC break interrupt exception handling is shown in section 6.3.1, PC Break Interrupt Due to Instruction Fetch, and 6.3.2, PC Break Interrupt Due to Data Access, taking the example of channel A.

### 6.3.1 PC Break Interrupt Due to Instruction Fetch

1. Set the break address in BARA.

   For a PC break caused by an instruction fetch, set the address of the first instruction byte as the break address.
2. Set the break conditions in BCR.

   Set bit 6 (CDA) to 0 to select the CPU because the bus master must be the CPU for a PC break caused by an instruction fetch. Set the address bits to be masked to bits 3 to 5 (BAMA0 to BAMA2). Set bits 1 and 2 (CSELA0 and CSELA1) to 00 to specify an instruction fetch as the break condition. Set bit 0 (BIEA) to 1 to enable break interrupts.
3. When the instruction at the set address is fetched, a PC break request is generated immediately before execution of the fetched instruction, and the condition match flag (CMFA) is set.
4. After priority determination by the interrupt controller, PC break interrupt exception handling is started.

### 6.3.2 PC Break Interrupt Due to Data Access

1. Set the break address in BARA.

   For a PC break caused by a data access, set the target ROM, RAM, I/O, or external address space address as the break address. Stack operations and branch address reads are included in data accesses.
2. Set the break conditions in BCRA.

   Select the bus master with bit 6 (CDA). Set the address bits to be masked to bits 3 to 5 (BAMA BAMA0 to BAMA2). Set bits 1 and 2 (CSELA0 and CSELA1) to 01, 10, or 11 to specify data access as the break condition. Set bit 0 (BIEA) to 1 to enable break interrupts.
3. After execution of the instruction that performs a data access on the set address, a PC break request is generated and the condition match flag (CMFA) is set.
4. After priority determination by the interrupt controller, PC break interrupt exception handling is started.

### 6.3.3 PC Break Operation at Consecutive Data Transfer

- When a PC break interrupt is generated at the transfer address of an EEPMOV.B instruction

  PC break exception handling is executed after all data transfers have been completed and the EEPMOV.B instruction has ended.
- When a PC break interrupt is generated at a DTC transfer address

  PC break exception handling is executed after the DTC has completed the specified number of data transfers, or after data for which the DISEL bit is set to 1 has been transferred.

### 6.3.4 Operation in Transitions to Power-Down Modes

The operation when a PC break interrupt is set for an instruction fetch at the address after a SLEEP instruction is shown below.

- When the SLEEP instruction causes a transition from high-speed (medium-speed) mode to sleep mode:

  After execution of the SLEEP instruction, a transition is not made to sleep mode, and PC break exception handling is executed. After execution of PC break exception handling, the instruction at the address after the SLEEP instruction is executed (figure 6.2 (A)).
- When the SLEEP instruction causes a transition to software standby mode:

  After execution of the SLEEP instruction, a transition is made to software standby mode, and PC break exception handling is not executed. However, the CMFA or CMFB flag is set (figure 6.2 (B)).

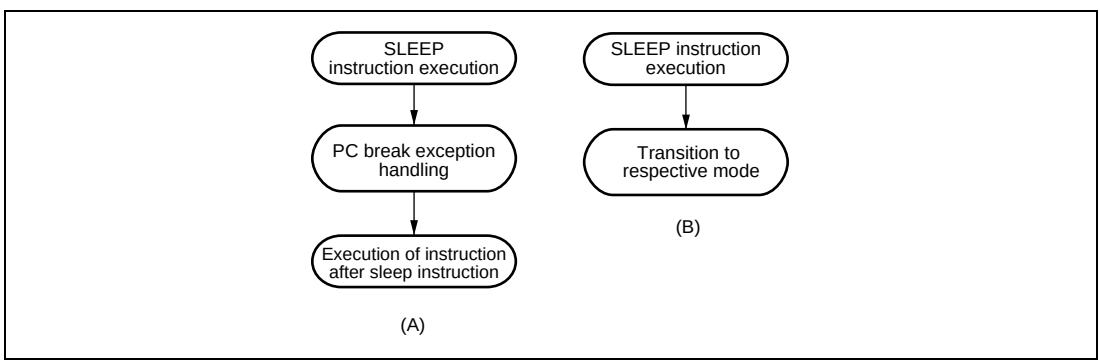



**Figure 6.2 Operation in Power-Down Mode Transitions**

### 6.3.5 When Instruction Execution Is Delayed by One State

While the break interrupt enable bit is set to 1, instruction execution is one state later than usual.

- For 1-word branch instructions (Bcc d:8, BSR, JSR, JMP, TRAPA, RTE, and RTS) in on-chip ROM or RAM.
- When break interrupt by instruction fetch is set, the set address indicates on-chip ROM or RAM space, and that address is used for data access, the instruction will be one state later than in normal operation.
- When break interrupt by instruction fetch is set and a break interrupt is generated, if the executing instruction immediately preceding the set instruction has one of the addressing modes shown below, and that address indicates on-chip ROM or RAM, the instruction will be one state later than in normal operation.

  Addressing modes: @ERn, @(d:16,ERn), @(d:32,ERn), @-ERn/ERn+, @aa:8, @aa:24, @aa:32, @(d:8,PC), @(d:16,PC), @@aa:8
- When break interrupt by instruction fetch is set and a break interrupt is generated, if the executing instruction immediately preceding the set instruction is NOP or SLEEP, or has #xx,Rn as its addressing mode, and that instruction is located in on-chip ROM or RAM, the instruction will be one state later than in normal operation.

## 6.4 Usage Notes

### 6.4.1 Module Stop Mode Setting

PBC operation can be disabled or enabled using the module stop control register. The initial setting is for PBC operation to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 6.4.2 PC Break Interrupts

The PC break interrupt is shared by channels A and B. The channel from which the request was issued must be determined by the interrupt handler.

### 6.4.3 CMFA and CMFB

The CMFA and CMFB flags are not automatically cleared to 0, so 0 must be written to CMFA or CMFB after first reading the flag while it is set to 1. If the flag is left set to 1, another interrupt will be requested after interrupt handling ends.

### 6.4.4 PC Break Interrupt when DTC Is Bus Master

A PC break interrupt generated when the DTC is the bus master is accepted after the bus mastership has been transferred to the CPU by the bus controller.

### 6.4.5 PC Break Set for Instruction Fetch at Address Following BSR, JSR, JMP, TRAPA, RTE, or RTS Instruction

Even if the instruction at the address following a BSR, JSR, JMP, TRAPA, RTE, or RTS instruction is fetched, it is not executed, and so a PC break interrupt is not generated by the instruction fetch at the next address.

### 6.4.6 I Bit Set by LDC, ANDC, ORC, or XORC Instruction

When the I bit is set by an LDC, ANDC, ORC, or XORC instruction, a PC break interrupt becomes valid two states after the end of the instruction execution. If a PC break interrupt is set for the instruction following one of these instructions, since interrupts, including NMI, are disabled for a 3-state period in the case of LDC, ANDC, ORC, and XOR, the next instruction is always executed. For details, see section 5, Interrupt Controller.

### 6.4.7 PC Break Set for Instruction Fetch at Address Following Bcc Instruction

A PC break interrupt is generated if the instruction at the next address is executed in accordance with the branch condition, and is not generated if the instruction at the next address is not executed.

### 6.4.8 PC Break Set for Instruction Fetch at Branch Destination Address of Bcc Instruction

A PC break interrupt is generated if the instruction at the branch destination is executed in accordance with the branch condition, and is not generated if the instruction at the branch destination is not executed.

The H8S/2600 CPU is driven by a system clock, denoted by the symbol φ.

The bus controller controls a memory cycle and a bus cycle. Different methods are used to access on-chip memory and on-chip support modules. The bus controller also has a bus arbitration function, and controls the operation of the internal bus masters: the CPU and data transfer controller (DTC).

## 7.1 Basic Timing

The period from one rising edge of φ to the next is referred to as a "state." The memory cycle or bus cycle consists of one, two, three, or four states. Different methods are used to access on-chip memory and on-chip support modules.

### 7.1.1 On-Chip Memory Access Timing (ROM, RAM)

On-chip memory is accessed in one state. The data bus is 16 bits wide, permitting both byte and word transfer instruction. Figure 7.1 shows the on-chip memory access cycle.

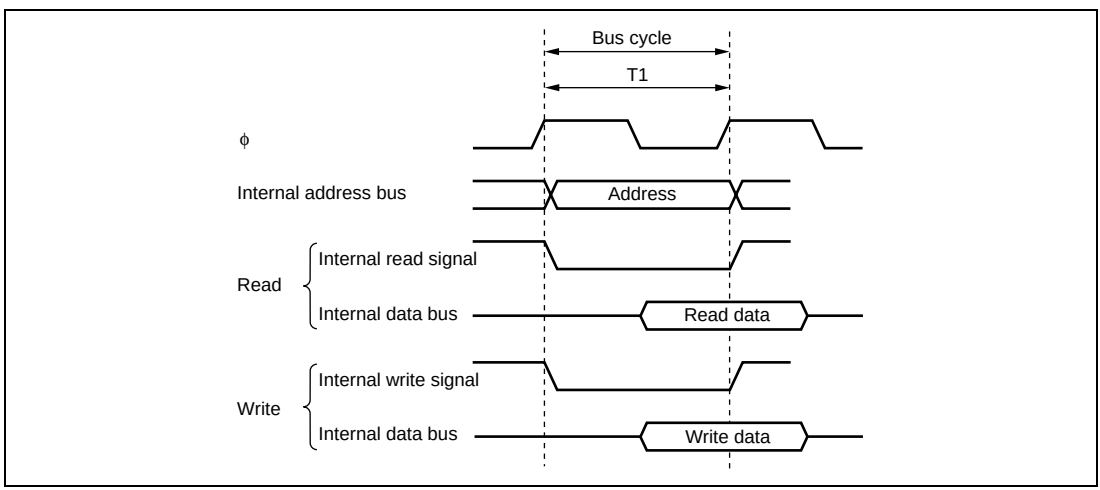



**Figure 7.1 On-Chip Memory Access Cycle**

### 7.1.2 On-Chip Support Module Access Timing

The on-chip support modules, except for the HCAN, SSU, and realtime input port data register, are accessed in two states. The data bus is either 8 bits or 16 bits wide, depending on the particular internal I/O register being accessed. For details, refer to section 22, List of Registers. Figure 7.2 shows access timing for the on-chip peripheral modules.

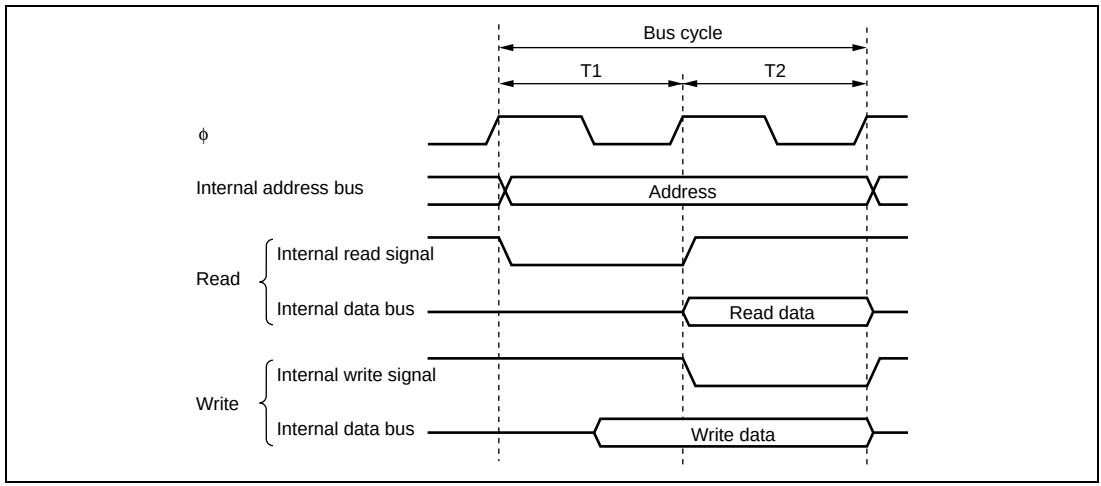



**Figure 7.2 On-Chip Support Module Access Cycle**

### 7.1.3 On-Chip HCAN Module Access Timing

On-chip HCAN module access is performed in four states. The data bus width is 16 bits. Wait states can be inserted by means of a wait request from the HCAN. On-chip HCAN module access timing is shown in figures 7.3.

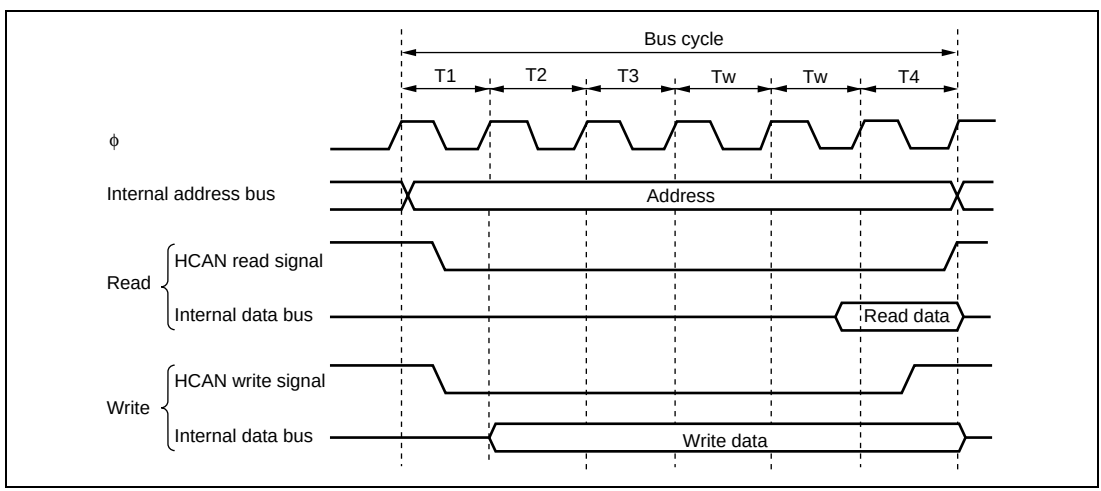



**Figure 7.3 On-Chip HCAN Module Access Cycle (with Wait States)**

### 7.1.4 On-chip SSU Module and Realtime Input Port Data Register Access Timing

The on-chip SSU module or realtime input port data register is accessed in three states. At this time, a data bus width is 16 bits. Figure 7.4 shows the SSU module access timing.

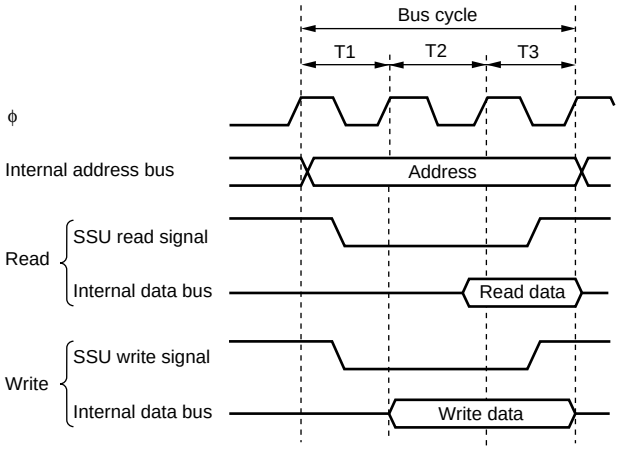



**Figure.7.4 On-chip SSU Module Access Cycle**

## 7.2 Bus Arbitration

The Bus Controller has a bus arbiter that arbitrates bus master operations.

There are two bus masters, the CPU and DTC, which perform read/write operations when they control the bus.

### 7.2.1 Order of Priority of the Bus Masters

Each bus master requests the bus mastership by means of a bus request signal. The bus arbiter detects the bus masters' bus request signals, and if the bus mastership is requested, sends a bus request acknowledge signal to the bus master making the request. If there are bus requests from more than one bus master, the bus request acknowledge signal is sent to the one with the highest priority. When a bus master receives the bus request acknowledge signal, it takes possession of the bus until that signal is cancelled.

The order of priority of the bus mastership is as follows:

(High) DTC > CPU (Low)

### 7.2.2 Bus Transfer Timing

Even if a bus request is received from a bus master with a higher priority than that of the bus master that has acquired the bus mastership and is currently operating, the bus mastership is not necessarily transferred immediately. The CPU is the lowest-priority bus master, and if a bus request is received from the DTC, the bus arbiter transfers the bus mastership to the bus master that issued the request. The timing for transfer of the bus mastership is as follows:

- The bus mastership is transferred at a break between bus cycles. However, if a bus cycle is executed in discrete operations, as in the case of a longword-size access, the bus mastership is not transferred between such operations. For details, refer to section 2.7, Bus Status in Instruction Execution in the H8S/2600 Series, H8S/2000 Series Programming Manual.
- If the CPU is in sleep mode, it transfers the bus mastership immediately.

The DTC can release the bus mastership after a vector read, a register information read (3 states), a single data transfer, or a register information write (3 states). It does not release the bus mastership during a register information read (3 states), a single data transfer, or a register information write (3 states).

# Section 8 Data Transfer Controller (DTC)

This LSI includes a data transfer controller (DTC). The DTC can be activated by an interrupt or software, to transfer data.

Figure 8.1 shows a block diagram of the DTC.

The DTC's register information is stored in the on-chip RAM. When the DTC is used, the RAME bit in SYSCR must be set to 1. A 32-bit bus connects the DTC to the on-chip RAM (1 kbyte), enabling 32-bit/1-state reading and writing of the DTC register information.

## 8.1 Features

- Transfer is possible over any number of channels
- Three transfer modes
  - Normal, repeat, and block transfer modes are available
- One activation source can trigger a number of data transfers (chain transfer)
- The direct specification of 16-Mbyte address space is possible
- Activation by software is possible
- Transfer can be set in byte or word units
- A CPU interrupt can be requested for the interrupt that activated the DTC
- Module stop mode can be set

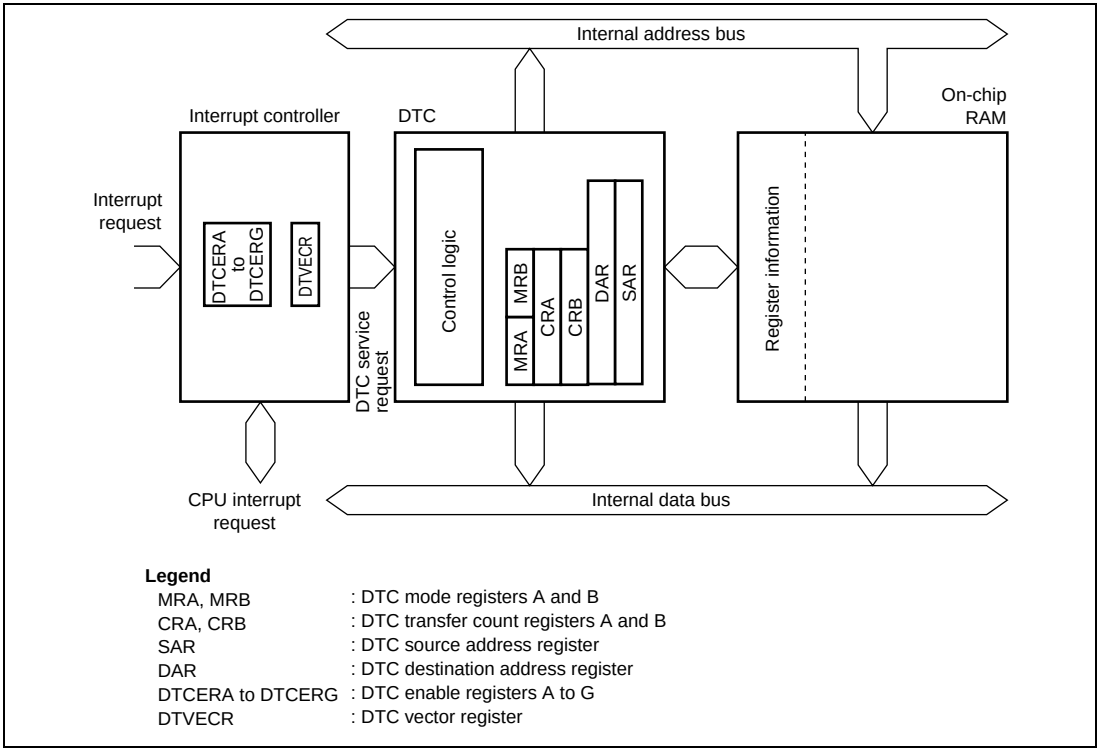



**Figure 8.1 Block Diagram of DTC**

## 8.2 Register Descriptions

The DTC has the following registers.

- DTC mode register A (MRA)
- DTC mode register B (MRB)
- DTC source address register (SAR)
- DTC destination address register (DAR)
- DTC transfer count register A (CRA)
- DTC transfer count register B (CRB)

These six registers cannot be directly accessed from the CPU.

When activated, the DTC reads a set of register information that is stored in on-chip RAM to the corresponding DTC registers and transfers data. After the data transfer, it writes a set of updated register information back to the RAM.

- DTC enable registers (DTCER)
- DTC vector register (DTVECR)

### 8.2.1 DTC Mode Register A (MRA)

MRA is an 8-bit register that selects the DTC operating mode.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | SM1 | Undefined | — | Source Address Mode 1 and 0 |
| 6 | SM0 | Undefined | — | These bits specify an SAR operation after a data transfer.<br>0X: SAR is fixed<br>10: SAR is incremented after a transfer (by +1 when Sz = 0; by +2 when Sz = 1)<br>11: SAR is decremented after a transfer (by –1 when Sz = 0; by –2 when Sz = 1) |
| 5 | DM1 | Undefined | — | Destination Address Mode 1 and 0 |
| 4 | DM0 | Undefined | — | These bits specify a DAR operation after a data transfer.<br>0X: DAR is fixed<br>10: DAR is incremented after a transfer (by +1 when Sz = 0; by +2 when Sz = 1)<br>11: DAR is decremented after a transfer (by –1 when Sz = 0; by –2 when Sz = 1) |
| 3 | MD1 | Undefined | — | DTC Mode |
| 2 | MD0 | Undefined | — | These bits specify the DTC transfer mode.<br>00: Normal mode<br>01: Repeat mode<br>10: Block transfer mode<br>11: Setting prohibited |
| 1 | DTS | Undefined | — | DTC Transfer Mode Select<br>Specifies whether the source side or the destination side is set to be a repeat area or block area, in repeat mode or block transfer mode.<br>0: Destination side is repeat area or block area<br>1: Source side is repeat area or block area |
| 0 | Sz | Undefined | — | DTC Data Transfer Size<br>Specifies the size of data to be transferred.<br>0: Byte-size transfer<br>1: Word-size transfer |

**Legend**

X : Don’t care

### 8.2.2 DTC Mode Register B (MRB)

MRB is an 8-bit register that selects the DTC operating mode.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CHNE | Undefined | — | DTC Chain Transfer Enable<br>When this bit is set to 1, a chain transfer will be performed. For details, refer to 8.5.4, Chain Transfer.<br>In data transfer with CHNE set to 1, determination of the end of the specified number of transfers, clearing of the interrupt source flag, and clearing of DTCER, are not performed. |
| 6 | DISEL | Undefined | — | DTC Interrupt Select<br>When this bit is set to 1, a CPU interrupt request is generated every time after the end of a data transfer. When this bit is set to 0, a CPU interrupt request is generated at the time when the specified number of data transfer ends. |
| 5 to 0 | — | Undefined | — | Reserved<br>These bits have no effect on DTC operation. Only 0 should be written to these bits. |

### 8.2.3 DTC Source Address Register (SAR)

SAR is a 24-bit register that designates the source address of data to be transferred by the DTC. For word-size transfer, specify an even source address.

### 8.2.4 DTC Destination Address Register (DAR)

DAR is a 24-bit register that designates the destination address of data to be transferred by the DTC. For word-size transfer, specify an even destination address.

### 8.2.5 DTC Transfer Count Register A (CRA)

CRA is a 16-bit register that designates the number of times data is to be transferred by the DTC.

In normal mode, the entire CRA functions as a 16-bit transfer counter (1 to 65536). It is decremented by 1 every time data is transferred, and transfer ends when the count reaches H′0000.

In repeat mode or block transfer mode, the CRA is divided into two parts; the upper 8 bits (CRAH) and the lower 8 bits (CRAL). CRAH holds the number of transfers while CRAL functions as an 8-bit transfer counter (1 to 256). CRAL is decremented by 1 every time data is transferred, and the contents of CRAH are sent when the count reaches H′00.

### 8.2.6 DTC Transfer Count Register B (CRB)

CRB is a 16-bit register that designates the number of times data is to be transferred by the DTC in block transfer mode. It functions as a 16-bit transfer counter (1 to 65536) that is decremented by 1 every time data is transferred, and transfer ends when the count reaches H′0000.

### 8.2.7 DTC Enable Registers (DTCER)

DTCER is comprised of seven registers; DTCERA to DTCERG, and is a register that specifies DTC activation interrupt sources. The correspondence between interrupt sources and DTCE bits is shown in table 8.1. For DTCE bit setting, use bit manipulation instructions such as BSET and BCLR for reading and writing. If all interrupts are masked, multiple activation sources can be set at one time (only at the initial setting) by writing data after executing a dummy read on the relevant register.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | DTCE7 | 0 | R/W | DTC Activation Enable |
| 6 | DTCE6 | 0 | R/W | Setting this bit to 1 specifies a relevant interrupt source as a DTC activation source. |
| 5 | DTCE5 | 0 | R/W | |
| 4 | DTCE4 | 0 | R/W | [Clearing conditions] |
| 3 | DTCE3 | 0 | R/W | • When the DISEL bit in MRB is 1 and the data transfer has ended |
| 2 | DTCE2 | 0 | R/W | • When the specified number of transfers have ended |
| 1 | DTCE1 | 0 | R/W | |
| 0 | DTCE0 | 0 | R/W | These bits are not cleared when the DISEL bit is 0 and the specified number of transfers have not been completed. |

### 8.2.8 DTC Vector Register (DTVECR)

DTVECR is an 8-bit readable/writable register that enables or disables DTC activation by software, and sets a vector number for the software activation interrupt.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | SWDTE | 0 | R/W | DTC Software Activation Enable<br>Setting this bit to 1 activates DTC. Only 1 can be written to this bit.<br>[Clearing conditions]<br>• When the DISEL bit is 0 and the specified number of transfers have not ended<br>• When 0 is written to the DISEL bit after a software-activated data transfer end interrupt (SWDTEND) request has been sent to the CPU.<br>When the DISEL bit is 1 and data transfer has ended or when the specified number of transfers have ended, this bit will not be cleared. |
| 6 | DTVEC6 | 0 | R/W | DTC Software Activation Vectors 0 to 6<br>These bits specify a vector number for DTC software activation.<br>The vector address is expressed as H′0400 + (vector number × 2). For example, when DTVEC6 to DTVEC0 = H′10, the vector address is H′0420. When the bit SWDTE is 0, these bits can be written. |
| 5 | DTVEC5 | 0 | R/W | |
| 4 | DTVEC4 | 0 | R/W | |
| 3 | DTVEC3 | 0 | R/W | |
| 2 | DTVEC2 | 0 | R/W | |
| 1 | DTVEC1 | 0 | R/W | |
| 0 | DTVEC0 | 0 | R/W | |

## 8.3 Activation Sources

The DTC operates when activated by an interrupt or by a write to DTVECR by software. An interrupt request can be directed to the CPU or DTC, as designated by the corresponding DTCER bit. At the end of a data transfer (or the last consecutive transfer in the case of chain transfer), the activation source or corresponding DTCER bit is cleared. The activation source flag, in the case of RXI_0, for example, is the RDRF flag of SCI_0.

When an interrupt has been designated a DTC activation source, the existing CPU mask level and interrupt controller priorities have no effect. If there is more than one activation source at the same time, the DTC operates in accordance with the default priorities.

Figure 8.2 shows a block diagram of DTC activation source control. For details, see section 5, Interrupt Controller.

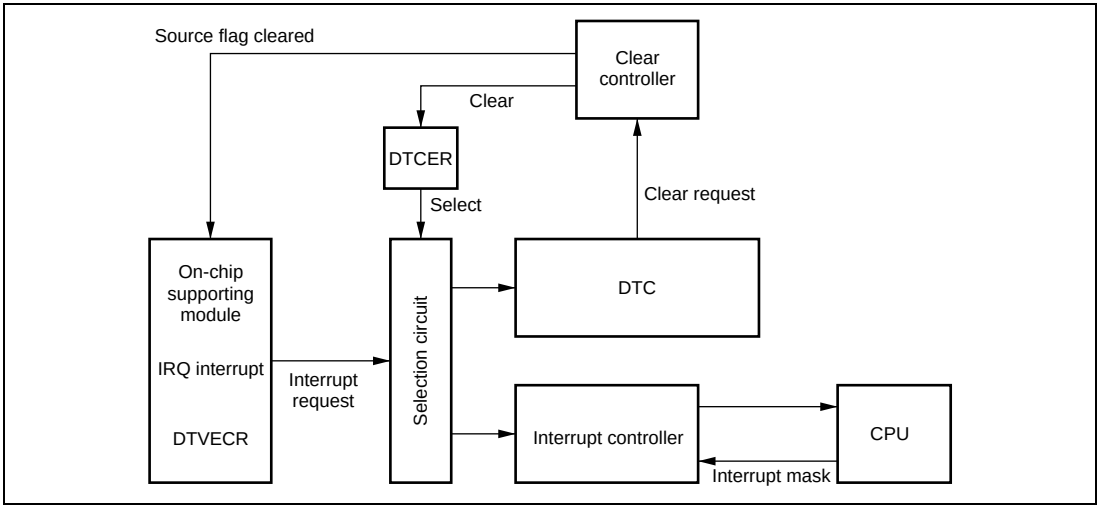



**Figure 8.2 Block Diagram of DTC Activation Source Control**

## 8.4 Location of Register Information and DTC Vector Table

Locate the register information in the on-chip RAM (addresses: H′FFEBC0 to H′FFEFBF). Register information should be located at an address that is a multiple of four within the range. Locating the register information in address space is shown in figure 8.3. Locate the MRA, SAR, MRB, DAR, CRA, and CRB registers, in that order, from the start address of the register information.

In the case of chain transfer, register information should be located in consecutive areas and the register information start address should be located at the vector address corresponding to the interrupt source as shown in figure 8.3. The DTC reads the start address of the register information from the vector address set for each activation source, and then reads the register information from that start address.

When the DTC is activated by software, the vector address is obtained from: H′0400 + (DTVECR[6:0] × 2). For example, if DTVECR is H′10, the vector address is H′0420. The configuration of the vector address is the same in both normal and advanced modes, a 2-byte unit being used in both cases. These two bytes specify the lower bits of the register information start address.

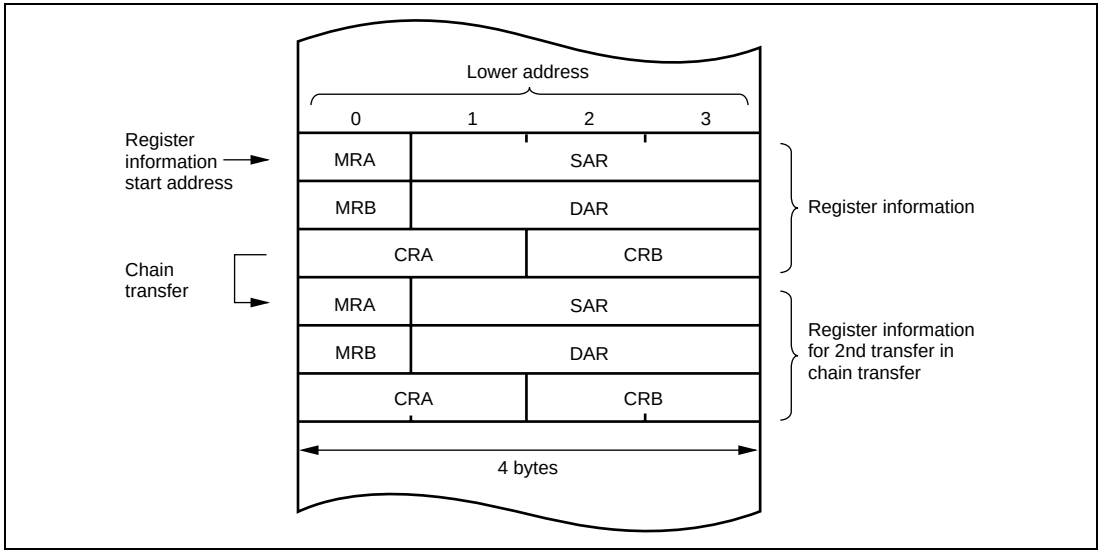



**Figure 8.3 Location of DTC Register Information in Address Space**

**Table 8.1 Interrupt Sources, DTC Vector Addresses, and Corresponding DTCEs**

| Interrupt Source | Origin of Interrupt Source | Vector Number | DTC Vector Address | DTCE* | Priority |
|---|---|---|---|---|---|
| Software | Write to DTVECR | DTVECR | H′0400 + (vector number × 2) | — | High |
| External pin | IRQ0 | 16 | H′0420 | DTCEA7 | |
| | IRQ1 | 17 | H′0422 | DTCEA6 | |
| | IRQ2 | 18 | H′0424 | DTCEA5 | |
| | IRQ3 | 19 | H′0426 | DTCEA4 | |
| | IRQ4 | 20 | H′0428 | DTCEA3 | |
| | IRQ5 | 21 | H′042A | DTCEA2 | |
| | Reserved for | 22 | H′042C | DTCEA1 | |
| | System use | 23 | H′042E | DTCEA0 | |
| A/D counter | ADI (A/D conversion end) | 28 | H′0438 | DTCEB6 | |
| TPU | TGIA_0 | 32 | H′0440 | DTCEB5 | |
| channel 0 | TGIB_0 | 33 | H′0442 | DTCEB4 | |
| | TGIC_0 | 34 | H′0444 | DTCEB3 | |
| | TGID_0 | 35 | H′0446 | DTCEB2 | |
| TPU | TGIA_1 | 40 | H′0450 | DTCEB1 | |
| channel 1 | TGIB_1 | 41 | H′0452 | DTCEB0 | |
| TPU | TGIA_2 | 44 | H′0458 | DTCEC7 | |
| channel 2 | TGIB_2 | 45 | H′045A | DTCEC6 | |
| TPU | TGIA_3 | 48 | H′0460 | DTCEC5 | |
| channel 3 | TGIB_3 | 49 | H′0462 | DTCEC4 | |
| | TGIC_3 | 50 | H′0464 | DTCEC3 | |
| | TGID_3 | 51 | H′0466 | DTCEC2 | |
| TPU | TGIA_4 | 56 | H′0470 | DTCEC1 | |
| channel 4 | TGIB_4 | 57 | H′0472 | DTCEC0 | |
| TPU | TGIA_5 | 60 | H′0478 | DTCED5 | |
| channel 5 | TGIB_5 | 61 | H′047A | DTCED4 | Low |

| Interrupt Source | Origin of Interrupt Source | Vector Number | DTC Vector Address | DTCE* | Priority |
|---|---|---|---|---|---|
| 8-bit timer channel 0 | CMIA_1 | 64 | H′0480 | DTCED3 | High |
| | | 65 | H′0482 | DTCED2 | |
| 8-bit timer channel 1 | CMIB_1 | 68 | H′0488 | DTCED1 | |
| | | 69 | H′048A | DTCED0 | |
| | Reserved | 72 | H′0490 | DTCEE7 | |
| | | 73 | H′0492 | DTCEE6 | |
| | | 74 | H′0494 | DTCEE5 | |
| | | 75 | H′0496 | DTCEE4 | |
| SCI channel 0 | RXI_0 | 81 | H′04A2 | DTCEE3 | |
| | TXI_0 | 82 | H′04A4 | DTCEE2 | |
| SCI channel 2 | RXI_2 | 89 | H′04B2 | DTCEF7 | |
| | TXI_2 | 90 | H′04B4 | DTCEF6 | |
| 8-bit timer channel 2 | CMIA_2 | 92 | H′04B8 | DTCEF5 | |
| | CMIB_2 | 93 | H′04BA | DTCEF4 | |
| 8-bit timer channel 3 | CMIA_3 | 96 | H′04C0 | DTCEF3 | |
| | CMIB3 | 97 | H′04C2 | DTCEF2 | |
| HCAN | Reserved for system use | 104 | H′04D0 | DTCEG7 | |
| | RM0 | 105 | H′04D2 | DTCEG6 | |
| | Reserved for system use | 106 | H′04D4 | DTCEG5 | |
| | Reserved for system use | 107 | H′04D6 | DTCEG4 | |
| SSU channel 0 | SSRx_i0 | 109 | H′04DA | DTCEG2 | |
| | SSTx_i0 | 110 | H′04DC | DTCEG1 | Low |

Note: * DTCE bits with no corresponding interrupt are reserved, and the write value should aslways be 0.

## 8.5 Operation

Register information is stored in on-chip RAM. When activated, the DTC reads register information in on-chip RAM and transfers data. After the data transfer, the DTC writes updated register information back to the on-chip RAM.

The pre-storage of register information in the on-chip RAM makes it possible to transfer data over any required number of channels. The transfer mode can be specified as normal, repeat, and block transfer mode. Setting the CHNE bit in MRB to 1 makes it possible to perform a number of transfers with a single activation source (chain transfer).

The 24-bit SAR designates the DTC transfer source address, and the 24-bit DAR designates the transfer destination address. After each transfer, SAR and DAR are independently incremented, decremented, or left fixed depending on its register information.

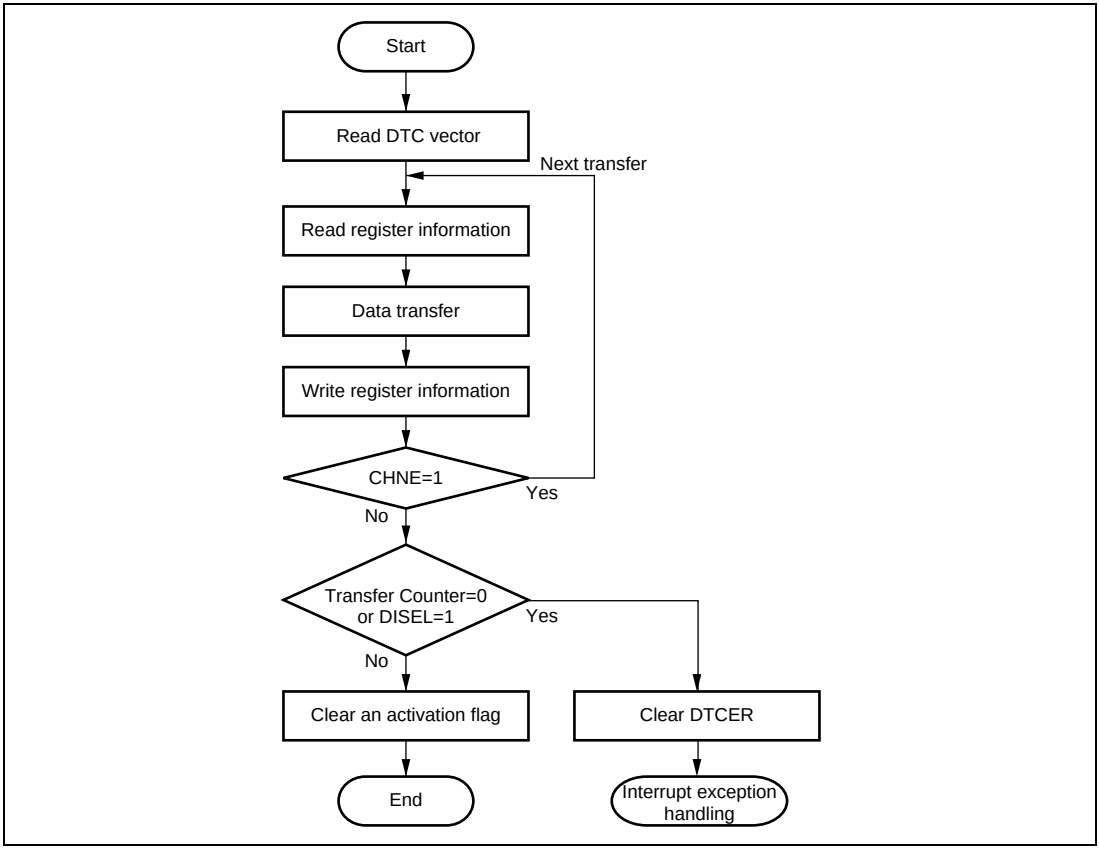



**Figure 8.4 Flowchart of DTC Operation**

### 8.5.1 Normal Mode

In normal mode, one operation transfers one byte or one word of data.

Table 8.2 lists the register information in normal mode.

From 1 to 65,536 transfers can be specified. Once the specified number of transfers have been completed, a CPU interrupt can be requested.

**Table 8.2 Register Information in Normal Mode**

| Name | Abbreviation | Function |
|---|---|---|
| DTC source address register | SAR | Designates source address |
| DTC destination address register | DAR | Designates destination address |
| DTC transfer count register A | CRA | Designates transfer count |
| DTC transfer count register B | CRB | Not used |

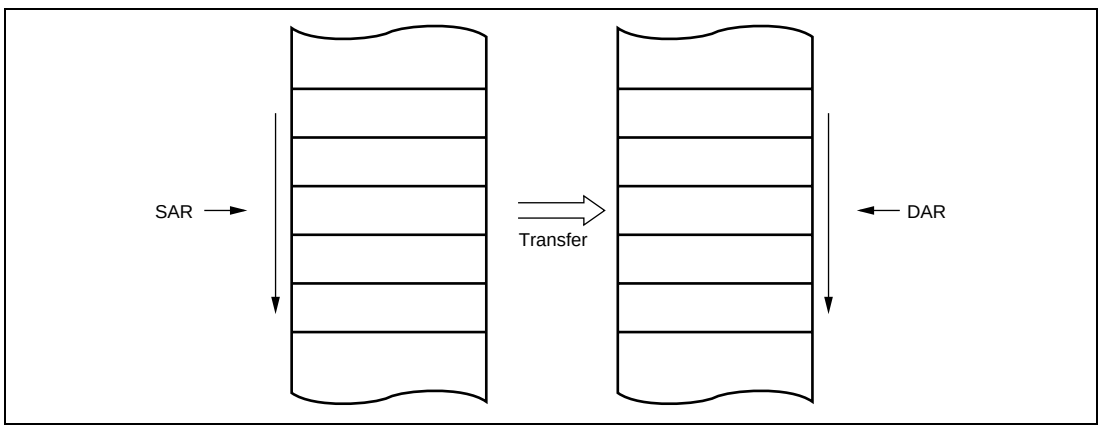



**Figure 8.5 Memory Mapping in Normal Mode**

### 8.5.2 Repeat Mode

In repeat mode, one operation transfers one byte or one word of data. Table 8.3 lists the register information in repeat mode.

From 1 to 256 transfers can be specified. Once the specified number of transfers have ended, the initial state of the transfer counter and the address register specified as the repeat area is restored, and transfer is repeated. In repeat mode the transfer counter value does not reach H′00, and therefore CPU interrupts cannot be requested when DISEL = 0.

**Table 8.3 Register Information in Repeat Mode**

| Name | Abbreviation | Function |
|---|---|---|
| DTC source address register | SAR | Designates source address |
| DTC destination address register | DAR | Designates destination address |
| DTC transfer count register AH | CRAH | Holds number of transfers |
| DTC transfer count register AL | CRAL | Designates transfer count |
| DTC transfer count register B | CRB | Not used |

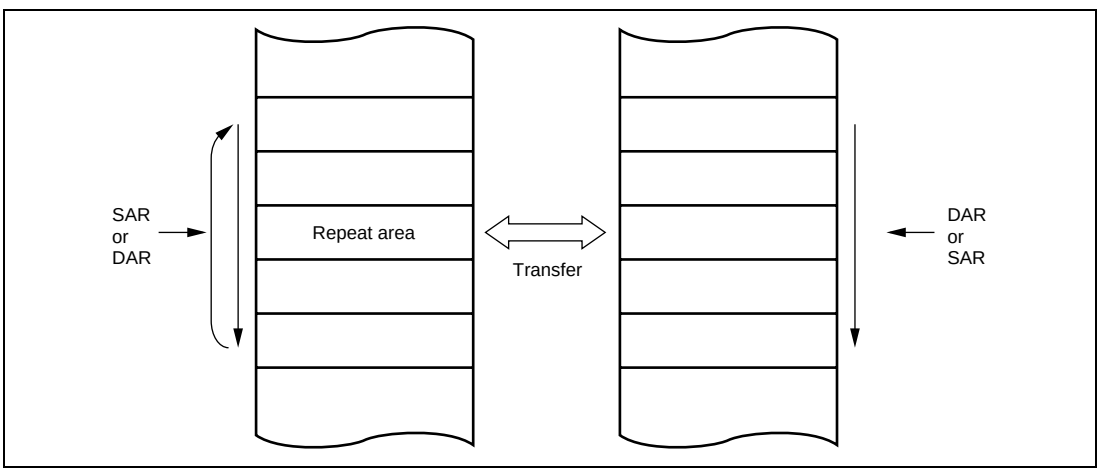



**Figure 8.6 Memory Mapping in Repeat Mode**

### 8.5.3 Block Transfer Mode

In block transfer mode, one operation transfers one block of data. Either the transfer source or the transfer destination is designated as a block area. Table 8.4 lists the register information in block transfer mode.

The block size can be between 1 and 256. When the transfer of one block ends, the initial state of the block size counter and the address register specified as the block area is restored. The other address register is then incremented, decremented, or left fixed.

From 1 to 65,536 transfers can be specified. Once the specified number of transfers have been completed, a CPU interrupt is requested.

**Table 8.4 Register Information in Block Transfer Mode**

| Name | Abbreviation | Function |
|---|---|---|
| DTC source address register | SAR | Designates source address |
| DTC destination address register | DAR | Designates destination address |
| DTC transfer count register AH | CRAH | Holds block size |
| DTC transfer count register AL | CRAL | Designates block size count |
| DTC transfer count register B | CRB | Transfer count |

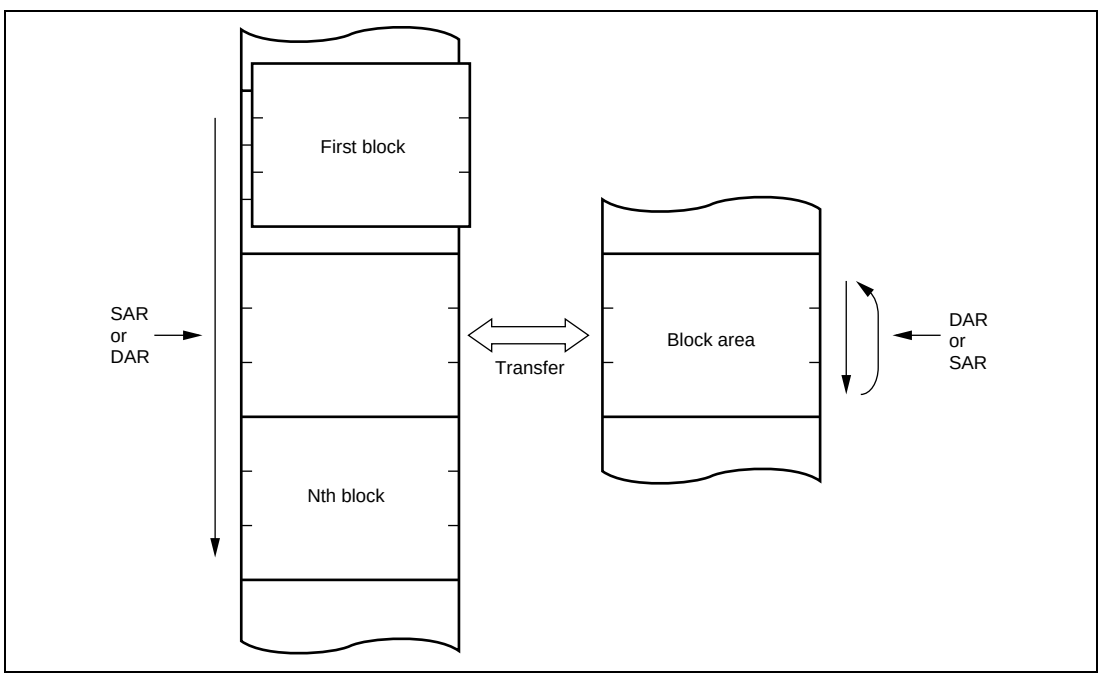



**Figure 8.7 Memory Mapping in Block Transfer Mode**

### 8.5.4 Chain Transfer

Setting the CHNE bit in MRB to 1 enables a number of data transfers to be performed consecutively in response to a single transfer request. SAR, DAR, CRA, CRB, MRA, and MRB, which define data transfers, can be set independently.

Figure 8.8 shows the outline of the chain transfer operation.

When activated, the DTC reads the register information start address stored at the vector address corresponding to the activation source, and then reads the first register information at that start address. After data transfer ends, the CHNE bit will be tested. When it has been set to 1, DTC reads the next register information located in a consecutive area and performs the data transfer. These sequences are repeated until the CHNE bit is cleared to 0.

In the case of transfer with CHNE set to 1, an interrupt request to the CPU is not generated at the end of the specified number of transfers or by setting of the DISEL bit to 1, and the interrupt source flag for the activation source is not affected.

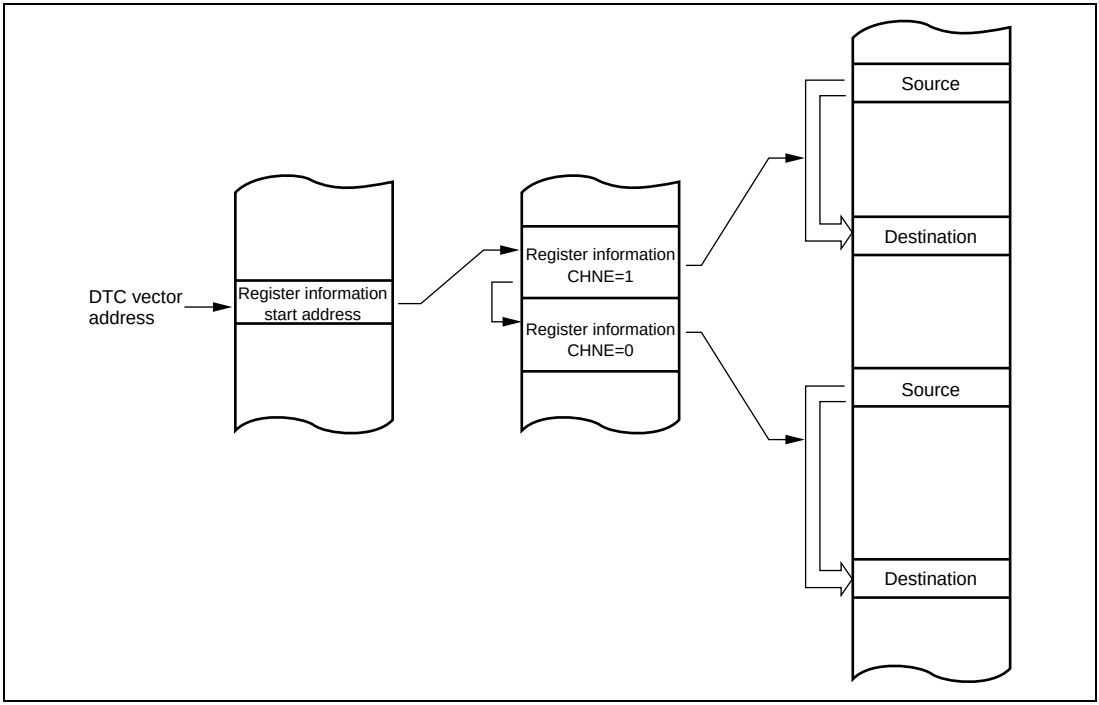



**Figure 8.8 Chain Transfer Operation**

### 8.5.5 Interrupts

An interrupt request is issued to the CPU when the DTC has completed the specified number of data transfers, or a data transfer for which the DISEL bit was set to 1. In the case of interrupt activation, the interrupt set as the activation source is generated. These interrupts to the CPU are subject to CPU mask level and interrupt controller priority level control.

In the case of software activation, a software-activated data transfer end interrupt (SWDTEND) is generated.

When the DISEL bit is 1 and one data transfer has been completed, or the specified number of transfers have been completed, after data transfer ends the SWDTE bit is held at 1 and an SWDTEND interrupt is generated. The interrupt handling routine will then clear the SWDTE bit to 0.

When the DTC is activated by software, an SWDTEND interrupt is not generated during a data transfer wait or during data transfer even if the SWDTE bit is set to 1.

### 8.5.6 Operation Timing

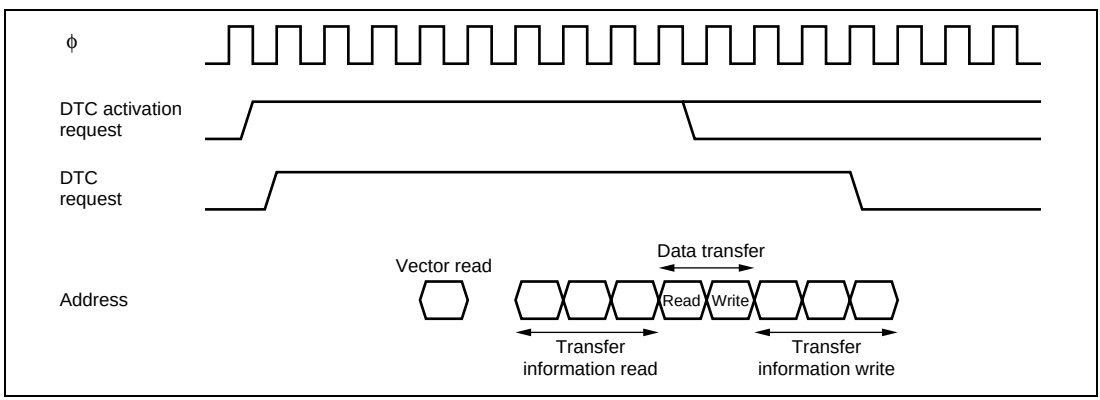



**Figure 8.9 DTC Operation Timing (Example in Normal Mode or Repeat Mode)**

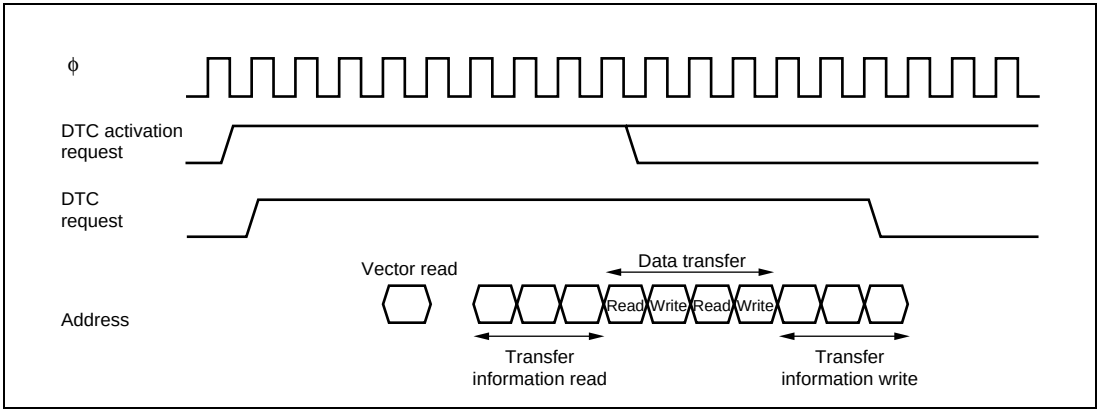



**Figure 8.10 DTC Operation Timing (Example of Block Transfer Mode, with Block Size of 2)**

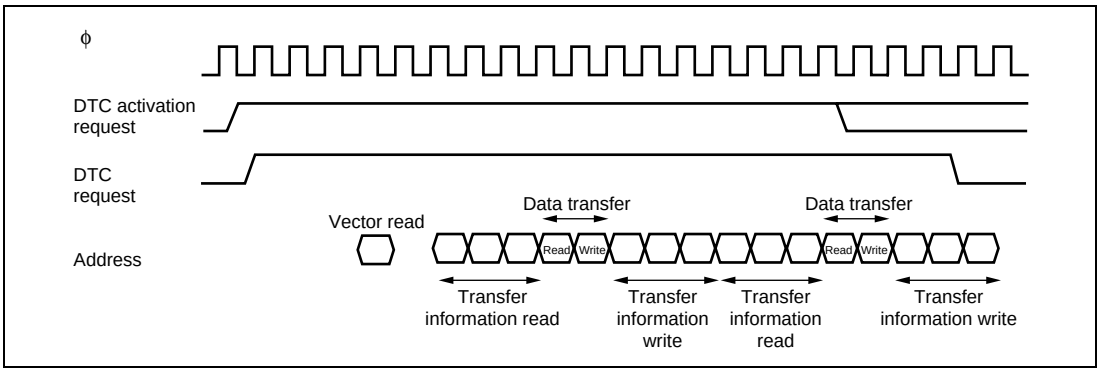



**Figure 8.11 DTC Operation Timing (Example of Chain Transfer)**

### 8.5.7 Number of DTC Execution States

Table 8.5 lists execution status for a single DTC data transfer, and table 8.6 shows the number of states required for each execution status.

**Table 8.5 DTC Execution Status**

| Mode | Vector Read I | Register Information Read/Write J | Data Read K | Data Write L | Internal Operations M |
|---|---|---|---|---|---|
| Normal | 1 | 6 | 1 | 1 | 3 |
| Repeat | 1 | 6 | 1 | 1 | 3 |
| Block transfer | 1 | 6 | N | N | 3 |

**Legend**

N: Block size (initial setting of CRAH and CRAL)

**Table 8.6 Number of States Required for Each Execution Status**

| Object to be Accessed | | On-Chip RAM | On-Chip ROM | On-Chip I/O Registers | | External Devices* | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bus width | | 32 | 16 | 8 | 16 | 8 | | 16 | |
| Access states | | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 3 |
| Execution status | Vector read $S_I$ | — | 1 | — | — | 4 | 6+2m | 2 | 3+m |
| | Register information read/write $S_J$ | 1 | — | — | — | — | — | — | — |
| | Byte data read $S_K$ | 1 | 1 | 2 | 2 | 2 | 3+m | 2 | 3+m |
| | Word data read $S_K$ | 1 | 1 | 4 | 2 | 4 | 6+2m | 2 | 3+m |
| | Byte data write $S_L$ | 1 | 1 | 2 | 2 | 2 | 3+m | 2 | 3+m |
| | Word data write $S_L$ | 1 | 1 | 4 | 2 | 4 | 6+2m | 2 | 3+m |
| | Internal operation $S_M$ | 1 | | | | | | | |

Note:* Not available in this LSI.

The number of execution states is calculated from using the formula below. Note that Σ is the sum of all transfers activated by one activation source (the number in which the CHNE bit is set to 1, plus 1).

$$\text{Number of execution states} = I \cdot (1 + S_I) + \Sigma (J \cdot S_J + K \cdot S_K + L \cdot S_L) + M \cdot S_M$$

For example, when the DTC vector address table is located in the on-chip ROM, normal mode is set, and data is transferred from on-chip ROM to an internal I/O register, then the time required for the DTC operation is 13 states. The time from activation to the end of the data write is 10 states.

## 8.6 Procedures for Using DTC

### 8.6.1 Activation by Interrupt

The procedure for using the DTC with interrupt activation is as follows:

1. Set the MRA, MRB, SAR, DAR, CRA, and CRB register information in on-chip RAM.
2. Set the start address of the register information in the DTC vector address.
3. Set the corresponding bit in DTCER to 1.
4. Set the enable bits for the interrupt sources to be used as the activation sources to 1. The DTC is activated when an interrupt used as an activation source is generated.
5. After one data transfer has been completed, or after the specified number of data transfers have been completed, the DTCE bit is cleared to 0 and a CPU interrupt is requested. If the DTC is to continue transferring data, set the DTCE bit to 1.

### 8.6.2 Activation by Software

The procedure for using the DTC with software activation is as follows:

1. Set the MRA, MRB, SAR, DAR, CRA, and CRB register information in on-chip RAM.
2. Set the start address of the register information in the DTC vector address.
3. Check that the SWDTE bit is 0.
4. Write 1 to SWDTE bit and the vector number to DTVECR.
5. Check the vector number written to DTVECR.
6. After one data transfer has been completed, if the DISEL bit is 0 and a CPU interrupt is not requested, the SWDTE bit is cleared to 0. If the DTC is to continue transferring data, set the SWDTE bit to 1. When the DISEL bit is 1, or after the specified number of data transfers have been completed, the SWDTE bit is held at 1 and a CPU interrupt is requested.

## 8.7 Examples of Use of the DTC

### 8.7.1 Normal Mode

An example is shown in which the DTC is used to receive 128 bytes of data via the SCI.

1. Set MRA to a fixed source address (SM1 = SM0 = 0), incrementing destination address (DM1 = 1, DM0 = 0), normal mode (MD1 = MD0 = 0), and byte size (Sz = 0). The DTS bit can have any value. Set MRB for one data transfer by one interrupt (CHNE = 0, DISEL = 0). Set the SCI RDR address in SAR, the start address of the RAM area where data will be received in DAR, and 128 (H′0080) in CRA. CRB can be set to any value.
2. Set the start address of the register information at the DTC vector address.

3. Set the corresponding bit in DTCER to 1.
4. Set the SCI to the appropriate receive mode. Set the RIE bit in SCR to 1 to enable the reception complete (RXI) interrupt. Since the generation of a receive error during the SCI reception operation will disable subsequent reception, the CPU should be enabled to accept receive error interrupts.
5. Each time the reception of one byte of data has been completed on the SCI, the RDRF flag in SSR is set to 1, an RXI interrupt is generated, and the DTC is activated. The receive data is transferred from RDR to RAM by the DTC. DAR is incremented and CRA is decremented. The RDRF flag is automatically cleared to 0.
6. When CRA becomes 0 after the 128 data transfers have been completed, the RDRF flag is held at 1, the DTCE bit is cleared to 0, and an RXI interrupt request is sent to the CPU. The interrupt handling routine will perform wrap-up processing.

### 8.7.2 Chain Transfer

An example of DTC chain transfer is shown in which pulse output is performed using the PPG. Chain transfer can be used to perform pulse output data transfer and PPG output trigger cycle updating. Repeat mode transfer to the PPG's NDR is performed in the first half of the chain transfer, and normal mode transfer to the TPU's TGR in the second half. This is because clearing of the activation source and interrupt generation at the end of the specified number of transfers are restricted to the second half of the chain transfer (transfer when CHNE = 0).

1. Perform settings for transfer to the PPG's NDR. Set MRA to incrementing source address (SM1 = 1, SM0 = 0), a fixed destination address (DM1 = DM0 = 0), repeat mode (MD1 = 0, MD0 = 1), and word size (Sz = 1). Set the source side as a repeat area (DTS = 1). Set MRB to chain mode (CHNE = 1, DISEL = 0). Set the data table start address in SAR, the NDRH address in DAR, and the data table size in CRAH and CRAL. CRB can be set to any value.
2. Perform settings for transfer to the TPU's TGR. Set MRA to incrementing source address (SM1 = 1, SM0 = 0), a fixed destination address (DM1 = DM0 = 0), normal mode (MD1 = MD0 = 0), and word size (Sz = 1). Set the data table start address in SAR, the TGRA address in DAR, and the data table size in CRA. CRB can be set to any value.
3. Locate the TPU transfer register information consecutively after the NDR transfer register information.
4. Set the start address of the NDR transfer register information to the DTC vector address.
5. Set the bit corresponding to TGIA in DTCER to 1.
6. Set TGRA as an output compare register (output disabled) with TIOR, and enable the TGIA interrupt with TIER.
7. Set the initial output value in PODR, and the next output value in NDR. Set bits in DDR and NDER for which output is to be performed to 1. Using PCR, select the TPU compare match to be used as the output trigger.
8. Set the CST bit in TSTR to 1, and start the TCNT count operation.

9. Each time a TGRA compare match occurs, the next output value is transferred to NDR and the set value of the next output trigger period is transferred to TGRA. The activation source TGFA flag is cleared.
10. When the specified number of transfers are completed (the TPU transfer CRA value is 0), the TGFA flag is held at 1, the DTCE bit is cleared to 0, and a TGIA interrupt request is sent to the CPU. Termination processing should be performed in the interrupt handling routine.

### 8.7.3 Software Activation

An example is shown in which the DTC is used to transfer a block of 128 bytes of data by means of software activation. The transfer source address is H′1000 and the destination address is H′2000. The vector number is H′60, so the vector address is H′04C0.

1. Set MRA to incrementing source address (SM1 = 1, SM0 = 0), incrementing destination address (DM1 = 1, DM0 = 0), block transfer mode (MD1 = 1, MD0 = 0), and byte size (Sz = 0). The DTS bit can have any value. Set MRB for one block transfer by one interrupt (CHNE = 0). Set the transfer source address (H′1000) in SAR, the destination address (H′2000) in DAR, and 128 (H′8080) in CRA. Set 1 (H′0001) in CRB.
2. Set the start address of the register information at the DTC vector address (H′04C0).
3. Check that the SWDTE bit in DTVECR is 0. Check that there is currently no transfer activated by software.
4. Write 1 to the SWDTE bit and the vector number (H′60) to DTVECR. The write data is H′E0.
5. Read DTVECR again and check that it is set to the vector number (H′60). If it is not, this indicates that the write failed. This is presumably because an interrupt occurred between steps 3 and 4 and led to a different software activation. To activate this transfer, go back to step 3.
6. If the write was successful, the DTC is activated and a block of 128 bytes of data is transferred.
7. After the transfer, an SWDTEND interrupt occurs. The interrupt handling routine should clear the SWDTE bit to 0 and perform other wrap-up processing.

## 8.8 Usage Notes

### 8.8.1 Module Stop Mode Setting

DTC operation can be disabled or enabled using the module stop control register. The initial setting is for DTC operation to be enabled. Register access is disabled by setting module stop mode. Note that module stop mode cannot be set during DTC being activated. For details, refer to section 21, Power-Down Modes.

### 8.8.2 On-Chip RAM

The MRA, MRB, SAR, DAR, CRA, and CRB registers are all located in on-chip RAM. When the DTC is used, the RAME bit in SYSCR must not be cleared to 0.

### 8.8.3 DTCE Bit Setting

For DTCE bit setting, use bit manipulation instructions such as BSET and BCLR. If all interrupts are masked, multiple activation sources can be set at one time (only at the initial setting) by writing data after executing a dummy read on the relevant register.

# Section 9 I/O Ports

Table 9.1 summarizes the port functions. The pins of each port also have other functions such as input/output or interrupt input pins of on-chip peripheral modules.

Each I/O port includes a data direction register (DDR) that controls input/output, a data register (DR) that stores output data, and a port register (PORT) used to read the pin states. The input-only ports do not have a DR or DDR register.

Ports A to D have built-in input pull-up MOS functions and input pull-up MOS control registers (PCR) to control the on/off state of input pull-up MOS.

Ports A to C include an open-drain control register (ODR) that controls the on/off state of the output buffer PMOS.

All the I/O ports can drive a single TTL load and a 30 pF capacitive load.

**Table 9.1 Port Functions**

| Port | Description | Port and Other Functions Name | Input/Output and Output Type |
|---|---|---|---|
| Port 1 | General I/O port also functioning as TPU_2, TPU_1, and TPU_0 I/O pins, PPG output pins, and interrupt input pins | P17/PO15/TIOCB2/TCLKD | |
| | | P16/PO14/TIOCA2/$\overline{\text{IRQ1}}$ | |
| | | P15/PO13/TIOCB1/TCLKC | |
| | | P14/PO12/TIOCA1/$\overline{\text{IRQ0}}$ | |
| | | P13/PO11/TIOCD0/TCLKB | |
| | | P12/PO10/TIOCC0/TCLKA | |
| | | P11/PO9/TIOCB0 | |
| | | P10/PO8/TIOCA0 | |
| Port 3 | General I/O port also functioning as SCI_0 I/O pins and interrupt input pins | P37 | |
| | | P36 | |
| | | P35/$\overline{\text{IRQ5}}$ | |
| | | P34 | |
| | | P33 | |
| | | P32/SCK0/$\overline{\text{IRQ4}}$ | |
| | | P31/RxD0 | |
| | | P30/TxD0 | |
| Port 4 | General input port also functioning as A/D converter analog inputs | P47/AN7 | |
| | | P46/AN6 | |
| | | P45/AN5 | |
| | | P44/AN4 | |
| | | P43/AN3 | |
| | | P42/AN2 | |
| | | P41/AN1 | |
| | | P40/AN0 | |
| Port 7 | General I/O port also functioning as TMR_0, TMR_1, TMR_2, and TMR_3 I/O pins | P77 | |
| | | P76 | |
| | | P75/TMO3 | |
| | | P74/TMO2 | |
| | | P73/TMO1 | |
| | | P72/TMO0 | |
| | | P71/TMCI23/TMRI23 | |
| | | P70/TMCI01/TMRI01 | |

| Port | Description | Port and Other Functions Name | Input/Output and Output Type |
|---|---|---|---|
| Port 9 | General input port also functioning as A/D converter analog inputs | P97/AN15<br>P96/AN14<br>P95/AN13<br>P94/AN12<br>P93/AN11<br>P92/AN10<br>P91/AN9<br>P90/AN8 | |
| Port A | General I/O port also functioning as SCI_2 I/O pins | PA3/SCK2<br>PA2/RxD2<br>PA1/TxD2<br>PA0 | Built-in input pull-up MOS<br>Push-pull or open-drain output selectable |
| Port B | General I/O port also functioning as TPU_5, TPU_4, and TPU_3 I/O pins | PB7/TIOCB5<br>PB6/TIOCA5<br>PB5/TIOCB4<br>PB4/TIOCA4<br>PB3/TIOCD3<br>PB2/TIOCC3<br>PB1/TIOCB3<br>PB0/TIOCA3 | Built-in input pull-up MOS<br>Push-pull or open-drain output selectable |
| Port C | General I/O port also functioning as SSU_0 and SSU_1 I/O pins | PC7/$\overline{\text{SCS1}}$<br>PC6/SSCK1<br>PC5/SSI1<br>PC4/SSO1<br>PC3/$\overline{\text{SCS0}}$<br>PC2/SSCK0<br>PC1/SSI0<br>PC0/SSO0 | Built-in input pull-up MOS<br>Push-pull or open-drain output selectable |

| Port | Description | Port and Other Functions Name | Input/Output and Output Type |
|---|---|---|---|
| Port D | General I/O port | PD7 | Built-in input pull-up MOS |
| | | PD6 | |
| | | PD5 | |
| | | PD4 | |
| | | PD3 | |
| | | PD2 | |
| | | PD1 | |
| | | PD0 | |
| Port F | General I/O port also functioning as interrupt input pins, an A/D converter start trigger input pin, and a system clock output pin (φ) | PF7/φ | |
| | | PF6 | |
| | | PF5 | |
| | | PF4 | |
| | | PF3/$\overline{\text{ADTRG}}$/$\overline{\text{IRQ3}}$ | |
| | | PF2 | |
| | | PF1 | |
| | | PF0/$\overline{\text{IRQ2}}$ | |

## 9.1 Port 1

Port 1 is an 8-bit I/O port and has the following registers.

- Port 1 data direction register (P1DDR)
- Port 1 data register (P1DR)
- Port 1 register (PORT1)

### 9.1.1 Port 1 Data Direction Register (P1DDR)

P1DDR is an 8-bit write-only register, the individual bits of which specify input or output for the pins of port 1.

P1DDR cannot be read; if it is, an undefined value will be read.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P17DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 1 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | P16DDR | 0 | W | |
| 5 | P15DDR | 0 | W | |
| 4 | P14DDR | 0 | W | |
| 3 | P13DDR | 0 | W | |
| 2 | P12DDR | 0 | W | |
| 1 | P11DDR | 0 | W | |
| 0 | P10DDR | 0 | W | |

### 9.1.2 Port 1 Data Register (P1DR)

P1DR is an 8-bit readable/writable register that stores output data for port 1 pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P17DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 6 | P16DR | 0 | R/W | |
| 5 | P15DR | 0 | R/W | |
| 4 | P14DR | 0 | R/W | |
| 3 | P13DR | 0 | R/W | |
| 2 | P12DR | 0 | R/W | |
| 1 | P11DR | 0 | R/W | |
| 0 | P10DR | 0 | R/W | |

### 9.1.3 Port 1 Register (PORT1)

PORT1 is an 8-bit read-only register that shows the pin states.

PORT1 cannot be modified.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P17 | Undefined* | R | If a port 1 read is performed while P1DDR bits are set to 1, the P1DR values are read. If a port 1 read is performed while P1DDR bits are cleared to 0, the pin states are read. |
| 6 | P16 | Undefined* | R | |
| 5 | P15 | Undefined* | R | |
| 4 | P14 | Undefined* | R | |
| 3 | P13 | Undefined* | R | |
| 2 | P12 | Undefined* | R | |
| 1 | P11 | Undefined* | R | |
| 0 | P10 | Undefined* | R | |

Note:* Determined by the states of pins P17 to P10.

### 9.1.4 Pin Functions

Port 1 pins also function as TPU I/O pins, PPG output pins, and interrupt input pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.2 P17 Pin Function**

| TPU Channel 2 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P17DDR | — | 0 | 1 | 1 |
| NDER15 | — | — | 0 | 1 |
| Pin function | TIOCB2 output | P17 input | P17 output | PO15 output |
| | | TIOCB2 input | | |
| | | TCLKD input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.3 P16 Pin Function**

| TPU Channel 2 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P16DDR | — | 0 | 1 | 1 |
| NDER14 | — | — | 0 | 1 |
| Pin function | TIOCA2 output | P16 input | P16 output | PO14 output |
| | | TIOCA2 input | | |
| | $\overline{\text{IRQ1}}$ input | | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.4 P15 Pin Function**

| TPU Channel 1 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P15DDR | — | 0 | 1 | 1 |
| NDER13 | — | — | 0 | 1 |
| Pin function | TIOCB1 output | P15 input | P15 output | PO13 output |
| | | TIOCB1 input | | |
| | | TCLKC input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.5 P14 Pin Function**

| TPU Channel 1 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P14DDR | — | 0 | 1 | 1 |
| NDER12 | — | — | 0 | 1 |
| Pin function | TIOCA1 output | P14 input | P14 output | PO12 output |
| | | TIOCA1 input | | |
| | $\overline{\text{IRQ0}}$ input | | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.6 P13 Pin Function**

| TPU Channel 0 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P13DDR | — | 0 | 1 | 1 |
| NDER11 | — | — | 0 | 1 |
| Pin function | TIOCD0 output | P13 input | P13 output | PO11 output |
| | | TIOCD0 input | | |
| | | TCLKB input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.7 P12 Pin Function**

| TPU Channel 0 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P12DDR | — | 0 | 1 | 1 |
| NDER10 | — | — | 0 | 1 |
| Pin function | TIOCC0 output | P12 input | P12 output | PO10 output |
| | | TIOCC0 input | | |
| | | TCLKA input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.8 P11 Pin Function**

| TPU Channel 0 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P11DDR | — | 0 | 1 | 1 |
| NDER9 | — | — | 0 | 1 |
| Pin function | TIOCB0 output | P11 input | P11 output | PO9 output |
| | | TIOCB0 input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.9 P10 Pin Function**

| TPU Channel 0 Setting* | Output | Input or Initial Value | | |
|---|---|---|---|---|
| P10DDR | — | 0 | 1 | 1 |
| NDER8 | — | — | 0 | 1 |
| Pin function | TIOCA0 output | P10 input | P10 output | PO8 output |
| | | TIOCA0 input | | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

## 9.2 Port 3

Port 3 is an 8-bit I/O port and has the following registers.

- Port 3 data direction register (P3DDR)
- Port 3 data register (P3DR)
- Port 3 register (PORT3)
- Port 3 open-drain control register (P3ODR)

### 9.2.1 Port 3 Data Direction Register (P3DDR)

P3DDR is an 8-bit write-only register, the individual bits of which specify input or output for the pins of port 3.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P37DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 3 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | P36DDR | 0 | W | |
| 5 | P35DDR | 0 | W | |
| 4 | P34DDR | 0 | W | |
| 3 | P33DDR | 0 | W | |
| 2 | P32DDR | 0 | W | |
| 1 | P31DDR | 0 | W | |
| 0 | P30DDR | 0 | W | |

### 9.2.2 Port 3 Data Register (P3DR)

P3DR is an 8-bit readable/writable register that stores output data for port 3 pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P37DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general I/O port. |
| 6 | P36DR | 0 | R/W | |
| 5 | P35DR | 0 | R/W | |
| 4 | P34DR | 0 | R/W | |
| 3 | P33DR | 0 | R/W | |
| 2 | P32DR | 0 | R/W | |
| 1 | P31DR | 0 | R/W | |
| 0 | P30DR | 0 | R/W | |

### 9.2.3 Port 3 Register (PORT3)

PORT3 is an 8-bit read-only register that shows the pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P37 | Undefined* | R | If a port 3 read is performed while P3DDR bits are set to 1, the P3DR values are read. If a port 3 read is performed while P3DDR bits are cleared to 0, the pin states are read. |
| 6 | P36 | Undefined* | R | |
| 5 | P35 | Undefined* | R | |
| 4 | P34 | Undefined* | R | |
| 3 | P33 | Undefined* | R | |
| 2 | P32 | Undefined* | R | |
| 1 | P31 | Undefined* | R | |
| 0 | P30 | Undefined* | R | |

Note:* Determined by the states of pins P37 to P30.

### 9.2.4 Port 3 Open-Drain Control Register (P3ODR)

P3ODR is an 8-bit readable/writable register that specifies the output type of port 3.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P37ODR | 0 | R/W | When a pin is specified as an output port, setting the corresponding bit to 1 specifies pin output to open-drain and the input pull-up MOS to the off state. Clearing this bit to 0 specifies that to push-pull output. |
| 6 | P36ODR | 0 | R/W | |
| 5 | P35ODR | 0 | R/W | |
| 4 | P34ODR | 0 | R/W | |
| 3 | P33ODR | 0 | R/W | |
| 2 | P32ODR | 0 | R/W | |
| 1 | P31ODR | 0 | R/W | |
| 0 | P30ODR | 0 | R/W | |

Note:* Determined by the states of pins P47 to P40.

### 9.2.5 Pin Functions

Port 3 pins also function as SCI_0 I/O pins and interrupt input pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.10 P37 Pin Function**

| P37DDR | 0 | 1 |
|---|---|---|
| Pin function | P37 input | P37 output |

**Table 9.11 P36 Pin Function**

| P36DDR | 0 | 1 |
|---|---|---|
| Pin function | P36 input | P36 output |

**Table 9.12 P35 Pin Function**

| P35DDR | 0 | 1 |
|---|---|---|
| Pin function | P35 input | P35 output |
| | $\overline{\text{IRQ5}}$ input* | |

**Table 9.13 P34 Pin Function**

| P34DDR | 0 | 1 |
|---|---|---|
| Pin function | P34 input | P34 output |

**Table 9.14 P33 Pin Function**

| P33DDR | 0 | 1 |
|---|---|---|
| Pin function | P33 input | P33 output |

**Table 9.15 P32 Pin Function**

| CKE1 in SCR_0 | 0 | | | | 1 |
|---|---|---|---|---|---|
| C/$\overline{\text{A}}$ in SMR_0 | 0 | | | 1 | — |
| CKE0 in SCR_0 | 0 | | 1 | — | — |
| P32DDR | 0 | 1 | — | — | — |
| Pin function | P32 input | P32 output | SCK0 output | SCK0 output | SCK0 input |
| | $\overline{\text{IRQ4}}$ input* | | | | |

**Table 9.16 P31 Pin Function**

| RE in SCR_0 | 0 | | 1 |
|---|---|---|---|
| P31DDR | 0 | 1 | — |
| Pin function | P31 input | P31 output | RxD0 output |

**Table 9.17 P30 Pin Function**

| TE in SCR_0 | 0 | | 1 |
|---|---|---|---|
| P30DDR | 0 | 1 | — |
| Pin function | P30 input | P30 output | TxD0 output |

Note:* When used as an external interrupt input pin, do not use as an I/O pin for another function.

## 9.3 Port 4

Port 4 is an input-only port. Port 4 pins also function as A/D converter analog input pins. Port 4 has the following register.

- Port 4 register (PORT4)

### 9.3.1 Port 4 Register (PORT4)

PORT4 is an 8-bit read-only register that shows port 4 pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P47 | Undefined* | R | The pin states are always read when a port 4 read is performed. |
| 6 | P46 | Undefined* | R | |
| 5 | P45 | Undefined* | R | |
| 4 | P44 | Undefined* | R | |
| 3 | P43 | Undefined* | R | |
| 2 | P42 | Undefined* | R | |
| 1 | P41 | Undefined* | R | |
| 0 | P40 | Undefined* | R | |

Note:* Determined by the states of pins P47 to P40.

## 9.4 Port 7

Port 7 is an 8-bit I/O port and has the following registers.

- Port 7 data direction register (P7DDR)
- Port 7 data register (P7DR)
- Port 7 register (PORT7)

### 9.4.1 Port 7 Data Direction Register (P7DDR)

P7DDR is an 8-bit write-only register, the individual bits of which specify input or output for the pins of port 7.

P7DDR cannot be read, if it is, an undefined value will be read.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P77DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 7 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | P76DDR | 0 | W | |
| 5 | P75DDR | 0 | W | |
| 4 | P74DDR | 0 | W | |
| 3 | P73DDR | 0 | W | |
| 2 | P72DDR | 0 | W | |
| 1 | P71DDR | 0 | W | |
| 0 | P70DDR | 0 | W | |

### 9.4.2 Port 7 Data Register (P7DR)

P7DR is an 8-bit readable/writable register that stores output data for port 7 pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P77DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 6 | P76DR | 0 | R/W | |
| 5 | P75DR | 0 | R/W | |
| 4 | P74DR | 0 | R/W | |
| 3 | P73DR | 0 | R/W | |
| 2 | P72DR | 0 | R/W | |
| 1 | P71DR | 0 | R/W | |
| 0 | P70DR | 0 | R/W | |

### 9.4.3 Port 7 Register (PORT7)

PORT7 is an 8-bit read-only register that shows the pin states.

PORT7 cannot be modified.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P77 | Undefined* | R | If a port 7 read is performed while P7DDR bits are set to 1, the P7DR values are read. If a port 7 read is performed while P7DDR bits are cleared to 0, the pin states are read. |
| 6 | P76 | Undefined* | R | |
| 5 | P75 | Undefined* | R | |
| 4 | P74 | Undefined* | R | |
| 3 | P73 | Undefined* | R | |
| 2 | P72 | Undefined* | R | |
| 1 | P71 | Undefined* | R | |
| 0 | P70 | Undefined* | R | |

Note:* Determined by the states of pins P77 to P70.

### 9.4.4 Pin Functions

Port 7 pins also function as TMR_3, TMR_2, TMR_1, and TMR_0 I/O pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.18 P77 Pin Function**

| P77DDR | 0 | 1 |
|---|---|---|
| Pin function | P77 input | P77 output |

**Table 9.19 P76 Pin Function**

| P76DDR | 0 | 1 |
|---|---|---|
| Pin function | P76 input | P76 output |

**Table 9.20 P75 Pin Function**

| OSC3 to OSC0 in TCSR_3 | All 0 | | Any of 1 |
|---|---|---|---|
| P75DDR | 0 | 1 | — |
| Pin function | P75 input | P75 output | TMO3 output |

**Table 9.21 P74 Pin Function**

| OSC3 to OSC0 in TCSR_2 | All 0 | | Any of 1 |
|---|---|---|---|
| P74DDR | 0 | 1 | — |
| Pin function | P74 input | P74 output | TMO2 output |

**Table 9.22 P73 Pin Function**

| OSC3 to OSC0 in TCSR_1 | All 0 | | Any of 1 |
|---|---|---|---|
| P73DDR | 0 | 1 | — |
| Pin function | P73 input | P73 output | TMO1 output |

**Table 9.23 P72 Pin Function**

| OSC3 to OSC0 in TCSR_0 | All 0 | | Any of 1 |
|---|---|---|---|
| P72DDR | 0 | 1 | — |
| Pin function | P72 input | P72 output | TMO0 output |

**Table 9.24 P71 Pin Function**

| P71DDR | 0 | 1 |
|---|---|---|
| Pin function | P71 input | P71 output |
| | TMCI23 input/TMRI23 input | |

**Table 9.25 P70 Pin Function**

| P70DDR | 0 | 1 |
|---|---|---|
| Pin function | P70 input | P70 output |
| | TMCI01 input/TMRI01 input | |

## 9.5 Port 9

Port 9 is an input-only port. Port 9 pins also function as A/D converter analog input pins. Port 9 has the following register.

- Port 9 register (PORT9)

### 9.5.1 Port 9 Register (PORT9)

PORT9 is an 8-bit read-only register that shows port 9 pin states.

PORT9 cannot be modified.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | P97 | Undefined* | R | The pin states are always read when a port 9 read is performed. |
| 6 | P96 | Undefined* | R | |
| 5 | P95 | Undefined* | R | |
| 4 | P94 | Undefined* | R | |
| 3 | P93 | Undefined* | R | |
| 2 | P92 | Undefined* | R | |
| 1 | P91 | Undefined* | R | |
| 0 | P90 | Undefined* | R | |

Note:* Determined by the states of pins P97 to P90.

## 9.6 Port A

Port A is a 4-bit I/O port that also has other functions. Port A has the following registers.

- Port A data direction register (PADDR)
- Port A data register (PADR)
- Port A register (PORTA)
- Port A pull-up MOS control register (PAPCR)
- Port A open-drain control register (PAODR)

### 9.6.1 Port A Data Direction Register (PADDR)

PADDR is an 8-bit write-only register, the individual bits of which specify whether the pins of port A are used for input or output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | Undefined | — | Reserved<br>These bits are read as undefined value and cannot be modified. |
| 3 | PA3DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port A pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 2 | PA2DDR | 0 | W | |
| 1 | PA1DDR | 0 | W | |
| 0 | PA0DDR | 0 | W | |

### 9.6.2 Port A Data Register (PADR)

PADR is an 8-bit readable/writable register that stores output data for port A pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | Undefined | — | Reserved<br>These bits are read as an undefined value and cannot be modified. |
| 3 | PA3DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 2 | PA2DR | 0 | R/W | |
| 1 | PA1DR | 0 | R/W | |
| 0 | PA0DR | 0 | R/W | |

### 9.6.3 Port A Register (PORTA)

PORTA is an 8-bit read-only register that shows port A pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | Undefined | — | Reserved<br>These bits are read as an undefined value. |
| 3 | PA3 | 0 | R | If a port A read is performed while PADDR bits are set to 1, the PADR values are read. If a port A read is performed while PADDR bits are cleared to 0, the pin states are read. |
| 2 | PA2 | 0 | R | |
| 1 | PA1 | 0 | R | |
| 0 | PA0 | 0 | R | |

Note:* Determined by the states of pins PA3 to PA0.

### 9.6.4 Port A Pull-Up MOS Control Register (PAPCR)

PAPCR is an 8-bit register that controls the input pull-up MOS function.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | Undefined | — | Reserved<br>These bits are read as an undefined value and cannot be modified. |
| 3 | PA3PCR | 0 | R/W | When a pin is specified as an input port, setting the corresponding bit to 1 turns on the input pull-up MOS for that pin. |
| 2 | PA2PCR | 0 | R/W | |
| 1 | PA1PCR | 0 | R/W | |
| 0 | PA0PCR | 0 | R/W | |

### 9.6.5 Port A Open-Drain Control Register (PAODR)

PAODR is an 8-bit readable/writable register that specifies the output type of port A.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | Undefined | — | Reserved<br>These bits are read as an undefined value and cannot be modified. |
| 3 | PA3ODR | 0 | R/W | When a pin is specified as an output port, setting the corresponding bit to 1 specifies pin output to open-drain and the input pull-up MOS to the off state. Clearing this bit to 0 specifies that to push-pull output. |
| 2 | PA2ODR | 0 | R/W | |
| 1 | PA1ODR | 0 | R/W | |
| 0 | PA0ODR | 0 | R/W | |

### 9.6.6 Pin Functions

Port A pins also function as SCI_2 I/O pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.26 PA3 Pin Function**

| CKE1 in SCR_2 | 0 | | | | 1 |
|---|---|---|---|---|---|
| C/A in SMR_2 | 0 | | | 1 | — |
| CKE0 in SCR_2 | 0 | | 1 | — | — |
| PA3DDR | 0 | 1 | — | — | — |
| Pin function | PA3 input | PA3 output | SCK2 output | SCK2 output | SCK2 input |

**Table 9.27 PA2 Pin Function**

| RE in SCR_2 | 0 | | 1 |
|---|---|---|---|
| PA2DDR | 0 | 1 | — |
| Pin function | PA2 input | PA2 output | RxD2 input |

**Table 9.28 PA1 Pin Function**

| TE in SCR_2 | 0 | | 1 |
|---|---|---|---|
| PA1DDR | 0 | 1 | — |
| Pin function | PA1 input | PA1 output | TxD2 output |

**Table 9.29 PA0 Pin Function**

| PA0DDR | 0 | 1 |
|---|---|---|
| Pin function | PA0 input | PA0 output |

## 9.7 Port B

Port B is an 8-bit I/O port that also has other functions. Port B has the following registers.

- Port B data direction register (PBDDR)
- Port B data register (PBDR)
- Port B register (PORTB)
- Port B pull-up MOS control register (PBPCR)
- Port B open-drain control register (PBODR)

### 9.7.1 Port B Data Direction Register (PBDDR)

PBDDR is an 8-bit write-only register, the individual bits of which specify whether the pins of port B are used for input or output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PB7DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 1 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | PB6DDR | 0 | W | |
| 5 | PB5DDR | 0 | W | |
| 4 | PB4DDR | 0 | W | |
| 3 | PB3DDR | 0 | W | |
| 2 | PB2DDR | 0 | W | |
| 1 | PB1DDR | 0 | W | |
| 0 | PB0DDR | 0 | W | |

### 9.7.2 Port B Data Register (PBDR)

PBDR is an 8-bit readable/writable register that stores output data for the port B pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PB7DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 6 | PB6DR | 0 | R/W | |
| 5 | PB5DR | 0 | R/W | |
| 4 | PB4DR | 0 | R/W | |
| 3 | PB3DR | 0 | R/W | |
| 2 | PB2DR | 0 | R/W | |
| 1 | PB1DR | 0 | R/W | |
| 0 | PB0DR | 0 | R/W | |

### 9.7.3 Port B Register (PORTB)

PORTB is an 8-bit read-only register that shows port B pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PB7 | 0 | R | If a port B read is performed while PBDDR bits are set to 1, the PBDR values are read. If a port B read is performed while PBDDR bits are cleared to 0, the pin states are read. |
| 6 | PB6 | 0 | R | |
| 5 | PB5 | 0 | R | |
| 4 | PB4 | 0 | R | |
| 3 | PB3 | 0 | R | |
| 2 | PB2 | 0 | R | |
| 1 | PB1 | 0 | R | |
| 0 | PB0 | 0 | R | |

Note:* Determined by the states of pins PB7 to PB0.

### 9.7.4 Port B Pull-Up MOS Control Register (PBPCR)

PBPCR is an 8-bit readable/writable register that controls the on/off state of input pull-up MOS of port B.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PB7PCR | 0 | R/W | When a pin is specified as an input port, setting the corresponding bit to 1 turns on the input pull-up MOS for that pin. |
| 6 | PB6PCR | 0 | R/W | |
| 5 | PB5PCR | 0 | R/W | |
| 4 | PB4PCR | 0 | R/W | |
| 3 | PB3PCR | 0 | R/W | |
| 2 | PB2PCR | 0 | R/W | |
| 1 | PB1PCR | 0 | R/W | |
| 0 | PB0PCR | 0 | R/W | |

### 9.7.5 Port B Open-Drain Control Register (PBODR)

PBODR is an 8-bit readable/writable register that specifies the output type of port B.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PB7ODR | 0 | R/W | When a pin function is specified as an output port, setting the corresponding bit to 1 specifies pin output as open-drain and the input pull-up MOS to the off state. Clearing this bit to 0 specifies push-pull output. |
| 6 | PB6ODR | 0 | R/W | |
| 5 | PB5ODR | 0 | R/W | |
| 4 | PB4ODR | 0 | R/W | |
| 3 | PB3ODR | 0 | R/W | |
| 2 | PB2ODR | 0 | R/W | |
| 1 | PB1ODR | 0 | R/W | |
| 0 | PB0ODR | 0 | R/W | |

### 9.7.6 Pin Functions

Port B pins also function as TPU I/O pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.30 PB7 Pin Function**

| TPU channel 5 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB7DDR | — | 0 | 1 |
| Pin function | TIOCB5 output | PB7 input | PB7 output |
| | | TIOCB5 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.31 PB6 Pin Function**

| TPU channel 5 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB6DDR | — | 0 | 1 |
| Pin function | TIOCA5 output | PB6 input | PB6 output |
| | | TIOCA5 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.32 PB5 Pin Function**

| TPU channel 4 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB5DDR | — | 0 | 1 |
| Pin function | TIOCB4 output | PB5 input | PB5 output |
| | | TIOCB4 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.33 PB4 Pin Function**

| TPU channel 4 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB4DDR | — | 0 | 1 |
| Pin function | TIOCA4 output | PB4 input | PB4 output |
| | | TIOCA4 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.34 PB3 Pin Function**

| TPU channel 3 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB3DDR | — | 0 | 1 |
| Pin function | TIOCD3 output | PB3 input | PB3 output |
| | | TIOCD3 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.35 PB2 Pin Function**

| TPU channel 3 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB2DDR | — | 0 | 1 |
| Pin function | TIOCC3 output | PB2 input | PB2 output |
| | | TIOCC3 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.36 PB1 Pin Function**

| TPU channel 3 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB1DDR | — | 0 | 1 |
| Pin function | TIOCB3 output | PB1 input | PB1 output |
| | | TIOCB3 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

**Table 9.37 PB0 Pin Function**

| TPU channel 3 setting* | Output | Input or Initial Value | |
|---|---|---|---|
| PB0DDR | — | 0 | 1 |
| Pin function | TIOCA3 output | PB0 input | PB0 output |
| | | TIOCA3 input | |

Note:* For details on the TPU channel specification, refer to section 10, 16-Bit Timer Pulse Unit (TPU).

## 9.8 Port C

Port C is an 8-bit I/O port that also has other functions. Port C has the following registers.

- Port C data direction register (PCDDR)

- Port C data register (PCDR)
- Port C register (PORTC)
- Port C pull-up MOS control register (PCPCR)
- Port C open-drain control register (PCODR)

### 9.8.1 Port C Data Direction Register (PCDDR)

PCDDR is an 8-bit write-only register, the individual bits of which specify whether the pins of port C are used for input or output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PC7DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 1 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | PC6DDR | 0 | W | |
| 5 | PC5DDR | 0 | W | |
| 4 | PC4DDR | 0 | W | |
| 3 | PC3DDR | 0 | W | |
| 2 | PC2DDR | 0 | W | |
| 1 | PC1DDR | 0 | W | |
| 0 | PC0DDR | 0 | W | |

### 9.8.2 Port C Data Register (PCDR)

PCDR is an 8-bit readable/writable register that stores output data for the port C pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PC7DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 6 | PC6DR | 0 | R/W | |
| 5 | PC5DR | 0 | R/W | |
| 4 | PC4DR | 0 | R/W | |
| 3 | PC3DR | 0 | R/W | |
| 2 | PC2DR | 0 | R/W | |
| 1 | PC1DR | 0 | R/W | |
| 0 | PC0DR | 0 | R/W | |

### 9.8.3 Port C Register (PORTC)

PORTC is an 8-bit read-only register that shows port C pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PC7 | 0 | R | If a port C read is performed while PCDDR bits are set to 1, the PCDR values are read. If a port C read is performed while PCDDR bits are cleared to 0, the pin states are read. |
| 6 | PC6 | 0 | R | |
| 5 | PC5 | 0 | R | |
| 4 | PC4 | 0 | R | |
| 3 | PC3 | 0 | R | |
| 2 | PC2 | 0 | R | |
| 1 | PC1 | 0 | R | |
| 0 | PC0 | 0 | R | |

Note:* Determined by the states of pins PC7 to PC0.

### 9.8.4 Port C Pull-Up MOS Control Register (PCPCR)

PCPCR is an 8-bit readable/writable register that controls the on/off state of input pull-up MOS of port C.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PC7PCR | 0 | R/W | When a pin is specified as an input port, setting the corresponding bit to 1 turns on the input pull-up MOS for that pin. |
| 6 | PC6PCR | 0 | R/W | |
| 5 | PC5PCR | 0 | R/W | |
| 4 | PC4PCR | 0 | R/W | |
| 3 | PC3PCR | 0 | R/W | |
| 2 | PC2PCR | 0 | R/W | |
| 1 | PC1PCR | 0 | R/W | |
| 0 | PC0PCR | 0 | R/W | |

### 9.8.5 Port C Open-Drain Control Register (PCODR)

PCODR is an 8-bit readable/writable register that specifies an output type of port C.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PC7ODR | 0 | R/W | When a pin is specified as an output port, setting the corresponding bit to 1 specifies pin output as open-drain and the input pull-up MOS to the off state. Clearing this bit to 0 specifies push-pull output. |
| 6 | PC6ODR | 0 | R/W | |
| 5 | PC5ODR | 0 | R/W | |
| 4 | PC4ODR | 0 | R/W | |
| 3 | PC3ODR | 0 | R/W | |
| 2 | PC2ODR | 0 | R/W | |
| 1 | PC1ODR | 0 | R/W | |
| 0 | PC0ODR | 0 | R/W | |

### 9.8.6 Pin Functions

Port C pins also function as SSU_1 and SSU_0 I/O pins. The correspondence between the register specification and the pin functions is shown below.

**Table 9.38 PC7 Pin Function**

| CSS1 | 0 | | | 1 | |
|---|---|---|---|---|---|
| CSS0 | 0 | | 1 | 0 | 1 |
| PC7DDR | 0 | 1 | — | — | — |
| Pin function | PC7 input | PC7 output | $\overline{\text{SCS1}}$ input | $\overline{\text{SCS1}}$ input/output auto switch | Setting prohibited |

**Table 9.39 PC6 Pin Function**

| MSS | 0 | | | 1 | |
|---|---|---|---|---|---|
| SCKS | 0 | | 1 | 1 | 0 |
| PC6DDR | 0 | 1 | — | — | — |
| Pin function | PC6 input | PC6 output | SSCK1 input | SSCK1 output | Setting prohibited |

**Table 9.40 PC5 Pin Function**

| MSS | 0 | | | | | 1 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| BIDE | 0 | | 1 | | | 0 | | | 1 | |
| RE | — | | | | | 0 | | 1 | — | |
| TE | 0 | | 1 | — | | — | | | — | |
| PC5DDR | 0 | 1 | — | 0 | 1 | 0 | 1 | — | 0 | 1 |
| Pin function | PC5 input | PC5 output | SSI1 output | PC5 input | PC5 output | PC5 input | PC5 output | SSI1 input | PC5 input | PC5 output |

**Table 9.41 PC4 Pin Function**

| MSS | 0 | | | 1 | | | — | | | | 0 | | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BIDE | 0 | | | | | | 1 | | | | | | |
| RE | 0 | | 1 | — | | | 0 | | 1 | | 0 | | |
| TE | — | | | 0 | | 1 | 0 | | 0 | 1 | 1 | | |
| PC4DDR | 0 | 1 | — | 0 | 1 | — | 0 | 1 | — | | | | |
| $\overline{SCS}$I input | — | | | | | | | | | | 0 | 1 | — |
| Pin function | PC4 input | PC4 output | SSO1 input | PC4 input | PC4 output | SSO1 output | PC4 input | PC4 output | SSO1 input | Setting prohibited | SSO1 output | SSO1 Hi-z | SSO1 output |

**Table 9.42 PC3 Pin Function**

| CSS1 | 0 | | | 1 | |
|---|---|---|---|---|---|
| CSS0 | 0 | | 1 | 0 | 1 |
| PC3DDR | 0 | 1 | — | — | — |
| Pin function | PC3 input | PC3 output | $\overline{SCS0}$ input | $\overline{SCS0}$ input/output auto switch | Setting prohibited |

**Table 9.43 PC2 Pin Function**

| MSS | 0 | | | 1 | |
|---|---|---|---|---|---|
| SCKS | 0 | | 1 | 1 | 0 |
| PC2DDR | 0 | 1 | — | — | — |
| Pin function | PC2 input | PC2 output | SSCK0 input | SSCK0 output | Setting prohibited |

**Table 9.44 PC1 Pin Function**

| MSS | 0 | | | | | 1 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| BIDE | 0 | | | 1 | | 0 | | | 1 | |
| RE | — | | | | | 0 | | 1 | — | |
| TE | 0 | | 1 | — | | — | | | — | |
| PC1DDR | 0 | 1 | — | 0 | 1 | 0 | 1 | — | 0 | 1 |
| Pin function | PC1 input | PC1 output | SSI0 output | PC1 input | PC1 output | PC1 input | PC1 output | SSI0 input | PC1 input | PC1 output |

**Table 9.45 PC 0Pin Function**

| MSS | 0 | | | 1 | | | — | | | | 0 | | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BIDE | 0 | | | | | | 1 | | | | | | |
| RE | 0 | | 1 | — | | | 0 | | 1 | | 0 | | |
| TE | — | | | 0 | | 1 | 0 | | 0 | 1 | 1 | | |
| PC0DDR | 0 | 1 | — | 0 | 1 | — | 0 | 1 | — | | | | |
| | — | | | | | | | | | | 0 | 1 | — |
| Pin function | PC0 in-put | PC0 out-put | SSO0 in-put | PC0 in-put | PC0 out-put | SSO0 out-put | PC0 in-put | PC0 out-put | SSO0 input | Setting pro-hibited | SSO0 out-put | SSO0 Hi-z | SSO0 out-put |

## 9.9 Port D

Port D is an 8-bit I/O port that also functions as the realtime input port pins.

The realtime input port stores the pin states of port D in PDRTIDR using the $\overline{\text{IRQ3}}$ pin as the trigger input. The falling, rising, or both edges of the $\overline{\text{IRQ3}}$ pin can be used as a trigger timing.

Port D has the following registers.

- Port D data direction register (PDDDR)
- Port D data register (PDDR)
- Port D register (PORTD)
- Port D pull-up MOS control register (PDPCR)
- Port D realtime input data register (PDRTIDR)

### 9.9.1 Port D Data Direction Register (PDDDR)

PDDDR is an 8-bit write-only register, the individual bits of which specify whether the pins of port D are used for input or output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PD7DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port 1 pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | PD6DDR | 0 | W | |
| 5 | PD5DDR | 0 | W | |
| 4 | PD4DDR | 0 | W | |
| 3 | PD3DDR | 0 | W | |
| 2 | PD2DDR | 0 | W | |
| 1 | PD1DDR | 0 | W | |
| 0 | PD0DDR | 0 | W | |

### 9.9.2 Port D Data Register (PDDR)

PDDR is an 8-bit readable/writable register that stores output data for the port D pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PD7DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 6 | PD6DR | 0 | R/W | |
| 5 | PD5DR | 0 | R/W | |
| 4 | PD4DR | 0 | R/W | |
| 3 | PD3DR | 0 | R/W | |
| 2 | PD2DR | 0 | R/W | |
| 1 | PD1DR | 0 | R/W | |
| 0 | PD0DR | 0 | R/W | |

### 9.9.3 Port D Register (PORTD)

PORTD is an 8-bit read-only register that shows port D pin states.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PD7 | Undefined* | R | If a port D read is performed while PDDDR bits are set to 1, the PDDR values are read. If a port D read is performed while PDDDR bits are cleared to 0, the pin states are read. |
| 6 | PD6 | Undefined* | R | |
| 5 | PD5 | Undefined* | R | |
| 4 | PD4 | Undefined* | R | |
| 3 | PD3 | Undefined* | R | |
| 2 | PD2 | Undefined* | R | |
| 1 | PD1 | Undefined* | R | |
| 0 | PD0 | Undefined* | R | |

Note:* Determined by the states of pins PD7 to PD0.

### 9.9.4 Port D Pull-up MOS Control Register (PDPCR)

PDPCR is an 8-bit readable/writable register that controls on/off states of the input pull-up MOS of port D.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PD7PCR | 0 | R/W | When the pin is in its input state, the input pull-up MOS of the input pin is on when the corresponding bit is set to 1. |
| 6 | PD6PCR | 0 | R/W | |
| 5 | PD5PCR | 0 | R/W | |
| 4 | PD4PCR | 0 | R/W | |
| 3 | PD3PCR | 0 | R/W | |
| 2 | PD2PCR | 0 | R/W | |
| 1 | PD1PCR | 0 | R/W | |
| 0 | PD0PCR | 0 | R/W | |

### 9.9.5 Port D RealTime Input Data Register (PDRTIDR)

The realtime input port stores the pin states of port D in PDRTIDR using the $\overline{\text{IRQ3}}$ pin as the trigger input. The falling, rising, or both edges of the $\overline{\text{IRQ3}}$ pin can be specified as a trigger timing by bits 7 and 6 in the IRQ sense control register L (ISCRL). For details of this setting, see section 5.3.3, IRQ Sense Control Registers H and L (ISCRH, ISCRL).

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PDRTIDR7 | 0 | R/W | Stores pin states using the $\overline{\text{IRQ3}}$ pin as a trigger input. |
| 6 | PDRTIDR6 | 0 | R/W | |
| 5 | PDRTIDR5 | 0 | R/W | |
| 4 | PDRTIDR4 | 0 | R/W | |
| 3 | PDRTIDR3 | 0 | R/W | |
| 2 | PDRTIDR2 | 0 | R/W | |
| 1 | PDRTIDR1 | 0 | R/W | |
| 0 | PDRTIDR0 | 0 | R/W | |

## 9.10 Port F

Port F is an 8-bit I/O port that also has other functions. Port F has the following registers.

- Port F data direction register (PFDDR)
- Port F data register (PFDR)
- Port F register (PORTF)

### 9.10.1 Port F Data Direction Register (PFDDR)

PFDDR is an 8-bit write-only register, the individual bits of which specify whether the pins of port F are used for input or output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PF7DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the PF7 pin a $\phi$ output pin. Clearing this bit to 0 makes the pin an input pin. |
| 6 | PF6DDR | 0 | W | When a pin is specified as a general purpose I/O port, setting this bit to 1 makes the corresponding port F pin an output pin. Clearing this bit to 0 makes the pin an input pin. |
| 5 | PF5DDR | 0 | W | |
| 4 | PF4DDR | 0 | W | |
| 3 | PF3DDR | 0 | W | |
| 2 | PF2DDR | 0 | W | |
| 1 | PF1DDR | 0 | W | |
| 0 | PF0DDR | 0 | W | |

### 9.10.2 Port F Data Register (PFDR)

PFDR is an 8-bit readable/writable register that stores output data for the port F pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | R/W | Reserved<br>The write value should always be 0. |
| 6 | PF6DR | 0 | R/W | Output data for a pin is stored when the pin is specified as a general purpose I/O port. |
| 5 | PF5DR | 0 | R/W | |
| 4 | PF4DR | 0 | R/W | |
| 3 | PF3DR | 0 | R/W | |
| 2 | PF2DR | 0 | R/W | |
| 1 | PF1DR | 0 | R/W | |
| 0 | PF0DR | 0 | R/W | |

### 9.10.3 Port F Register (PORTF)

PORTF is an 8-bit read-only register that shows port F pin states.

PORTF cannot be modified.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PF7 | Undefined* | R | If a port F read is performed while PFDDR bits are set to 1, the PFDR values are read. If a port F read is performed while PFDDR bits are cleared to 0, the pin states are read. |
| 6 | PF6 | Undefined* | R | |
| 5 | PF5 | Undefined* | R | |
| 4 | PF4 | Undefined* | R | |
| 3 | PF3 | Undefined* | R | |
| 2 | PF2 | Undefined* | R | |
| 1 | PF1 | Undefined* | R | |
| 0 | PF0 | Undefined* | R | |

Note:* Determined by the states of pins PF7 to PF0.

### 9.10.4 Pin Functions

Port F is an 8-bit I/O port. Port F pins also function as external interrupt input, $\overline{\text{IRQ2}}$ and $\overline{\text{IRQ3}}$, A/D trigger input ($\overline{\text{ADTRG}}$), and system clock output ($\phi$).

**Table 9.46 PF7 Pin Function**

| PF7DDR | 0 | 1 |
|---|---|---|
| Pin function | PF7 input | $\phi$ output |

**Table 9.47 PF6 Pin Function**

| PF6DDR | 0 | 1 |
|---|---|---|
| Pin function | PF6 input | PF6 output |

**Table 9.48 PF5 Pin Function**

| PF5DDR | 0 | 1 |
|---|---|---|
| Pin function | PF5 input | PF5 output |

**Table 9.49 PF4 Pin Function**

| PF4DDR | 0 | 1 |
|---|---|---|
| Pin function | PF4 input | PF4 output |

**Table 9.50 PF3 Pin Function**

| PF3DDR | 0 | 1 |
|---|---|---|
| Pin function | PF3 input | PF3 output |
| | $\overline{\text{ADTRG}}$ input*[1] | |
| | $\overline{\text{IRQ3}}$ input*[2] | |

Notes: 1. ADTRG input when TRGS0 = TRGS1 = 1.
2. When used as an external interrupt input pin, do not use as an I/O pin for another function. This pin also functions as the trigger input for the realtime input port.

**Table 9.51 PF2 Pin Function**

| PF2DDR | 0 | 1 |
|---|---|---|
| Pin function | PF2 input | PF2 output |

**Table 9.52 PF1 Pin Function**

| PF1DDR | 0 | 1 |
|---|---|---|
| Pin function | PF1 input | PF1 output |

**Table 9.53 PF0 Pin Function**

| PF0DDR | 0 | 1 |
|---|---|---|
| Pin function | PF0 input | PF0 output |
| | $\overline{\text{IRQ2}}$ input* | |

Note:* When used as an external interrupt input pin, do not use as an I/O pin for another function.

# Section 10 16-Bit Timer Pulse Unit (TPU)

This LSI has an on-chip 16-bit timer pulse unit (TPU) comprised of six 16-bit timer channels.

The function list of the 16-bit timer unit and its block diagram are shown in table 10.1 and figure 10.1, respectively.

## 10.1 Features

- Maximum 16-pulse input/output
- Selection of 8 counter input clocks for each channel
- The following operations can be set for each channel:
  - — Waveform output at compare match
  - — Input capture function
  - — Counter clear operation
  - — Synchronous operation:
    Multiple timer counters (TCNT) can be written to simultaneously
    Simultaneous clearing by compare match and input capture is possible
    Register simultaneous input/output is possible by synchronous counter operation
  - — A maximum 15-phase PWM output is possible in combination with synchronous operation
- Buffer operation settable for channels 0 and 3
- Phase counting mode settable independently for each of channels 1, 2, 4, and 5
- Cascaded operation
- Fast access via internal 16-bit bus
- 26 interrupt sources
- Automatic transfer of register data
- Programmable pulse generator (PPG) output trigger can be generated
- A/D converter conversion start trigger can be generated
- Module stop mode can be set

**Table 10.1 TPU Functions**

| Item | | Channel 0 | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 |
|---|---|---|---|---|---|---|---|
| Count clock | | φ/1<br>φ/4<br>φ/16<br>φ/64<br>TCLKA<br>TCLKB<br>TCLKC<br>TCLKD | φ/1<br>φ/4<br>φ/16<br>φ/64<br>φ/256<br>TCLKA<br>TCLKB | φ/1<br>φ/4<br>φ/16<br>φ/64<br>φ/1024<br>TCLKA<br>TCLKB<br>TCLKC | φ/1<br>φ/4<br>φ/16<br>φ/64<br>φ/256<br>φ/1024<br>φ/4096<br>TCLKA | φ/1<br>φ/4<br>φ/16<br>φ/64<br>φ/1024<br>TCLKA<br>TCLKC | φ/1<br>φ/4<br>φ/16<br>φ/64<br>φ/256<br>TCLKA<br>TCLKC<br>TCLKD |
| General registers (TGR) | | TGRA_0<br>TGRB_0 | TGRA_1<br>TGRB_1 | TGRA_2<br>TGRB_2 | TGRA_3<br>TGRB_3 | TGRA_4<br>TGRB_4 | TGRA_5<br>TGRB_5 |
| General registers/ buffer registers | | TGRC_0<br>TGRD_0 | — | — | TGRC_3<br>TGRD_3 | — | — |
| I/O pins | | TIOCA0<br>TIOCB0<br>TIOCC0<br>TIOCD0 | TIOCA1<br>TIOCB1 | TIOCA2<br>TIOCB2 | TIOCA3<br>TIOCB3<br>TIOCC3<br>TIOCD3 | TIOCA4<br>TIOCB4 | TIOCA5<br>TIOCB5 |
| Counter clear function | | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture |
| Compare match output | 0 output | ○ | ○ | ○ | ○ | ○ | ○ |
| | 1 output | ○ | ○ | ○ | ○ | ○ | ○ |
| | Toggle output | ○ | ○ | ○ | ○ | ○ | ○ |
| Input capture function | | ○ | ○ | ○ | ○ | ○ | ○ |
| Synchronous operation | | ○ | ○ | ○ | ○ | ○ | ○ |
| PWM mode | | ○ | ○ | ○ | ○ | ○ | ○ |
| Phase counting mode | | — | ○ | ○ | — | ○ | ○ |
| Buffer operation | | ○ | — | — | ○ | — | — |

| Item | Channel 0 | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 |
|---|---|---|---|---|---|---|
| DTC activation | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture |
| A/D converter trigger | TGRA_0 compare match or input capture | TGRA_1 compare match or input capture | TGRA_2 compare match or input capture | TGRA_3 compare match or input capture | TGRA_4 compare match or input capture | TGRA_5 compare match or input capture |
| PPG trigger | TGRA_0/ TGRB_0 compare match or input capture | TGRA_1/ TGRB_1 compare match or input capture | TGRA_2/ TGRB_2 compare match or input capture | TGRA_3/ TGRB_3 compare match or input capture | — | — |
| Interrupt sources | 5 sources<br>• Compare match or input capture 0A<br>• Compare match or input capture 0B<br>• Compare match or input capture 0C<br>• Compare match or input capture 0D<br>• Overflow | 4 sources<br>• Compare match or input capture 1A<br>• Compare match or input capture 1B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 2A<br>• Compare match or input capture 2B<br>• Overflow<br>• Underflow | 5 sources<br>• Compare match or input capture 3A<br>• Compare match or input capture 3B<br>• Compare match or input capture 3C<br>• Compare match or input capture 3D<br>• Overflow | 4 sources<br>• Compare match or input capture 4A<br>• Compare match or input capture 4B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 5A<br>• Compare match or input capture 5B<br>• Overflow<br>• Underflow |

**Legend**

◯ : Possible

— : Not possible

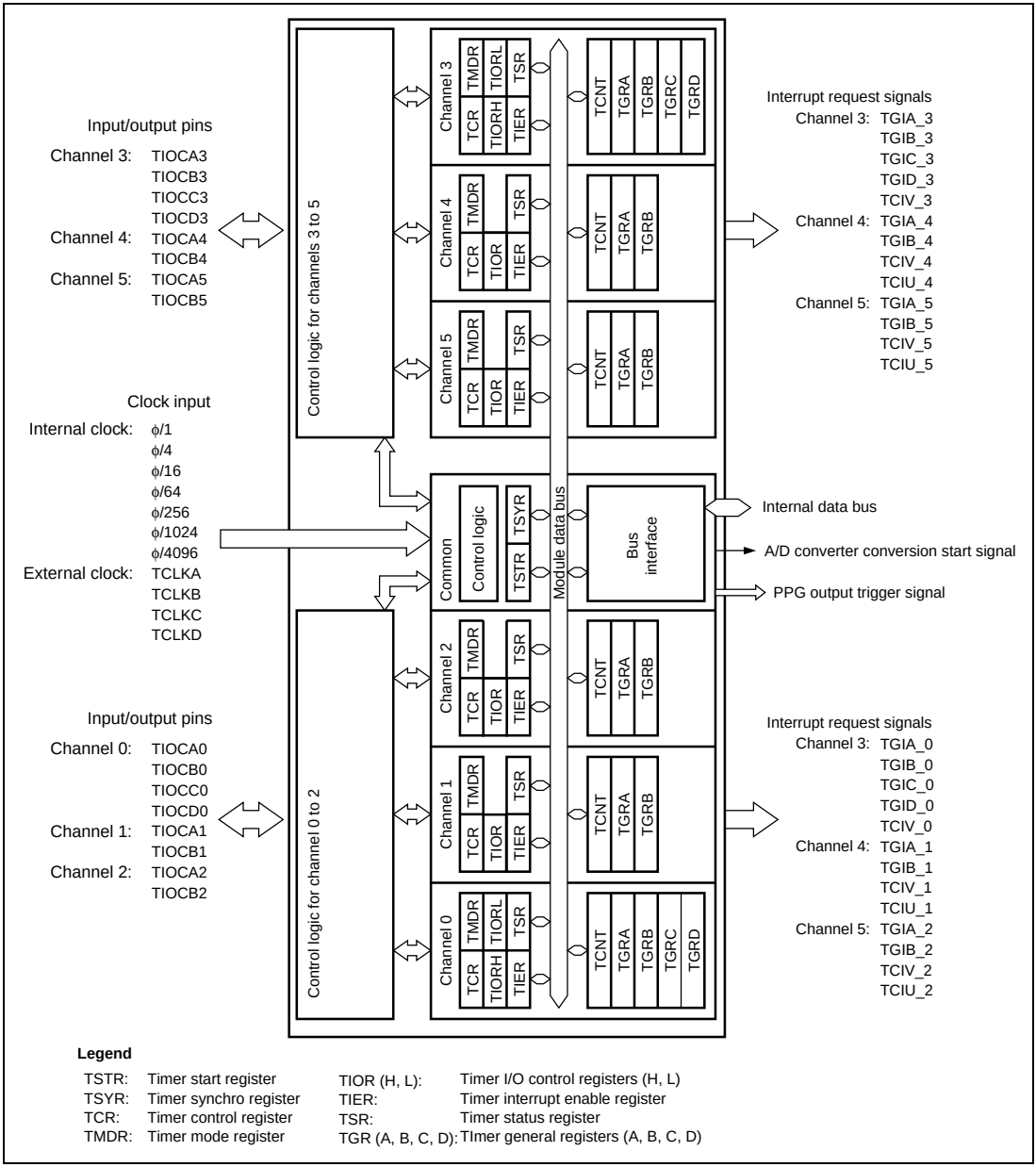



**Figure 10.1 Block Diagram of TPU**

## 10.2 Input/Output Pins

**Table 10.2 TPU Pins**

| Channel | Symbol | I/O | Function |
|---|---|---|---|
| All | TCLKA | Input | External clock A input pin<br>(Channel 1 and 5 phase counting mode A phase input) |
| | TCLKB | Input | External clock B input pin<br>(Channel 1 and 5 phase counting mode B phase input) |
| | TCLKC | Input | External clock C input pin<br>(Channel 2 and 4 phase counting mode A phase input) |
| | TCLKD | Input | External clock D input pin<br>(Channel 2 and 4 phase counting mode B phase input) |
| 0 | TIOCA0 | I/O | TGRA_0 input capture input/output compare output/PWM output pin |
| | TIOCB0 | I/O | TGRB_0 input capture input/output compare output/PWM output pin |
| | TIOCC0 | I/O | TGRC_0 input capture input/output compare output/PWM output pin |
| | TIOCD0 | I/O | TGRD_0 input capture input/output compare output/PWM output pin |
| 1 | TIOCA1 | I/O | TGRA_1 input capture input/output compare output/PWM output pin |
| | TIOCB1 | I/O | TGRB_1 input capture input/output compare output/PWM output pin |
| 2 | TIOCA2 | I/O | TGRA_2 input capture input/output compare output/PWM output pin |
| | TIOCB2 | I/O | TGRB_2 input capture input/output compare output/PWM output pin |
| 3 | TIOCA3 | I/O | TGRA_3 input capture input/output compare output/PWM output pin |
| | TIOCB3 | I/O | TGRB_3 input capture input/output compare output/PWM output pin |
| | TIOCC3 | I/O | TGRC_3 input capture input/output compare output/PWM output pin |
| | TIOCD3 | I/O | TGRD_3 input capture input/output compare output/PWM output pin |
| 4 | TIOCA4 | I/O | TGRA_4 input capture input/output compare output/PWM output pin |
| | TIOCB4 | I/O | TGRB_4 input capture input/output compare output/PWM output pin |
| 5 | TIOCA5 | I/O | TGRA_5 input capture input/output compare output/PWM output pin |
| | TIOCB5 | I/O | TGRB_5 input capture input/output compare output/PWM output pin |

## 10.3 Register Descriptions

The TPU has the following registers. To distinguish registers in each channel, an underscore and the channel number are added as a suffix to the register name; TCR for channel 0 is expressed as TCR_0.

- Timer control register_0 (TCR_0)
- Timer mode register_0 (TMDR_0)
- Timer I/O control register H_0 (TIORH_0)
- Timer I/O control register L_0 (TIORL_0)
- Timer interrupt enable register_0 (TIER_0)
- Timer status register_0 (TSR_0)
- Timer counter_0 (TCNT_0)
- Timer general register A_0 (TGRA_0)
- Timer general register B_0 (TGRB_0)
- Timer general register C_0 (TGRC_0)
- Timer general register D_0 (TGRD_0)
- Timer control register_1 (TCR_1)
- Timer mode register_1 (TMDR_1)
- Timer I/O control register _1 (TIOR_1)
- Timer interrupt enable register_1 (TIER_1)
- Timer status register_1 (TSR_1)
- Timer counter_1 (TCNT_1)
- Timer general register A_1 (TGRA_1)
- Timer general register B_1 (TGRB_1)
- Timer control register_2 (TCR_2)
- Timer mode register_2 (TMDR_2)
- Timer I/O control register_2 (TIOR_2)
- Timer interrupt enable register_2 (TIER_2)
- Timer status register_2 (TSR_2)
- Timer counter_2 (TCNT_2)
- Timer general register A_2 (TGRA_2)
- Timer general register B_2 (TGRB_2)
- Timer control register_3 (TCR_3)
- Timer mode register_3 (TMDR_3)
- Timer I/O control register H_3 (TIORH_3)
- Timer I/O control register L_3 (TIORL_3)
- Timer interrupt enable register_3 (TIER_3)

- Timer status register_3 (TSR_3)
- Timer counter_3 (TCNT_3)
- Timer general register A_3 (TGRA_3)
- Timer general register B_3 (TGRB_3)
- Timer general register C_3 (TGRC_3)
- Timer general register D_3 (TGRD_3)
- Timer control register_4 (TCR_4)
- Timer mode register_4 (TMDR_4)
- Timer I/O control register _4 (TIOR_4)
- Timer interrupt enable register_4 (TIER_4)
- Timer status register_4 (TSR_4)
- Timer counter_4 (TCNT_4)
- Timer general register A_4 (TGRA_4)
- Timer general register B_4 (TGRB_4)
- Timer control register_5 (TCR_5)
- Timer mode register_5 (TMDR_5)
- Timer I/O control register_5 (TIOR_5)
- Timer interrupt enable register_5 (TIER_5)
- Timer status register_5 (TSR_5)
- Timer counter_5 (TCNT_5)
- Timer general register A_5 (TGRA_5)
- Timer general register B_5 (TGRB_5)

Common Registers

- Timer start register (TSTR)
- Timer synchro register (TSYR)

### 10.3.1 Timer Control Register (TCR)

The TCR registers are 8-bit readable/writable registers that control the TCNT operation for each channel. The TPU has a total of six TCR registers, one for each channel (channel 0 to 5). TCR register settings should be conducted only when TCNT operation is stopped.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7<br>6<br>5 | CCLR2<br>CCLR1<br>CCLR0 | 0<br>0<br>0 | R/W<br>R/W<br>R/W | Counter Clear 0 to 2<br>These bits select the TCNT counter clearing source. See tables 10.3 and 10.4 for details. |
| 4<br>3 | CKEG1<br>CKEG0 | 0<br>0 | R/W<br>R/W | Clock Edge 0 and 1<br>These bits select the input clock edge. When the input clock is counted using both edges, the input clock period is halved (e.g. $\phi$/4 both edges = $\phi$/2 rising edge). If phase counting mode is used on channels 1, 2, 4, and 5, this setting is ignored and the phase counting mode setting has priority. Internal clock edge selection is valid when the input clock is $\phi$/4 or slower. This setting is ignored if the input clock is $\phi$/1, or when overflow/underflow of another channel is selected.<br>00: Count at rising edge<br>01: Count at falling edge<br>1X: Count at both edges<br>**Legend** X: Don't care |
| 2<br>1<br>0 | TPSC2<br>TPSC1<br>TPSC0 | 0<br>0<br>0 | R/W<br>R/W<br>R/W | Time Prescaler 0 to 2<br>These bits select the TCNT counter clock. The clock source can be selected independently for each channel. See tables10.5 to10.10 for details. |

**Table 10.3 CCLR0 to CCLR2 (Channels 0 and 3)**

| Channel | Bit 7 CCLR2 | Bit 6 CCLR1 | Bit 5 CCLR0 | Description |
|---|---|---|---|---|
| 0, 3 | 0 | 0 | 0 | TCNT clearing disabled |
| | | | 1 | TCNT cleared by TGRA compare match/input capture |
| | | 1 | 0 | TCNT cleared by TGRB compare match/input capture |
| | | | 1 | TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*[1] |
| | 1 | 0 | 0 | TCNT clearing disabled |
| | | | 1 | TCNT cleared by TGRC compare match/input capture*[2] |
| | | 1 | 0 | TCNT cleared by TGRD compare match/input capture*[2] |
| | | | 1 | TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*[1] |

Notes: 1. Synchronous operation is set by setting the SYNC bit in TSYR to 1.
2. When TGRC or TGRD is used as a buffer register, TCNT is not cleared because the buffer register setting has priority, and compare match/input capture does not occur.

**Table 10.4 CCLR0 to CCLR2 (Channels 1, 2, 4, and 5)**

| Channel | Bit 7 Reserved*[2] | Bit 6 CCLR1 | Bit 5 CCLR0 | Description |
|---|---|---|---|---|
| 1, 2, 4, 5 | 0 | 0 | 0 | TCNT clearing disabled |
| | | | 1 | TCNT cleared by TGRA compare match/input capture |
| | | 1 | 0 | TCNT cleared by TGRB compare match/input capture |
| | | | 1 | TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*[1] |

Notes: 1. Synchronous operation is selected by setting the SYNC bit in TSYR to 1.
2. Bit 7 is reserved in channels 1, 2, 4, and 5. It is always read as 0 and cannot be modified.

**Table 10.5 TPSC0 to TPSC2 (Channel 0)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | External clock: counts on TCLKB pin input |
| | | 1 | 0 | External clock: counts on TCLKC pin input |
| | | | 1 | External clock: counts on TCLKD pin input |

**Table 10.6 TPSC0 to TPSC2 (Channel 1)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | External clock: counts on TCLKB pin input |
| | | 1 | 0 | Internal clock: counts on $\phi$/256 |
| | | | 1 | Counts on TCNT2 overflow/underflow |

Note: This setting is ignored when channel 1 is in phase counting mode.

**Table 10.7 TPSC0 to TPSC2 (Channel 2)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 2 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | External clock: counts on TCLKB pin input |
| | | 1 | 0 | External clock: counts on TCLKC pin input |
| | | | 1 | Internal clock: counts on $\phi$/1024 |

Note: This setting is ignored when channel 2 is in phase counting mode.

**Table 10.8 TPSC0 to TPSC2 (Channel 3)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 3 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | Internal clock: counts on $\phi$/1024 |
| | | 1 | 0 | Internal clock: counts on $\phi$/256 |
| | | | 1 | Internal clock: counts on $\phi$/4096 |

**Table 10.9 TPSC0 to TPSC2 (Channel 4)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 4 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | External clock: counts on TCLKC pin input |
| | | 1 | 0 | Internal clock: counts on $\phi$/1024 |
| | | | 1 | Counts on TCNT5 overflow/underflow |

Note: This setting is ignored when channel 4 is in phase counting mode.

**Table 10.10 TPSC0 to TPSC2 (Channel 5)**

| Channel | Bit 2 TPSC2 | Bit 1 TPSC1 | Bit 0 TPSC0 | Description |
|---|---|---|---|---|
| 5 | 0 | 0 | 0 | Internal clock: counts on $\phi$/1 |
| | | | 1 | Internal clock: counts on $\phi$/4 |
| | | 1 | 0 | Internal clock: counts on $\phi$/16 |
| | | | 1 | Internal clock: counts on $\phi$/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA pin input |
| | | | 1 | External clock: counts on TCLKC pin input |
| | | 1 | 0 | Internal clock: counts on $\phi$/256 |
| | | | 1 | External clock: counts on TCLKD pin input |

Note: This setting is ignored when channel 5 is in phase counting mode.

### 10.3.2 Timer Mode Register (TMDR)

The TMDR registers are 8-bit readable/writable registers that are used to set the operating mode of each channel. The TPU has six TMDR registers, one for each channel. TMDR register settings should be changed only when TCNT operation is stopped.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |
| 5 | BFB | 0 | R/W | Buffer Operation B<br>Specifies whether TGRB is to operate in the normal way, or TGRB and TGRD are to be used together for buffer operation. When TGRD is used as a buffer register, TGRD input capture/output compare is not generated.<br>In channels 1, 2, 4, and 5, which have no TGRD, bit 5 is reserved. It is always read as 0 and cannot be modified.<br>0: TGRB operates normally<br>1: TGRB and TGRD used together for buffer operation |
| 4 | BFA | 0 | R/W | Buffer Operation A<br>Specifies whether TGRA is to operate in the normal way, or TGRA and TGRC are to be used together for buffer operation. When TGRC is used as a buffer register, TGRC input capture/output compare is not generated.<br>In channels 1, 2, 4, and 5, which have no TGRC, bit 4 is reserved. It is always read as 0 and cannot be modified.<br>0: TGRA operates normally<br>1:TGRA and TGRC used together for buffer operation |
| 3<br>2<br>1<br>0 | MD3<br>MD2<br>MD1<br>MD0 | 0<br>0<br>0<br>0 | R/W<br>R/W<br>R/W<br>R/W | Modes 0 to 3<br>These bits are used to set the timer operating mode.<br>MD3 is a reserved bit. In a write, it should always be written with 0. See table 10.11 for details. |

**Table 10.11 MD0 to MD3**

| Bit 3<br>MD3*[1] | Bit 2<br>MD2*[2] | Bit 1<br>MD1 | Bit 0<br>MD0 | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal operation |
| | | | 1 | Reserved |
| | | 1 | 0 | PWM mode 1 |
| | | | 1 | PWM mode 2 |
| | 1 | 0 | 0 | Phase counting mode 1 |
| | | | 1 | Phase counting mode 2 |
| | | 1 | 0 | Phase counting mode 3 |
| | | | 1 | Phase counting mode 4 |
| 1 | X | X | X | — |

**Legend**

X: Don't care

Notes: 1. MD3 is a reserved bit. In a write, it should always be written with 0.

2. Phase counting mode cannot be set for channels 0 and 3. In this case, 0 should always be written to MD2.

### 10.3.3 Timer I/O Control Register (TIOR)

The TIOR registers are 8-bit readable/writable registers that control the TGR registers. The TPU has eight TIOR registers, two each for channels 0 and 3, and one each for channels 1, 2, 4, and 5.

Care is required as TIOR is affected by the TMDR setting. The initial output specified by TIOR is valid when the counter is stopped (the CST bit in TSTR is cleared to 0). Note also that, in PWM mode 2, the output at the point at which the counter is cleared to 0 is specified.

When TGRC or TGRD is designated for buffer operation, this setting is invalid and the register operates as a buffer register.

- TIORH_0, TIOR_1, TIOR_2, TIORH_3, TIOR_4, TIOR_5

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7 | IOB3 | 0 | R/W | I/O Control B0 to B3 |
| 6 | IOB2 | 0 | R/W | Specify the function of TGRB. |
| 5 | IOB1 | 0 | R/W | |
| 4 | IOB0 | 0 | R/W | |
| 3 | IOA3 | 0 | R/W | I/O Control A0 to A3 |
| 2 | IOA2 | 0 | R/W | Specify the function of TGRA. |
| 1 | IOA1 | 0 | R/W | |
| 0 | IOA0 | 0 | R/W | |

- TIORL_0, TIORL_3

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7 | IOD3 | 0 | R/W | I/O Control D0 to D3 |
| 6 | IOD2 | 0 | R/W | Specify the function of TGRD. |
| 5 | IOD1 | 0 | R/W | |
| 4 | IOD0 | 0 | R/W | |
| 3 | IOC3 | 0 | R/W | I/O Control C0 to C3 |
| 2 | IOC2 | 0 | R/W | Specify the function of TGRC. |
| 1 | IOC1 | 0 | R/W | |
| 0 | IOC0 | 0 | R/W | |

**Table 10.12 TIORH_0 (Channel 0)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_0 Function** | **TIOCB0 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCB0 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB0 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB0 pin<br>Input capture at both edges. |
| | 1 | X | X | | Capture input source is channel 1/count clock<br>Input capture at TCNT_1 count- up/count-down* |

**Legend**

X: Don't care

Note:* When bits TPSC0 to TPSC2 in TCR_1 are set to B′000 and φ/1 is used as the TCNT_1 count clock, this setting is invalid and input capture is not generated.

**Table 10.13 TIORL_0 (Channel 0)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOD3** | **Bit 6 IOD2** | **Bit 5 IOD1** | **Bit 4 IOD0** | **TGRD_0 Function** | **TIOCD0 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register*[2] | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register*[2] | Capture input source is the TIOCD0 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCD0 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCD0 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 1/count clock<br>Input capture at TCNT_1 count-up/count-down*[1] |

**Legend**

X: Don't care

Notes: 1. When bits TPSC0 to TPSC2 in TCR_1 are set to B′000 and φ/1 is used as the TCNT_1 count clock, this setting is invalid and input capture is not generated.

2. When the BFB bit in TMDR_0 is set to 1 and TGRD_0 is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**Table 10.14 TIOR_1 (Channel 1)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_1 Function** | **TIOCB1 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCB1 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB1 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB1 pin<br>Input capture at both edges |
| | 1 | X | X | | TGRC_0 compare match/ input capture<br>Input capture at generation of TGRC_0 compare match/input capture |

**Legend**
X: Don’t care

**Table 10.15 TIOR_2 (Channel 2)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_2 Function** | **TIOCB2 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | X | 0 | 0 | Input capture register | Capture input source is the TIOCB2 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB2 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB2 pin<br>Input capture at both edges |

**Legend**

X: Don't care

**Table 10.16 TIORH_3 (Channel 3)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_3 Function** | **TIOCB3 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCB3 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB3 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB3 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 4/count clock<br>Input capture at TCNT_4 count-up/count-down* |

**Legend**

X: Don't care

Note:* When bits TPSC0 to TPSC2 in TCR_4 are set to B′000 and φ/1 is used as the TCNT_4 count clock, this setting is invalid and input capture is not generated.

**Table 10.17 TIORL_3 (Channel 3)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOD3** | **Bit 6 IOD2** | **Bit 5 IOD1** | **Bit 4 IOD0** | **TGRD_3 Function** | **TIOCD3 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register*[2] | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register*[2] | Capture input source is the TIOCD3 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCD3 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCD3 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 4/count clock<br>Input capture at TCNT_4 count-up/count-down*[1] |

**Legend**

X: Don't care

Notes: 1. When bits TPSC0 to TPSC2 in TCR_4 are set to B′000 and φ/1 is used as the TCNT_4 count clock, this setting is invalid and input capture is not generated.

2. When the BFB bit in TMDR_3 is set to 1 and TGRD_3 is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**Table 10.18 TIOR_4 (Channel 4)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_4 Function** | **TIOCB4 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCB4 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB4 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB4 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is TGRC_3 compare match/input capture<br>Input capture at generation of TGRC_3 compare match/input capture |

**Legend**

X: Don’t care

**Table 10.19 TIOR_5 (Channel 5)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 7 IOB3** | **Bit 6 IOB2** | **Bit 5 IOB1** | **Bit 4 IOB0** | **TGRB_5 Function** | **TIOCB5 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | X | 0 | 0 | Input capture register | Capture input source is the TIOCB5 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCB5 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCB5 pin<br>Input capture at both edges |

**Legend**

X: Don't care

**Table 10.20 TIORH_0 (Channel 0)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_0 Function** | **TIOCA0 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCA0 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA0 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA0 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 1/count clock<br>Input capture at TCNT_1 count-up/count-down |

**Legend**

X: Don’t care

**Table 10.21 TIORL_0 (Channel 0)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOC3** | **Bit 2 IOC2** | **Bit 1 IOC1** | **Bit 0 IOC0** | **TGRC_0 Function** | **TIOCC0 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register* | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register* | Capture input source is the TIOCC0 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCC0 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCC0 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 1/count clock<br>Input capture at TCNT_1 count-up/count-down |

**Legend**

X: Don't care

Note:* When the BFA bit in TMDR_0 is set to 1 and TGRC_0 is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**Table 10.22 TIOR_1 (Channel 1)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_1 Function** | **TIOCA1 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCA1 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA1 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA1 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is TGRA_0 compare match/input capture<br>Input capture at generation of channel 0/TGRA_0 compare match/input capture |

**Legend**

X: Don’t care

**Table 10.23 TIOR_2 (Channel 2)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_2 Function** | **TIOCA2 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | X | 0 | 0 | Input capture register | Capture input source is the TIOCA2 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA2 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA2 pin<br>Input capture at both edges |

**Legend**

X: Don’t care

**Table 10.24 TIORH_3 (Channel 3)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_3 Function** | **TIOCA3 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCA3 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA3 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA3 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 4/count clock<br>Input capture at TCNT_4 count-up/count-down |

**Legend**

X: Don’t care

**Table 10.25 TIORL_3 (Channel 3)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOC3** | **Bit 2 IOC2** | **Bit 1 IOC1** | **Bit 0 IOC0** | **TGRC_3 Function** | **TIOCC3 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register* | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register* | Capture input source is the TIOCC3 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCC3 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCC3 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is channel 4/count clock<br>Input capture at TCNT_4 count-up/count-down |

**Legend**

X: Don't care

Note:* When the BFA bit in TMDR_3 is set to 1 and TGRC_3 is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**Table 10.26 TIOR_4 (Channel 4)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_4 Function** | **TIOCA4 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is the TIOCA4 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA4 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA4 pin<br>Input capture at both edges |
| | 1 | X | X | | Capture input source is TGRA_3 compare match/input capture<br>Input capture at generation of TGRA_3 compare match/input capture |

**Legend**

X: Don’t care

**Table 10.27 TIOR_5 (Channel 5)**

| | | | | Description | |
|---|---|---|---|---|---|
| **Bit 3 IOA3** | **Bit 2 IOA2** | **Bit 1 IOA1** | **Bit 0 IOA0** | **TGRA_5 Function** | **TIOCA5 Pin Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| | | | 1 | | Initial output is 0<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 0<br>1 output at compare match |
| | | | 1 | | Initial output is 0<br>Toggle output at compare match |
| | 1 | 0 | 0 | | Output disabled |
| | | | 1 | | Initial output is 1<br>0 output at compare match |
| | | 1 | 0 | | Initial output is 1<br>1 output at compare match |
| | | | 1 | | Initial output is 1<br>Toggle output at compare match |
| 1 | X | 0 | 0 | Input capture register | Capture input source is the TIOCA5 pin<br>Input capture at rising edge |
| | | | 1 | | Capture input source is the TIOCA5 pin<br>Input capture at falling edge |
| | | 1 | X | | Capture input source is the TIOCA5 pin<br>Input capture at both edges |

**Legend**

X: Don’t care

### 10.3.4 Timer Interrupt Enable Register (TIER)

The TIER registers are 8-bit readable/writable registers that control enabling or disabling of interrupt requests for each channel. The TPU has six TIER registers, one for each channel.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7 | TTGE | 0 | R/W | A/D Conversion Start Request Enable<br>Enables or disables generation of A/D conversion start requests by TGRA input capture/compare match.<br>0: A/D conversion start request generation disabled<br>1: A/D conversion start request generation enabled |
| 6 | — | 1 | — | Reserved<br>This bit is always read as 1 and cannot be modified. |
| 5 | TCIEU | 0 | R/W | Underflow Interrupt Enable<br>Enables or disables interrupt requests (TCIU) by the TCFU flag when the TCFU flag in TSR is set to 1 in channels 1, 2, 4, and 5.<br>In channels 0 and 3, bit 5 is reserved. It is always read as 0 and cannot be modified.<br>0: Interrupt requests (TCIU) by TCFU disabled<br>1: Interrupt requests (TCIU) by TCFU enabled |
| 4 | TCIEV | 0 | R/W | Overflow Interrupt Enable<br>Enables or disables interrupt requests (TCIV) by the TCFV flag when the TCFV flag in TSR is set to 1.<br>0: Interrupt requests (TCIV) by TCFV disabled<br>1: Interrupt requests (TCIV) by TCFV enabled |
| 3 | TGIED | 0 | R/W | TGR Interrupt Enable D<br>Enables or disables interrupt requests (TGID) by the TGFD bit when the TGFD bit in TSR is set to 1 in channels 0 and 3.<br>In channels 1, 2, 4, and 5, bit 3 is reserved. It is always read as 0 and cannot be modified.<br>0: Interrupt requests (TGID) by TGFD bit disabled<br>1: Interrupt requests (TGID) by TGFD bit enabled |

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 2 | TGIEC | 0 | R/W | TGR Interrupt Enable C<br>Enables or disables interrupt requests (TGIC) by the TGFC bit when the TGFC bit in TSR is set to 1 in channels 0 and 3.<br>In channels 1, 2, 4, and 5, bit 2 is reserved. It is always read as 0 and cannot be modified.<br>0: Interrupt requests (TGIC) by TGFC bit disabled<br>1: Interrupt requests (TGIC) by TGFC bit enabled |
| 1 | TGIEB | 0 | R/W | TGR Interrupt Enable B<br>Enables or disables interrupt requests (TGIB) by the TGFB bit when the TGFB bit in TSR is set to 1.<br>0: Interrupt requests (TGIB) by TGFB bit disabled<br>1: Interrupt requests (TGIB) by TGFB bit enabled |
| 0 | TGIEA | 0 | R/W | TGR Interrupt Enable A<br>Enables or disables interrupt requests (TGIA) by the TGFA bit when the TGFA bit in TSR is set to 1.<br>0: Interrupt requests (TGIA) by TGFA bit disabled<br>1: Interrupt requests (TGIA) by TGFA bit enabled |

### 10.3.5 Timer Status Register (TSR)

The TSR registers are 8-bit readable/writable registers that indicate the status of each channel. The TPU has six TSR registers, one for each channel.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7 | TCFD | 1 | R | Count Direction Flag<br>Status flag that shows the direction in which TCNT counts in channels 1, 2, 4, and 5.<br>In channels 0 and 3, bit 7 is reserved. It is always read as 1 and cannot be modified.<br>0: TCNT counts down<br>1: TCNT counts up |
| 6 | — | 1 | — | Reserved<br>This bit is always read as 1 and cannot be modified. |
| 5 | TCFU | 0 | R/(W) | Underflow Flag<br>Status flag that indicates that TCNT underflow has occurred when channels 1, 2, 4, and 5 are set to phase counting mode. Only 0 can be written, for flag clearing.<br>In channels 0 and 3, bit 5 is reserved. It is always read as 0 and cannot be modified.<br>[Setting condition]<br>When the TCNT value underflows (changes from H′0000 to H′FFFF)<br>[Clearing condition]<br>When 0 is written to TCFU after reading TCFU = 1 |
| 4 | TCFV | 0 | R/(W) | Overflow Flag<br>Status flag that indicates that TCNT overflow has occurred. Only 0 can be written, for flag clearing.<br>[Setting condition]<br>When the TCNT value overflows (changes from H′FFFF to H′0000 )<br>[Clearing condition]<br>When 0 is written to TCFV after reading TCFV = 1 |

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 3 | TGFD | 0 | R/(W) | Input Capture/Output Compare Flag D<br>Status flag that indicates the occurrence of TGRD input capture or compare match in channels 0 and 3. Only 0 can be written, for flag clearing. In channels 1, 2, 4, and 5, bit 3 is reserved. It is always read as 0 and cannot be modified.<br>[Setting conditions]<br>• When TCNT = TGRD and TGRD is functioning as output compare register<br>• When TCNT value is transferred to TGRD by input capture signal and TGRD is functioning as input capture register<br>[Clearing conditions]<br>• When DTC is activated by TGID interrupt and the DISEL bit of MRB in DTC is 0<br>• When 0 is written to TGFD after reading TGFD = 1 |
| 2 | TGFC | 0 | R/(W) | Input Capture/Output Compare Flag C<br>Status flag that indicates the occurrence of TGRC input capture or compare match in channels 0 and 3. Only 0 can be written, for flag clearing. In channels 1, 2, 4, and 5, bit 2 is reserved. It is always read as 0 and cannot be modified.<br>[Setting conditions]<br>• When TCNT = TGRC and TGRC is functioning as output compare register<br>• When TCNT value is transferred to TGRC by input capture signal and TGRC is functioning as input capture register<br>[Clearing conditions]<br>• When DTC is activated by TGIC interrupt and the DISEL bit of MRB in DTC is 0<br>• When 0 is written to TGFC after reading TGFC = 1 |

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 1 | TGFB | 0 | R/(W) | Input Capture/Output Compare Flag B<br>Status flag that indicates the occurrence of TGRB input capture or compare match. Only 0 can be written, for flag clearing.<br>[Setting conditions]<br>• When TCNT = TGRB and TGRB is functioning as output compare register<br>• When TCNT value is transferred to TGRB by input capture signal and TGRB is functioning as input capture register<br>[Clearing conditions]<br>• When DTC is activated by TGIB interrupt and the DISEL bit of MRB in DTC is 0<br>• When 0 is written to TGFB after reading TGFB = 1 |
| 0 | TGFA | 0 | R/(W) | Input Capture/Output Compare Flag A<br>Status flag that indicates the occurrence of TGRA input capture or compare match. Only 0 can be written, for flag clearing.<br>[Setting conditions]<br>• When TCNT = TGRA and TGRA is functioning as output compare register<br>• When TCNT value is transferred to TGRA by input capture signal and TGRA is functioning as input capture register<br>[Clearing conditions]<br>• When DTC is activated by TGIA interrupt and the DISEL bit of MRB in DTC is 0<br>• When 0 is written to TGFA after reading TGFA = 1 |

### 10.3.6 Timer Counter (TCNT)

The TCNT registers are 16-bit readable/writable counters. The TPU has six TCNT counters, one for each channel.

The TCNT counters are initialized to H′0000 by a reset, and in hardware standby mode.

The TCNT counters cannot be accessed in 8-bit units; they must always be accessed as a 16-bit unit.

### 10.3.7 Timer General Register (TGR)

The TGR registers are dual function 16-bit readable/writable registers, functioning as either output compare or input capture registers. The TPU has 16 TGR registers, four each for channels 0 and 3 and two each for channels 1, 2, 4, and 5. TGRC and TGRD for channels 0 and 3 can also be designated for operation as buffer registers. The TGR registers cannot be accessed in 8-bit units; they must always be accessed as a 16-bit unit. TGR buffer register combinations are TGRA—TGRC and TGRB—TGRD.

### 10.3.8 Timer Start Register (TSTR)

TSTR is an 8-bit readable/writable register that selects operation/stoppage for channels 0 to 5. When setting the operating mode in TMDR or setting the count clock in TCR, first stop the TCNT counter.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 5 | CST5 | 0 | R/W | Counter Start 0 to 5 (CST0 to CST5)<br>These bits select operation or stoppage for TCNT.<br>If 0 is written to the CST bit during operation with the TIOC pin designated for output, the counter stops but the TIOC pin output compare output level is retained. If TIOR is written to when the CST bit is cleared to 0, the pin output level will be changed to the set initial output value.<br>0: TCNT_0 to TCNT_5 count operation is stopped<br>1: TCNT_0 to TCNT_5 performs count operation |
| 4 | CST4 | 0 | R/W | |
| 3 | CST3 | 0 | R/W | |
| 2 | CST2 | 0 | R/W | |
| 1 | CST1 | 0 | R/W | |
| 0 | CST0 | 0 | R/W | |

### 10.3.9 Timer Synchro Register (TSYR)

TSYR is an 8-bit readable/writable register that selects independent operation or synchronous operation for the channel 0 to 5 TCNT counters. A channel performs synchronous operation when the corresponding bit in TSYR is set to 1.

| Bit | Bit Name | Initial value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | R/W | Reserved<br>The write value should always be 0. |
| 5 | SYNC5 | 0 | R/W | Timer Synchro 0 to 5<br>These bits are used to select whether operation is independent of or synchronized with other channels.<br>When synchronous operation is selected, the TCNT synchronous presetting of multiple channels, and synchronous clearing by counter clearing on another channel, are possible.<br>To set synchronous operation, the SYNC bits for at least two channels must be set to 1. To set synchronous clearing, in addition to the SYNC bit , the TCNT clearing source must also be set by means of bits CCLR0 to CCLR2 in TCR.<br>0: TCNT_0 to TCNT_5 operates independently (TCNT presetting /clearing is unrelated to other channels)<br>1: TCNT_0 to TCNT_5 performs synchronous operation<br>TCNT synchronous presetting/synchronous clearing is possible |
| 4 | SYNC4 | 0 | R/W | |
| 3 | SYNC3 | 0 | R/W | |
| 2 | SYNC2 | 0 | R/W | |
| 1 | SYNC1 | 0 | R/W | |
| 0 | SYNC0 | 0 | R/W | |

## 10.4 Operation

### 10.4.1 Basic Functions

Each channel has a TCNT and TGR register. TCNT performs up-counting, and is also capable of free-running operation, periodic counting, and external event counting.

Each TGR can be used as an input capture register or output compare register.

**Counter Operation:** When one of bits CST0 to CST5 is set to 1 in TSTR, the TCNT counter for the corresponding channel begins counting. TCNT can operate as a free-running counter, periodic counter, for example.

1. Example of count operation setting procedure

   Figure 10.2 shows an example of the count operation setting procedure.

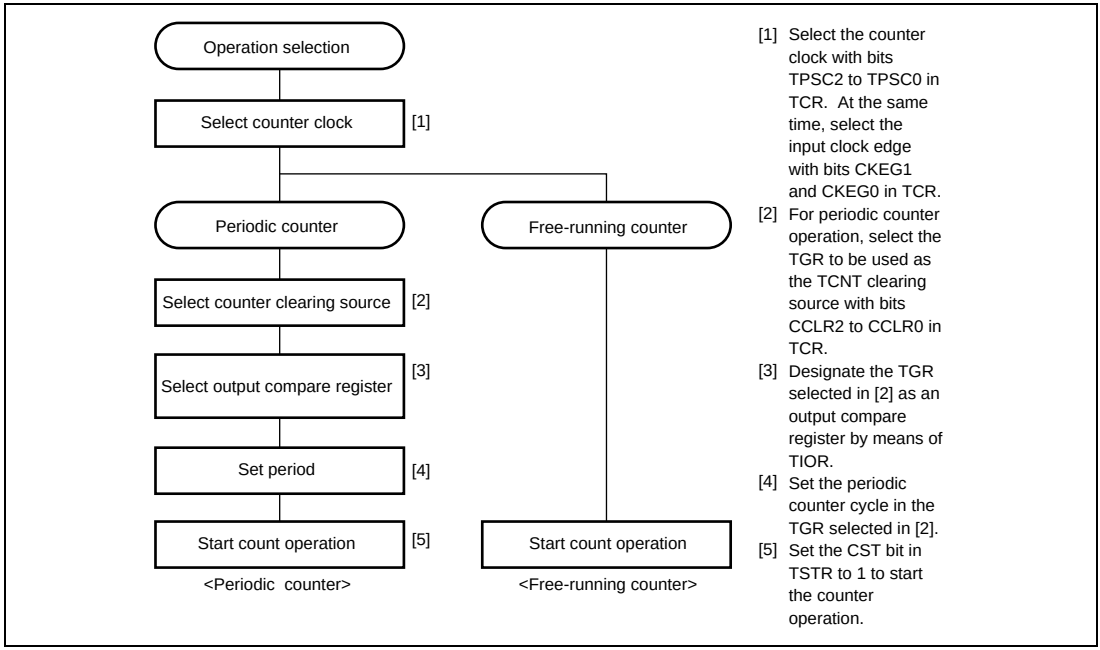



**Figure 10.2 Example of Counter Operation Setting Procedure**

2. Free-running count operation and periodic count operation

   Immediately after a reset, the TPU's TCNT counters are all designated as free-running counters. When the relevant bit in TSTR is set to 1 the corresponding TCNT counter starts up-count operation as a free-running counter. When TCNT overflows (from H′FFFF to H′0000), the TCFV bit in TSR is set to 1. If the value of the corresponding TCIEV bit in TIER is 1 at this point, the TPU requests an interrupt. After overflow, TCNT starts counting up again from H′0000.

   Figure 10.3 illustrates free-running counter operation.

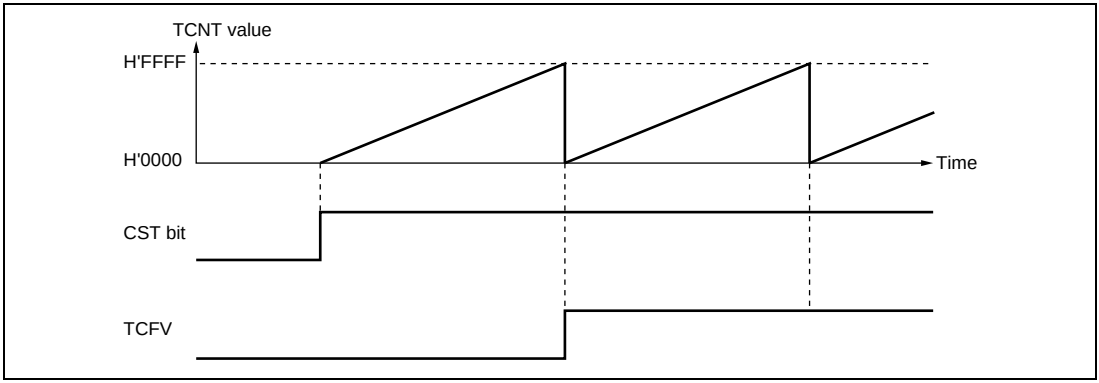



**Figure 10.3 Free-Running Counter Operation**

When compare match is selected as the TCNT clearing source, the TCNT counter for the relevant channel performs periodic count operation. The TGR register for setting the period is designated as an output compare register, and counter clearing by compare match is selected by means of bits CCLR0 to CCLR2 in TCR. After the settings have been made, TCNT starts up-count operation as a periodic counter when the corresponding bit in TSTR is set to 1. When the count value matches the value in TGR, the TGF bit in TSR is set to 1 and TCNT is cleared to H′0000.

If the value of the corresponding TGIE bit in TIER is 1 at this point, the TPU requests an interrupt. After a compare match, TCNT starts counting up again from H′0000.

Figure 10.4 illustrates periodic counter operation.

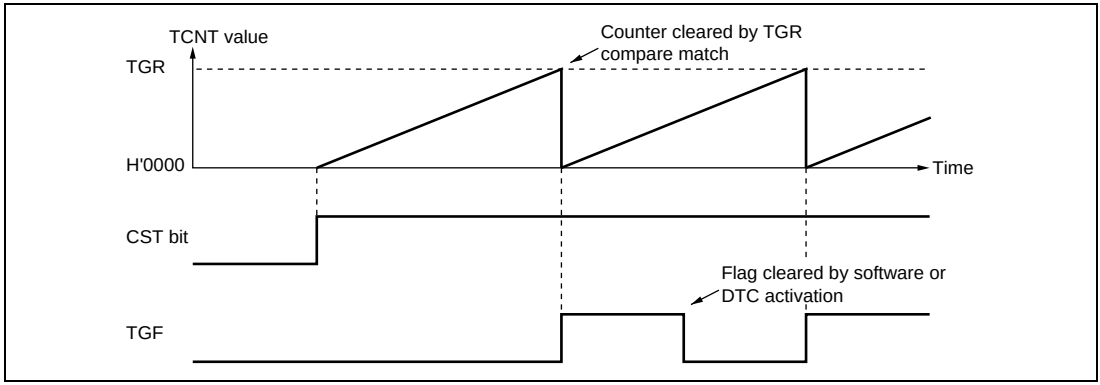



**Figure 10.4 Periodic Counter Operation**

**Waveform Output by Compare Match:** The TPU can perform 0, 1, or toggle output from the corresponding output pin using compare match.

1. Example of setting procedure for waveform output by compare match

   Figure 10.5 shows an example of the setting procedure for waveform output by compare match

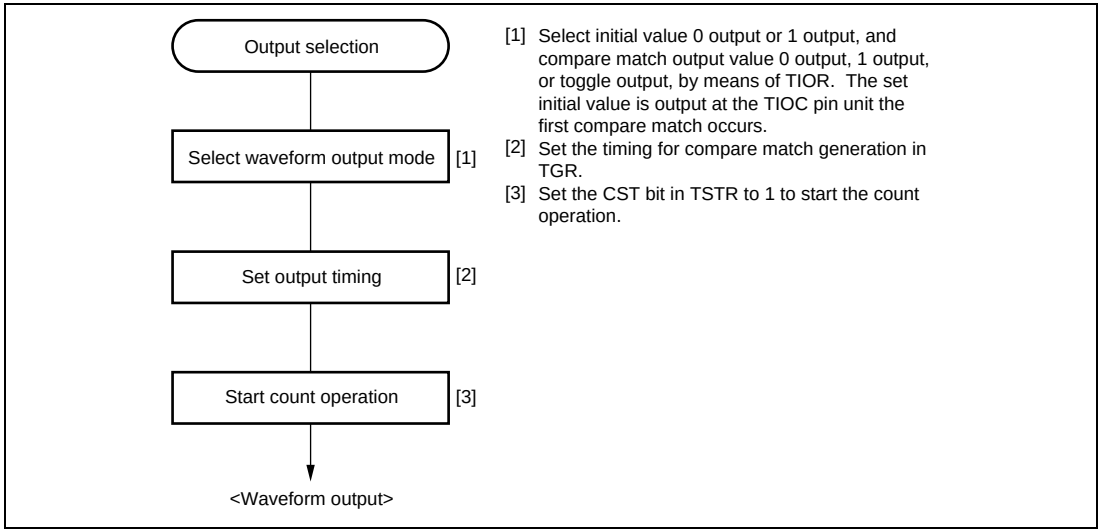



**Figure 10.5 Example of Setting Procedure for Waveform Output by Compare Match**

2. Examples of waveform output operation

Figure 10.6 shows an example of 0 output/1 output.

In this example TCNT has been designated as a free-running counter, and settings have been made such that 1 is output by compare match A, and 0 is output by compare match B. When the set level and the pin level coincide, the pin level does not change.

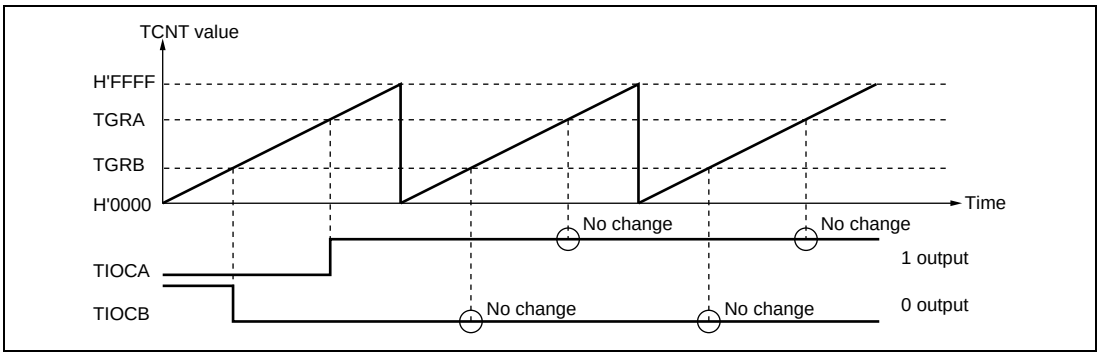



**Figure 10.6 Example of 0 Output/1 Output Operation**

Figure 10.7 shows an example of toggle output.

In this example, TCNT has been designated as a periodic counter (with counter clearing on compare match B), and settings have been made such that the output is toggled by both compare match A and compare match B.

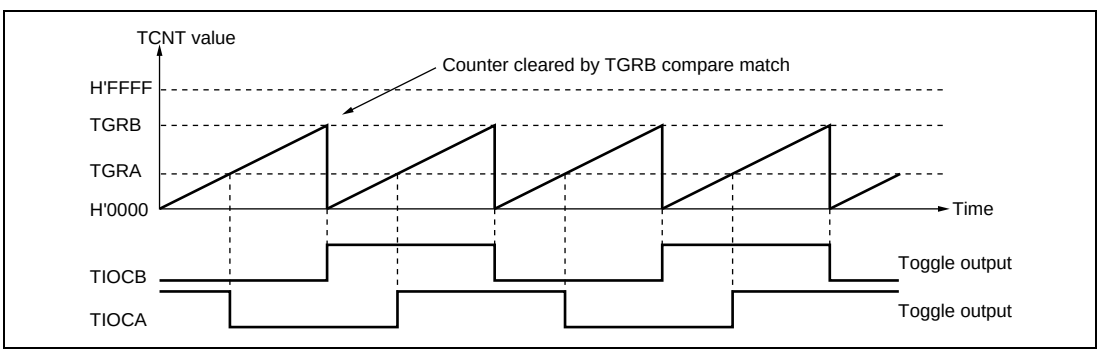



**Figure 10.7 Example of Toggle Output Operation**

**Input Capture Function:** The TCNT value can be transferred to TGR on detection of the TIOC pin input edge.

Rising edge, falling edge, or both edges can be selected as the detected edge. For channels 0, 1, 3, and 4, it is also possible to specify another channel's counter input clock or compare match signal as the input capture source.

Note: When another channel's counter input clock is used as the input capture input for channels 0 and 3, φ/1 should not be selected as the counter input clock used for input capture input. Input capture will not be generated if φ/1 is selected.

1. Example of input capture operation setting procedure
   Figure 10.8 shows an example of the input capture operation setting procedure.

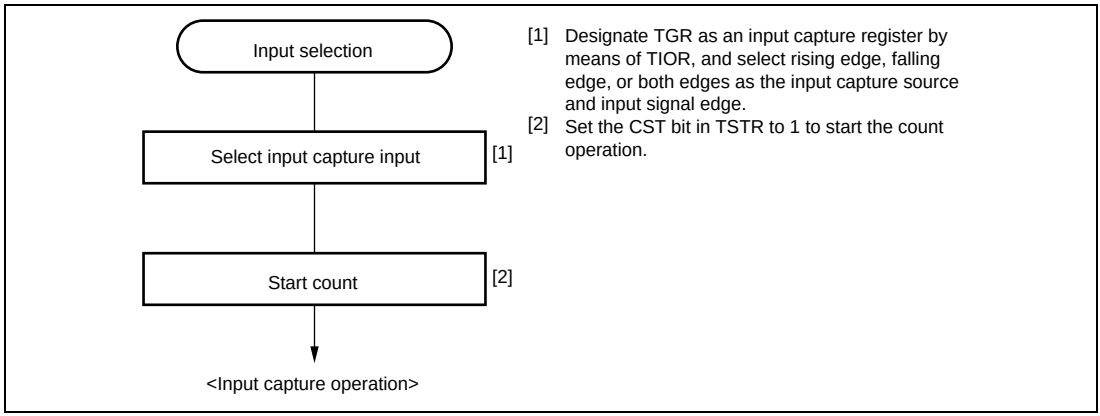



**Figure 10.8 Example of Input Capture Operation Setting Procedure**

2. Example of input capture operation

Figure 10.9 shows an example of input capture operation.

In this example both rising and falling edges have been selected as the TIOCA pin input capture input edge, the falling edge has been selected as the TIOCB pin input capture input edge, and counter clearing by TGRB input capture has been designated for TCNT.

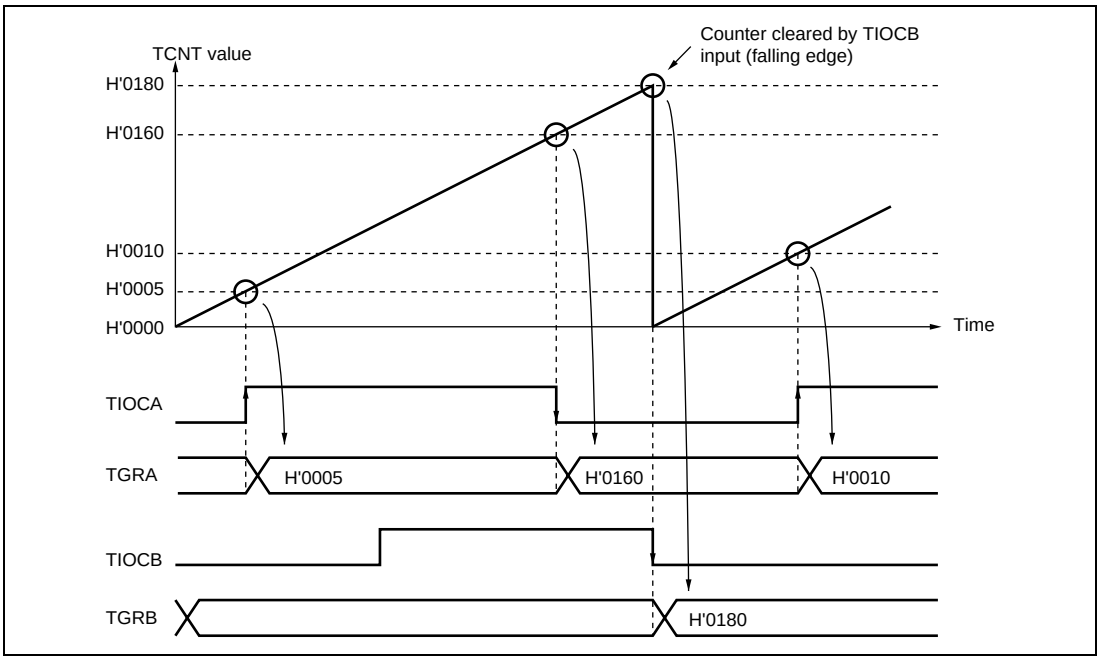



**Figure 10.9 Example of Input Capture Operation**

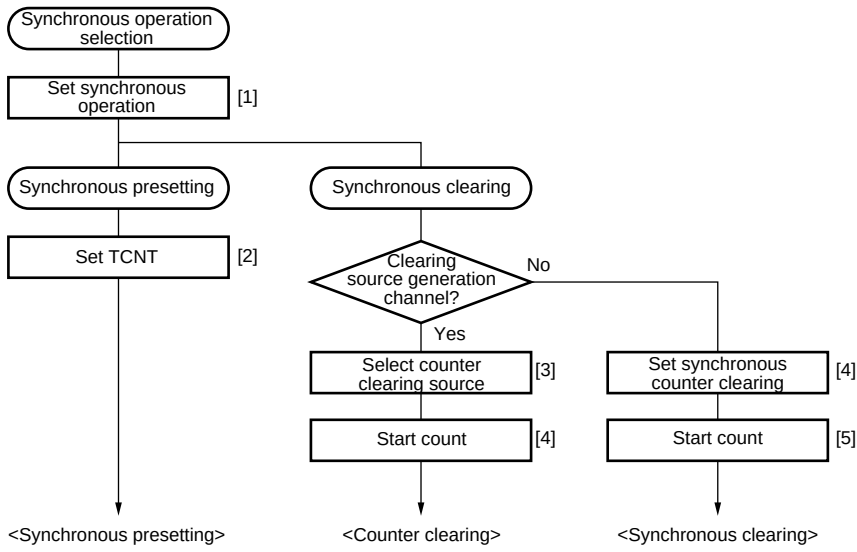
### 10.4.2 Synchronous Operation

In synchronous operation, the values in a number of TCNT counters can be rewritten simultaneously (synchronous presetting). Also, a number of TCNT counters can be cleared simultaneously by making the appropriate setting in TCR (synchronous clearing).

Synchronous operation enables TGR to be incremented with respect to a single time base.

Channels 0 to 5 can all be designated for synchronous operation.

**Example of Synchronous Operation Setting Procedure:** Figure 10.10 shows an example of the synchronous operation setting procedure.




[1] Set to 1 the SYNC bits in TSYR corresponding to the channels to be designated for synchronous operation.
[2] When the TCNT counter of any of the channels designated for synchronous operation is written to, the same value is simultaneously written to the other TCNT counters.
[3] Use bits CCLR2 to CCLR0 in TCR to specify TCNT clearing by input capture/output compare, etc.
[4] Use bits CCLR2 to CCLR0 in TCR to designate synchronous clearing for the counter clearing source.
[5] Set to 1 the CST bits in TSTR for the relevant channels, to start the count operation.

**Figure 10.10 Example of Synchronous Operation Setting Procedure**

**Example of Synchronous Operation:** Figure 10.11 shows an example of synchronous operation.

In this example, synchronous operation and PWM mode 1 have been designated for channels 0 to 2, TGRB_0 compare match has been set as the channel 0 counter clearing source, and synchronous clearing has been set for the channel 1 and 2 counter clearing source.

Three-phase PWM waveforms are output from pins TIOCA0, TIOCA1, and TIOCA2. At this time, synchronous presetting, and synchronous clearing by TGRB_0 compare match, are performed for channel 0 to 2 TCNT counters, and the data set in TGRB_0 is used as the PWM cycle.

For details of PWM modes, see section 10.4.5, PWM Modes.

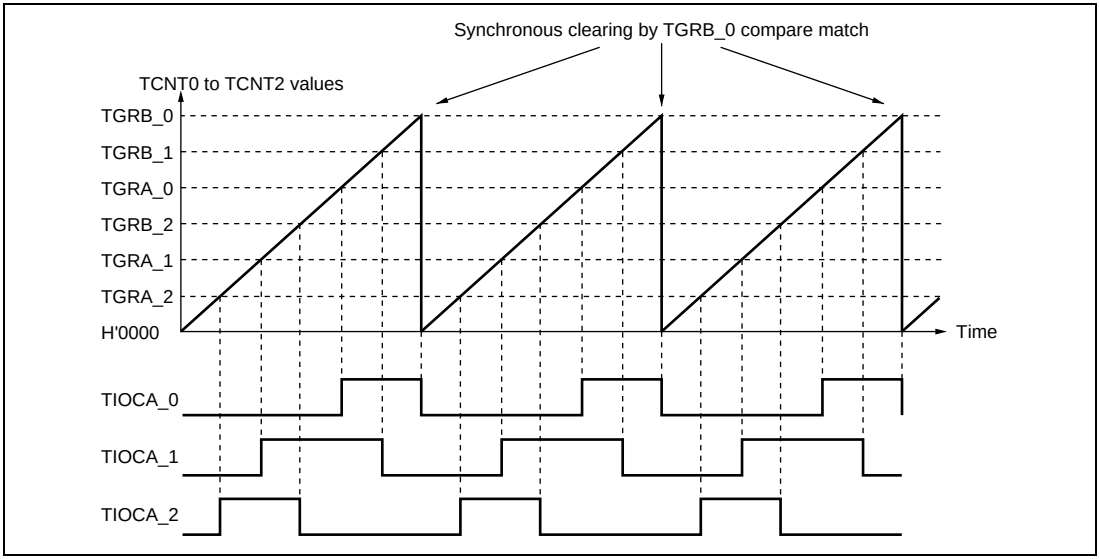



**Figure 10.11 Example of Synchronous Operation**

### 10.4.3 Buffer Operation

Buffer operation, provided for channels 0 and 3, enables TGRC and TGRD to be used as buffer registers.

Buffer operation differs depending on whether TGR has been designated as an input capture register or as a compare match register.

Table 10.28 shows the register combinations used in buffer operation.

**Table 10.28 Register Combinations in Buffer Operation**

| Channel | Timer General Register | Buffer Register |
|---|---|---|
| 0 | TGRA_0 | TGRC_0 |
| | TGRB_0 | TGRD_0 |
| 3 | TGRA_3 | TGRC_3 |
| | TGRB_3 | TGRD_3 |

- When TGR is an output compare register

  When a compare match occurs, the value in the buffer register for the corresponding channel is transferred to the timer general register.

  This operation is illustrated in figure 10.12.

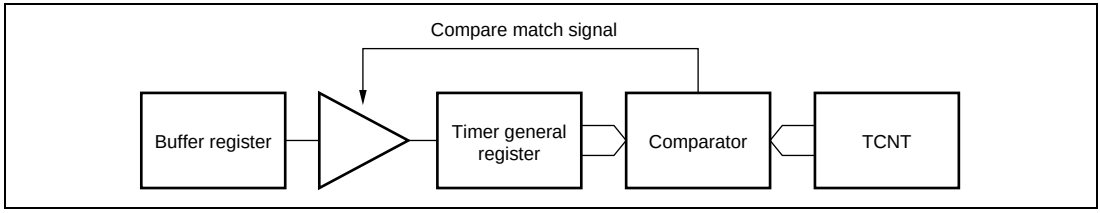



**Figure 10.12 Compare Match Buffer Operation**

- When TGR is an input capture register

  When input capture occurs, the value in TCNT is transferred to TGR and the value previously held in the timer general register is transferred to the buffer register.

  This operation is illustrated in figure 10.13.

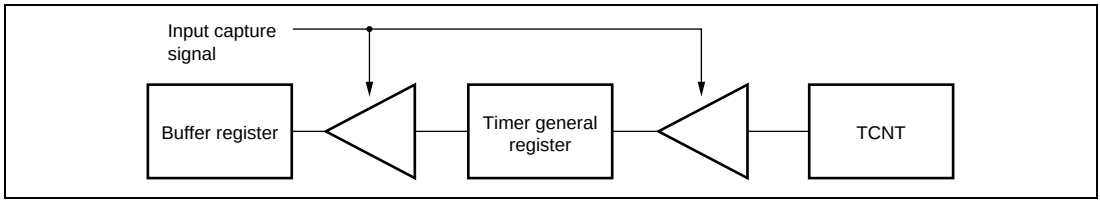



**Figure 10.13 Input Capture Buffer Operation**

**Example of Buffer Operation Setting Procedure:** Figure 10.14 shows an example of the buffer operation setting procedure.

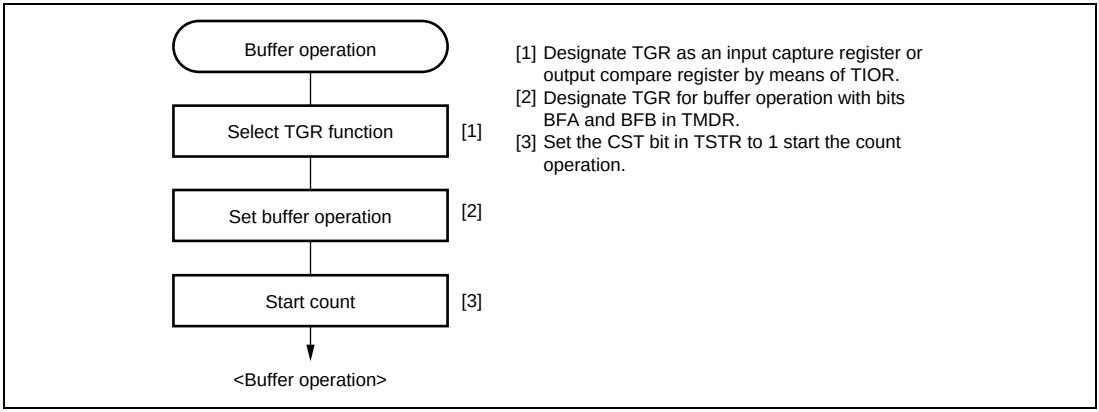



**Figure 10.14 Example of Buffer Operation Setting Procedure**

**Examples of Buffer Operation:**

1. When TGR is an output compare register

   Figure 10.15 shows an operation example in which PWM mode 1 has been designated for channel 0, and buffer operation has been designated for TGRA and TGRC. The settings used in this example are TCNT clearing by compare match B, 1 output at compare match A, and 0 output at compare match B.

   As buffer operation has been set, when compare match A occurs the output changes and the value in buffer register TGRC is simultaneously transferred to timer general register TGRA. This operation is repeated each time that compare match A occurs.

   For details of PWM modes, see section 10.4.5, PWM Modes.

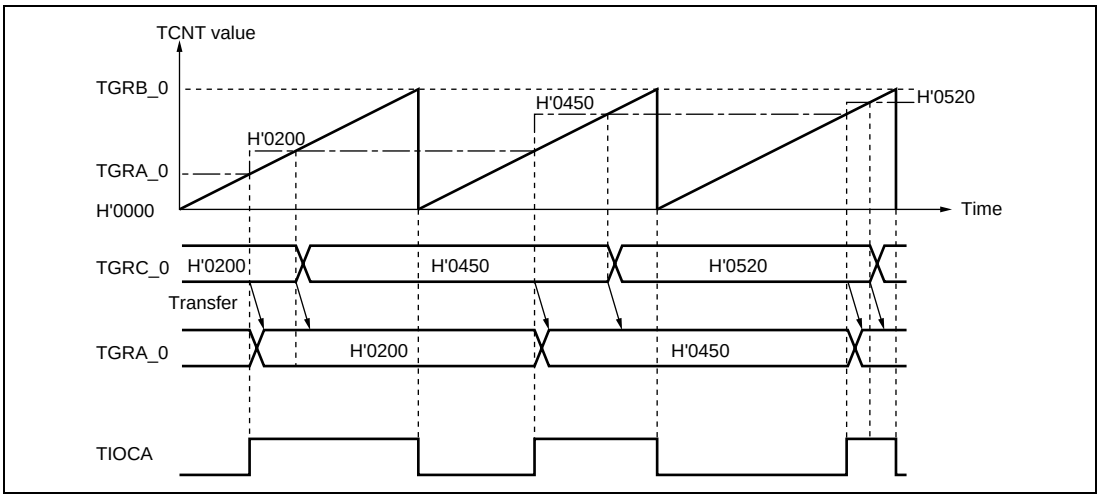



**Figure 10.15 Example of Buffer Operation (1)**

2. When TGR is an input capture register

   Figure 10.16 shows an operation example in which TGRA has been designated as an input capture register, and buffer operation has been designated for TGRA and TGRC.

   Counter clearing by TGRA input capture has been set for TCNT, and both rising and falling edges have been selected as the TIOCA pin input capture input edge.

   As buffer operation has been set, when the TCNT value is stored in TGRA upon the occurrence of input capture A, the value previously stored in TGRA is simultaneously transferred to TGRC.

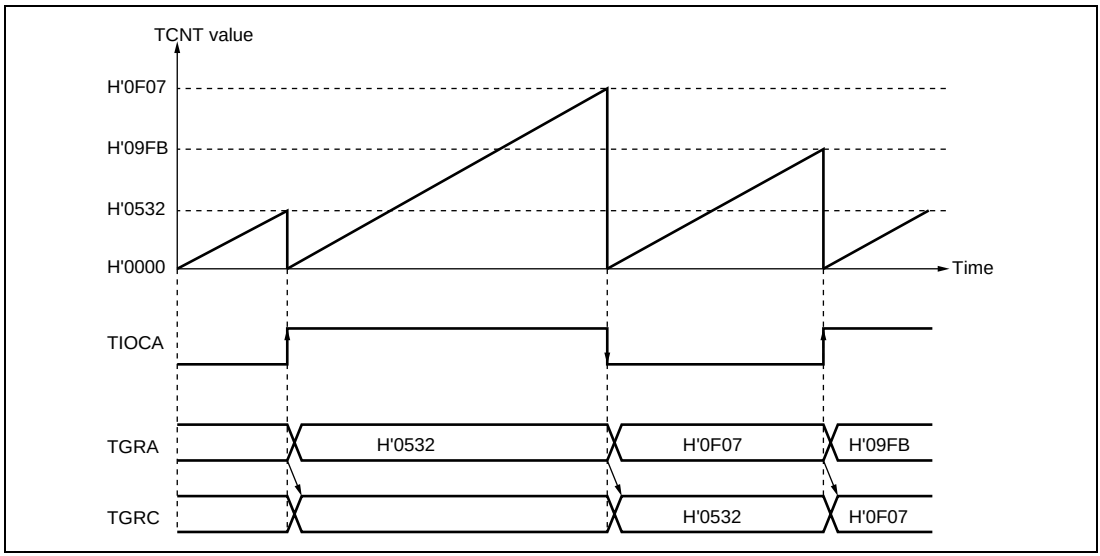



**Figure 10.16 Example of Buffer Operation (2)**

### 10.4.4 Cascaded Operation

In cascaded operation, two 16-bit counters for different channels are used together as a 32-bit counter.

This function works by counting the channel 1 (channel 4) counter clock upon overflow/underflow of TCNT_2 (TCNT_5) as set in bits TPSC0 to TPSC2 in TCR.

Underflow occurs only when the lower 16-bit TCNT is in phase-counting mode.

Table 10.29 shows the register combinations used in cascaded operation.

Note: When phase counting mode is set for channel 1 or 4, the counter clock setting is invalid and the counters operates independently in phase counting mode.

**Table 10.29 Cascaded Combinations**

| Combination | Upper 16 Bits | Lower 16 Bits |
|---|---|---|
| Channels 1 and 2 | TCNT_1 | TCNT_2 |
| Channels 4 and 5 | TCNT_4 | TCNT_5 |

**Example of Cascaded Operation Setting Procedure:** Figure 10.17 shows an example of the setting procedure for cascaded operation.

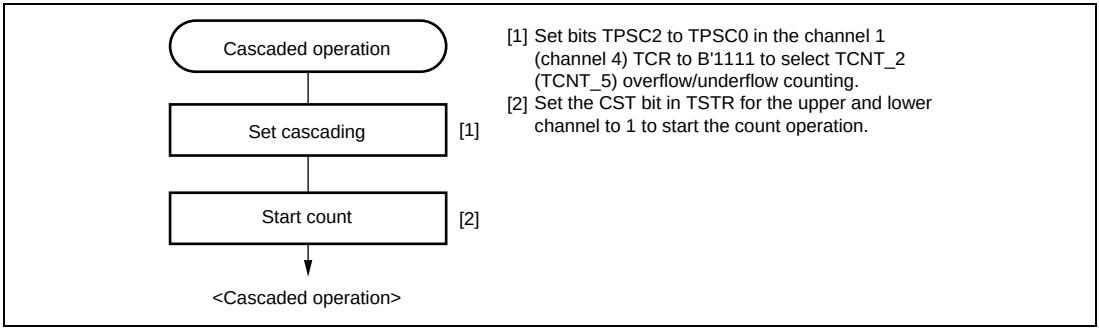



**Figure 10.17 Cascaded Operation Setting Procedure**

**Examples of Cascaded Operation:** Figure 10.18 illustrates the operation when TCNT_2 overflow/underflow counting has been set for TCNT_1, when TGRA_1 and TGRA_2 have been designated as input capture registers, and when TIOC pin rising edge has been selected.

When a rising edge is input to the TIOCA1 and TIOCA2 pins simultaneously, the upper 16 bits of the 32-bit data are transferred to TGRA_1, and the lower 16 bits to TGRA_2.

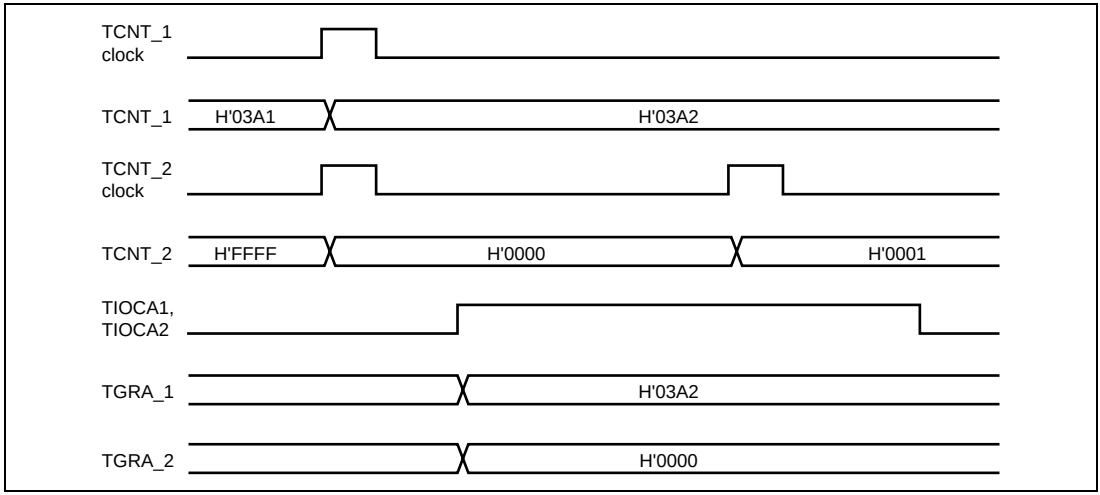



**Figure 10.18 Example of Cascaded Operation (1)**

Figure 10.19 illustrates the operation when TCNT_2 overflow/underflow counting has been set for TCNT_1 and phase counting mode has been designated for channel 2.

TCNT_1 is incremented by TCNT_2 overflow and decremented by TCNT_2 underflow.

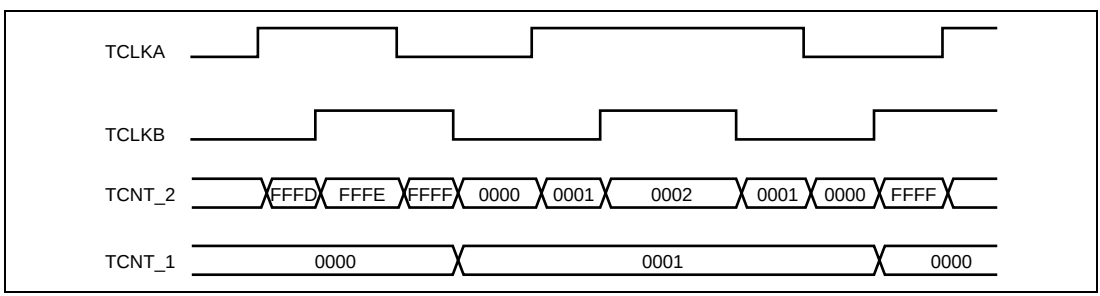



**Figure 10.19 Example of Cascaded Operation (2)**

### 10.4.5 PWM Modes

In PWM mode, PWM waveforms are output from the output pins. The output level can be selected as 0, 1, or toggle output in response to a compare match of each TGR.

TGR registers settings can be used to output a PWM waveform in the range of 0% to 100% duty cycle.

Designating TGR compare match as the counter clearing source enables the period to be set in that register. All channels can be designated for PWM mode independently. Synchronous operation is also possible.

There are two PWM modes, as described below.

- PWM mode 1

  PWM output is generated from the TIOCA and TIOCC pins by pairing TGRA with TGRB and TGRC with TGRD. The output specified by bits IOA0 to IOA3 and IOC0 to IOC3 in TIOR is output from the TIOCA and TIOCC pins at compare matches A and C, and the output specified by bits IOB0 to IOB3 and IOD0 to IOD3 in TIOR is output at compare matches B and D. The initial output value is the value set in TGRA or TGRC. If the set values of paired TGRs are identical, the output value does not change when a compare match occurs.

  In PWM mode 1, a maximum 8-phase PWM output is possible.

- PWM mode 2

  PWM output is generated using one TGR as the cycle register and the others as duty cycle registers. The output specified in TIOR is performed by means of compare matches. Upon counter clearing by a synchronization register compare match, the output value of each pin is the initial value set in TIOR. If the set values of the cycle and duty cycle registers are identical, the output value does not change when a compare match occurs.

  In PWM mode 2, a maximum 15-phase PWM output is possible in combination use with synchronous operation.

The correspondence between PWM output pins and registers is shown in table 10.30.

**Table 10.30 PWM Output Registers and Output Pins**

| | | Output Pins | |
|---|---|---|---|
| **Channel** | **Registers** | **PWM Mode 1** | **PWM Mode 2** |
| 0 | TGRA_0 | TIOCA0 | TIOCA0 |
| | TGRB_0 | | TIOCB0 |
| | TGRC_0 | TIOCC0 | TIOCC0 |
| | TGRD_0 | | TIOCD0 |
| 1 | TGRA_1 | TIOCA1 | TIOCA1 |
| | TGRB_1 | | TIOCB1 |
| 2 | TGRA_2 | TIOCA2 | TIOCA2 |
| | TGRB_2 | | TIOCB2 |
| 3 | TGRA_3 | TIOCA3 | TIOCA3 |
| | TGRB_3 | | TIOCB3 |
| | TGRC_3 | TIOCC3 | TIOCC3 |
| | TGRD_3 | | TIOCD3 |

**Table 10.30 PWM Output Registers and Output Pins (cont)**

| Channel | Registers | Output Pins: PWM Mode 1 | Output Pins: PWM Mode 2 |
|---|---|---|---|
| 4 | TGR4A_4 | TIOCA4 | TIOCA4 |
| | TGR4B_4 | | TIOCB4 |
| 5 | TGRA_5 | TIOCA5 | TIOCA5 |
| | TGRB_5 | | TIOCB5 |

Note:* In PWM mode 2, PWM output is not possible for the TGR register in which the period is set.

**Example of PWM Mode Setting Procedure:** Figure 10.20 shows an example of the PWM mode setting procedure.

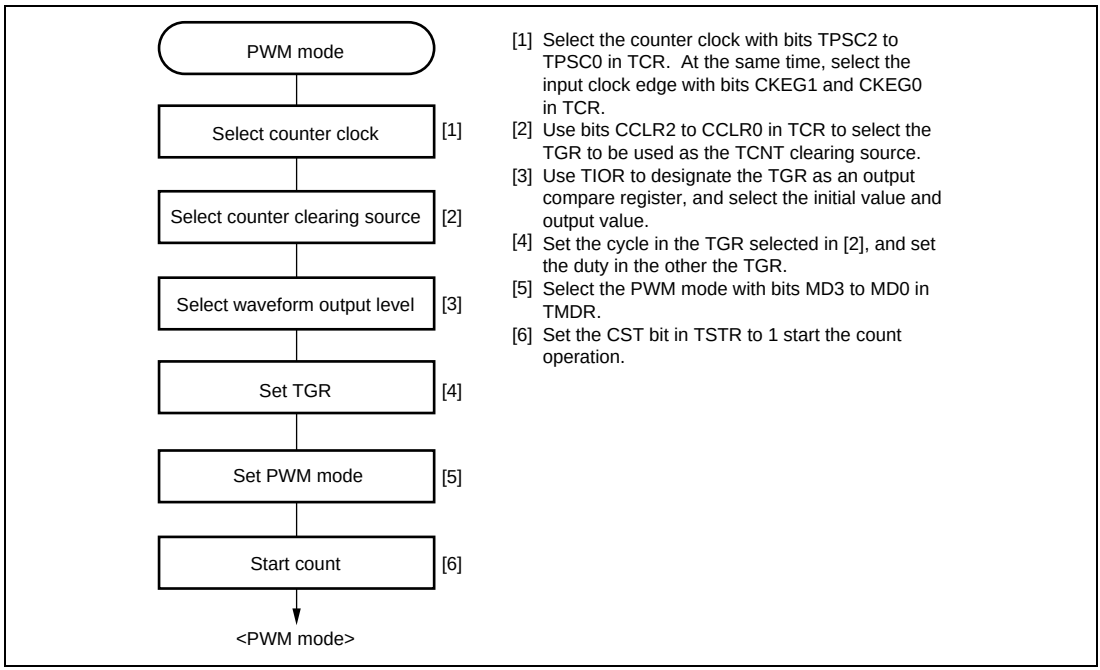



**Figure 10.20 Example of PWM Mode Setting Procedure**

**Examples of PWM Mode Operation:** Figure 10.21 shows an example of PWM mode 1 operation.

In this example, TGRA compare match is set as the TCNT clearing source, 0 is set for the TGRA initial output value and output value, and 1 is set as the TGRB output value.

In this case, the value set in TGRA is used as the period, and the values set in the TGRB registers are used as the duty cycle levels.

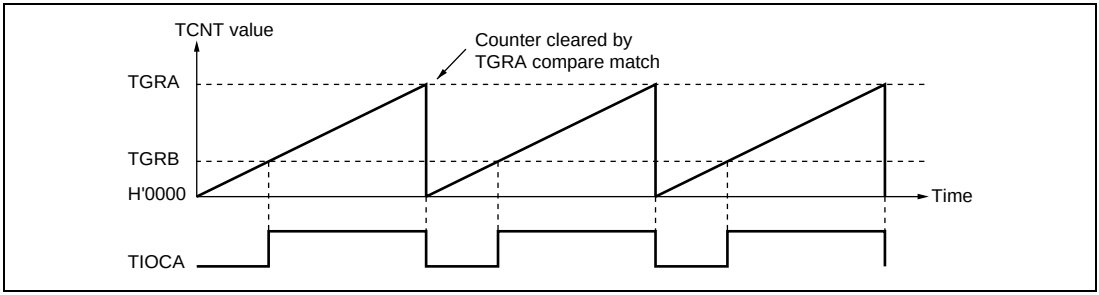



**Figure 10.21 Example of PWM Mode Operation (1)**

Figure 10.22 shows an example of PWM mode 2 operation.

In this example, synchronous operation is designated for channels 0 and 1, TGRB_1 compare match is set as the TCNT clearing source, and 0 is set for the initial output value and 1 for the output value of the other TGR registers (TGRA_0 to TGRD_0, TGRA_1), outputting a 5-phase PWM waveform.

In this case, the value set in TGRB_1 is used as the cycle, and the values set in the other TGRs are used as the duty cycle levels.

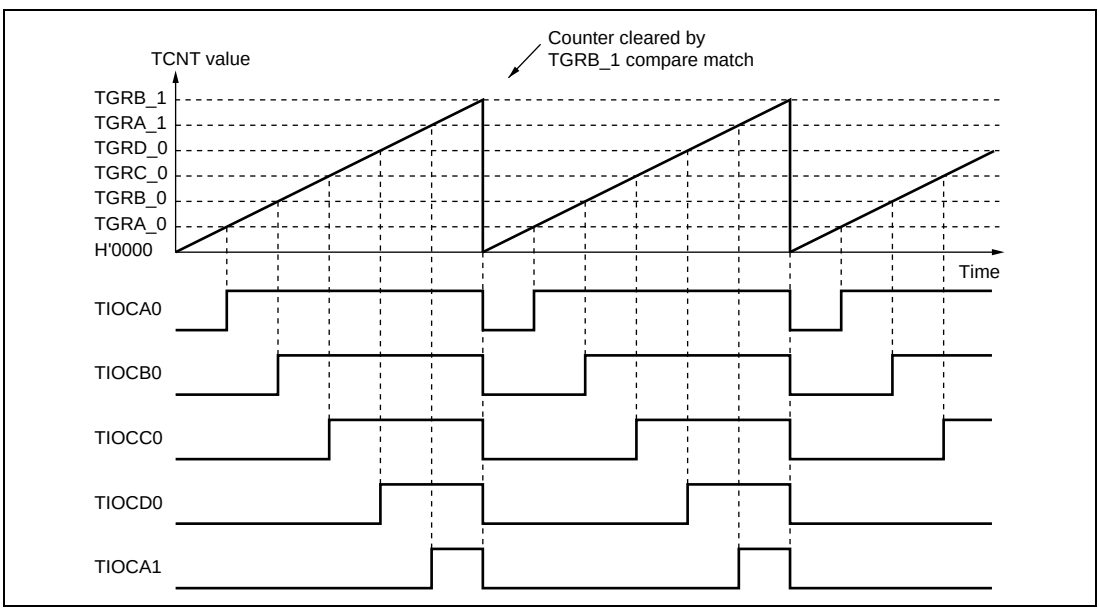



**Figure 10.22 Example of PWM Mode Operation (2)**

Figure 10.23 shows examples of PWM waveform output with 0% duty cycle and 100% duty cycle in PWM mode.

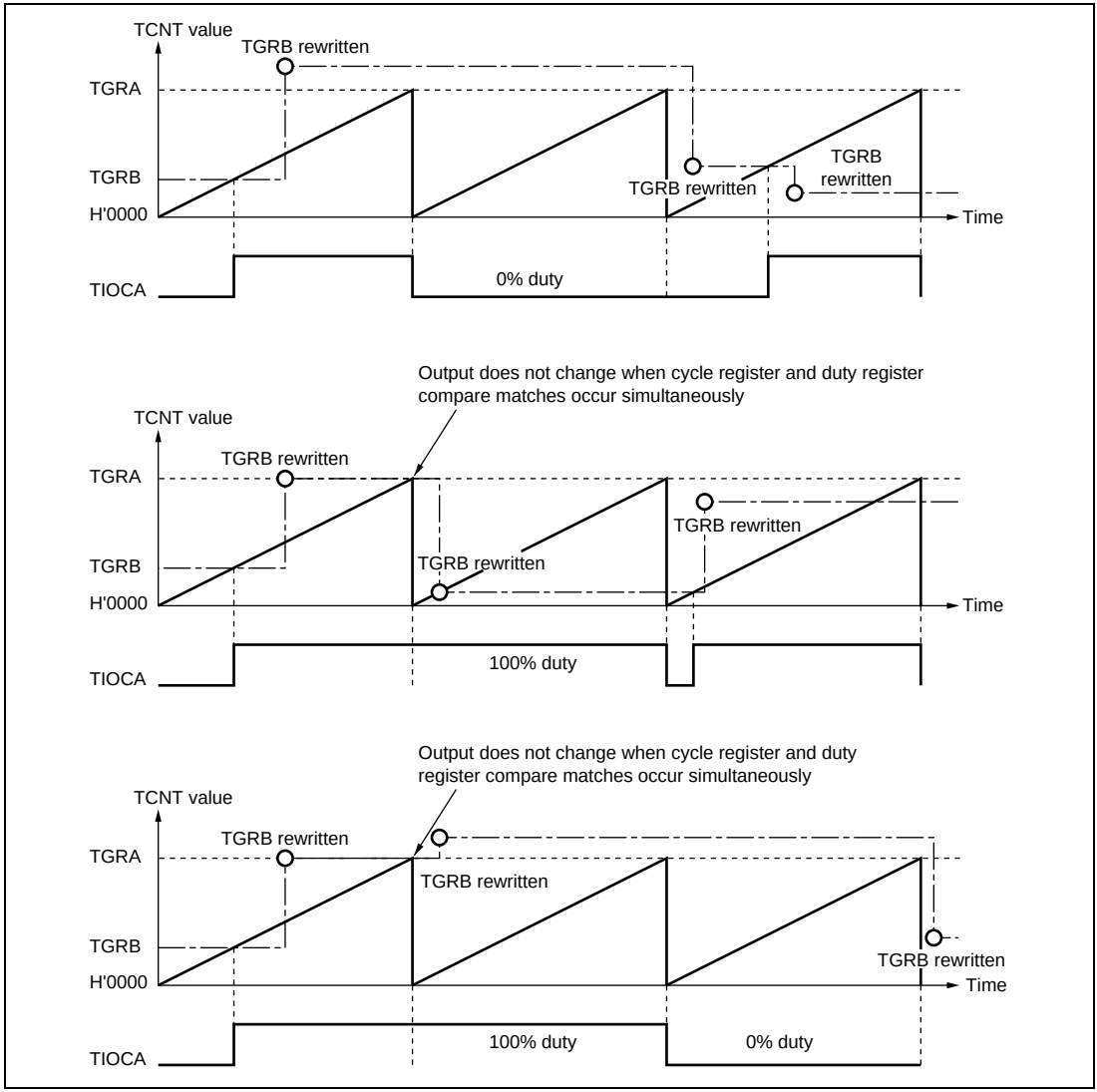



**Figure 10.23 Example of PWM Mode Operation (3)**

### 10.4.6 Phase Counting Mode

In phase counting mode, the phase difference between two external clock inputs is detected and TCNT is incremented/decremented accordingly. This mode can be set for channels 1, 2, 4, and 5.

When phase counting mode is set, an external clock is selected as the counter input clock and TCNT operates as an up/down-counter regardless of the setting of bits TPSC0 to TPSC2 and bits CKEG0 and CKEG1 in TCR. However, the functions of bits CCLR0 and CCLR1 in TCR, and of TIOR, TIER, and TGR, are valid, and input capture/compare match and interrupt functions can be used.

This can be used for two-phase encoder pulse input.

If overflow occurs when TCNT is counting up, the TCFV flag in TSR is set; if underflow occurs when TCNT is counting down, the TCFU flag is set.

The TCFD bit in TSR is the count direction flag. Reading the TCFD flag reveals whether TCNT is counting up or down.

Table 10.31 shows the correspondence between external clock pins and channels.

**Table 10.31 Phase Counting Mode Clock Input Pins**

| | External Clock Pins | |
|---|---|---|
| **Channels** | **A-Phase** | **B-Phase** |
| When channel 1 or 5 is set to phase counting mode | TCLKA | TCLKB |
| When channel 2 or 4 is set to phase counting mode | TCLKC | TCLKD |

**Example of Phase Counting Mode Setting Procedure:** Figure 10.24 shows an example of the phase counting mode setting procedure.

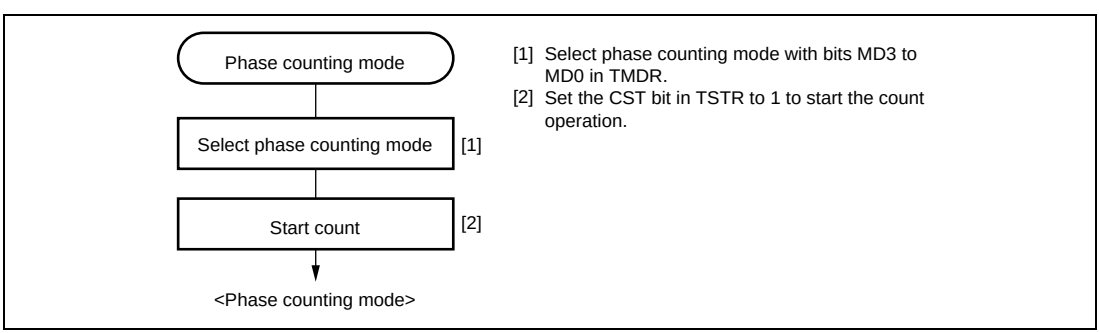



**Figure 10.24 Example of Phase Counting Mode Setting Procedure**

**Examples of Phase Counting Mode Operation:** In phase counting mode, TCNT counts up or down according to the phase difference between two external clocks. There are four modes, according to the count conditions.

1. Phase counting mode 1

   Figure 10.25 shows an example of phase counting mode 1 operation, and table 10.32 summarizes the TCNT up/down-count conditions.

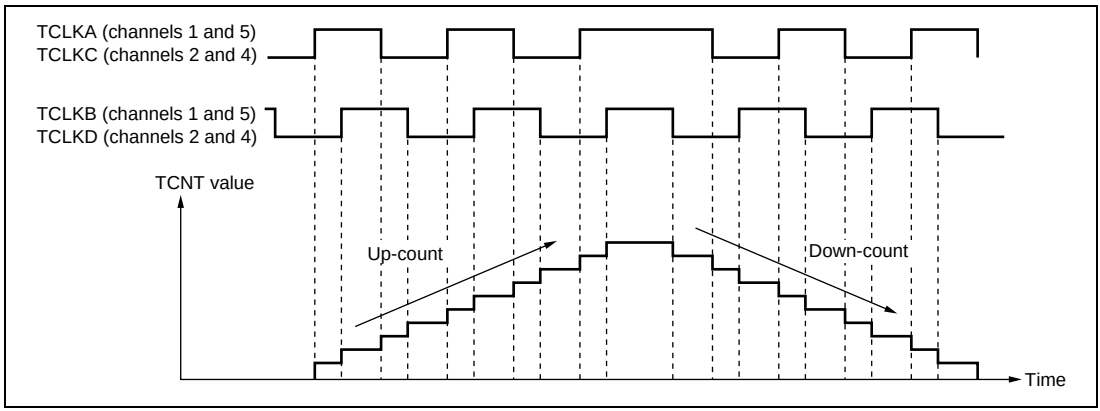



**Figure 10.25 Example of Phase Counting Mode 1 Operation**

**Table 10.32 Up/Down-Count Conditions in Phase Counting Mode 1**

| TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4) | TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4) | Operation |
|---|---|---|
| High level | ⌐↑ | Up-count |
| Low level | ⌐↓ | |
| ⌐↑ | Low level | |
| ⌐↓ | High level | |
| High level | ⌐↓ | Down-count |
| Low level | ⌐↑ | |
| ⌐↑ | High level | |
| ⌐↓ | Low level | |

**Legend**

⌐↑ : Rising edge

⌐↓ : Falling edge

2. Phase counting mode 2

   Figure 10.26 shows an example of phase counting mode 2 operation, and table 10.33 summarizes the TCNT up/down-count conditions.

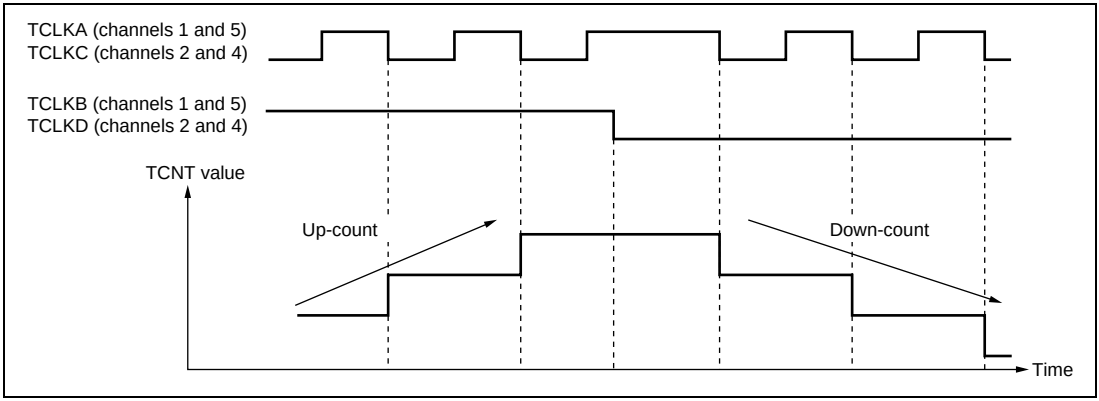



**Figure 10.26 Example of Phase Counting Mode 2 Operation**

**Table 10.33 Up/Down-Count Conditions in Phase Counting Mode 2**

| TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4) | TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4) | Operation |
|---|---|---|
| High level | ⌐↑ | Don't care |
| Low level | ⌐↓ | Don't care |
| ⌐↑ | Low level | Don't care |
| ⌐↓ | High level | Up-count |
| High level | ⌐↓ | Don't care |
| Low level | ⌐↑ | Don't care |
| ⌐↑ | High level | Don't care |
| ⌐↓ | Low level | Down-count |

**Legend**

⌐↑ : Rising edge

⌐↓ : Falling edge

3. Phase counting mode 3

   Figure 10.27 shows an example of phase counting mode 3 operation, and table 10.34 summarizes the TCNT up/down-count conditions.

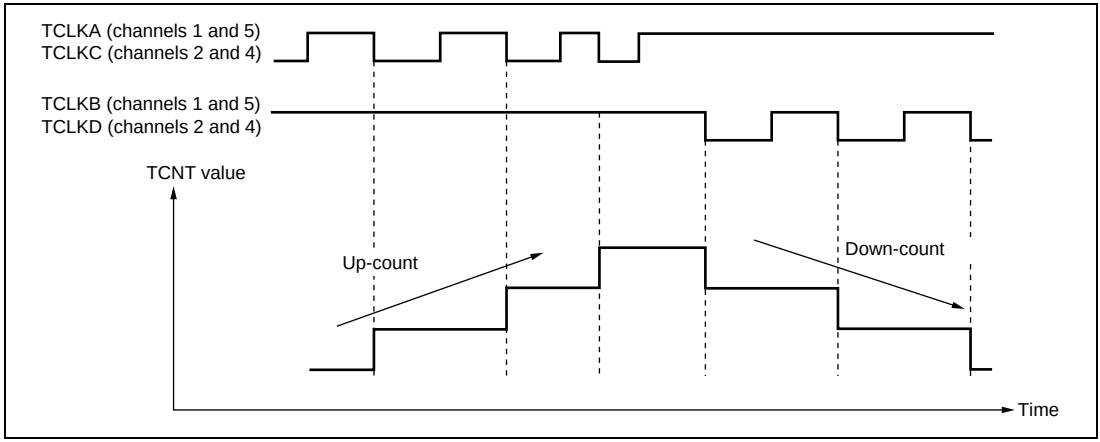



**Figure 10.27 Example of Phase Counting Mode 3 Operation**

**Table 10.34 Up/Down-Count Conditions in Phase Counting Mode 3**

| TCLKA (Channels 1 and 5)<br>TCLKC (Channels 2 and 4) | TCLKB (Channels 1 and 5)<br>TCLKD (Channels 2 and 4) | Operation |
|---|---|---|
| High level | ⌐↑ | Don't care |
| Low level | ⌐↓ | Don't care |
| ⌐↑ | Low level | Don't care |
| ⌐↓ | High level | Up-count |
| High level | ⌐↓ | Down-count |
| Low level | ⌐↑ | Don't care |
| ⌐↑ | High level | Don't care |
| ⌐↓ | Low level | Don't care |

**Legend**

⌐↑ : Rising edge

⌐↓ : Falling edge

4. Phase counting mode 4

   Figure 10.28 shows an example of phase counting mode 4 operation, and table 10.35 summarizes the TCNT up/down-count conditions.

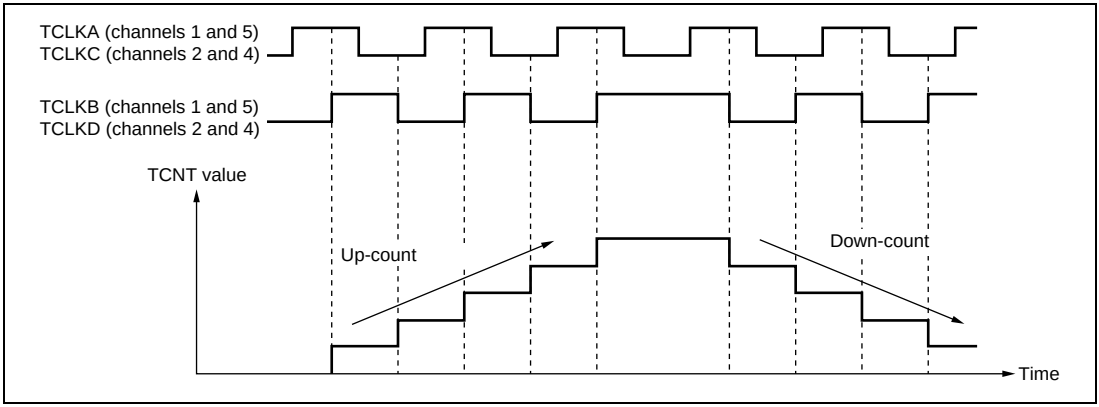



**Figure 10.28 Example of Phase Counting Mode 4 Operation**

**Table 10.35 Up/Down-Count Conditions in Phase Counting Mode 4**

| TCLKA (Channels 1 and 5)<br>TCLKC (Channels 2 and 4) | TCLKB (Channels 1 and 5)<br>TCLKD (Channels 2 and 4) | Operation |
|---|---|---|
| High level | ↑ | Up-count |
| Low level | ↓ | |
| ↑ | Low level | Don't care |
| ↓ | High level | |
| High level | ↓ | Down-count |
| Low level | ↑ | |
| ↑ | High level | Don't care |
| ↓ | Low level | |

**Legend**

↑ : Rising edge

↓ : Falling edge

**Phase Counting Mode Application Example:** Figure 10.29 shows an example in which channel 1 is in phase counting mode, and channel 1 is coupled with channel 0 to input servo motor 2-phase encoder pulses in order to detect position or speed.

Channel 1 is set to phase counting mode 1, and the encoder pulse A-phase and B-phase are input to TCLKA and TCLKB.

Channel 0 operates with TCNT counter clearing by TGRC_0 compare match; TGRA_0 and TGRC_0 are used for the compare match function and are set with the speed control period and position control period. TGRB_0 is used for input capture, with TGRB_0 and TGRD_0 operating in buffer mode. The channel 1 counter input clock is designated as the TGRB_0 input capture source, and the pulse widths of 2-phase encoder 4-multiplication pulses are detected.

TGRA_1 and TGRB_1 for channel 1 are designated for input capture, and channel 0 TGRA_0 and TGRC_0 compare matches are selected as the input capture source and store the up/down-counter values for the control periods.

This procedure enables the accurate detection of position and speed.

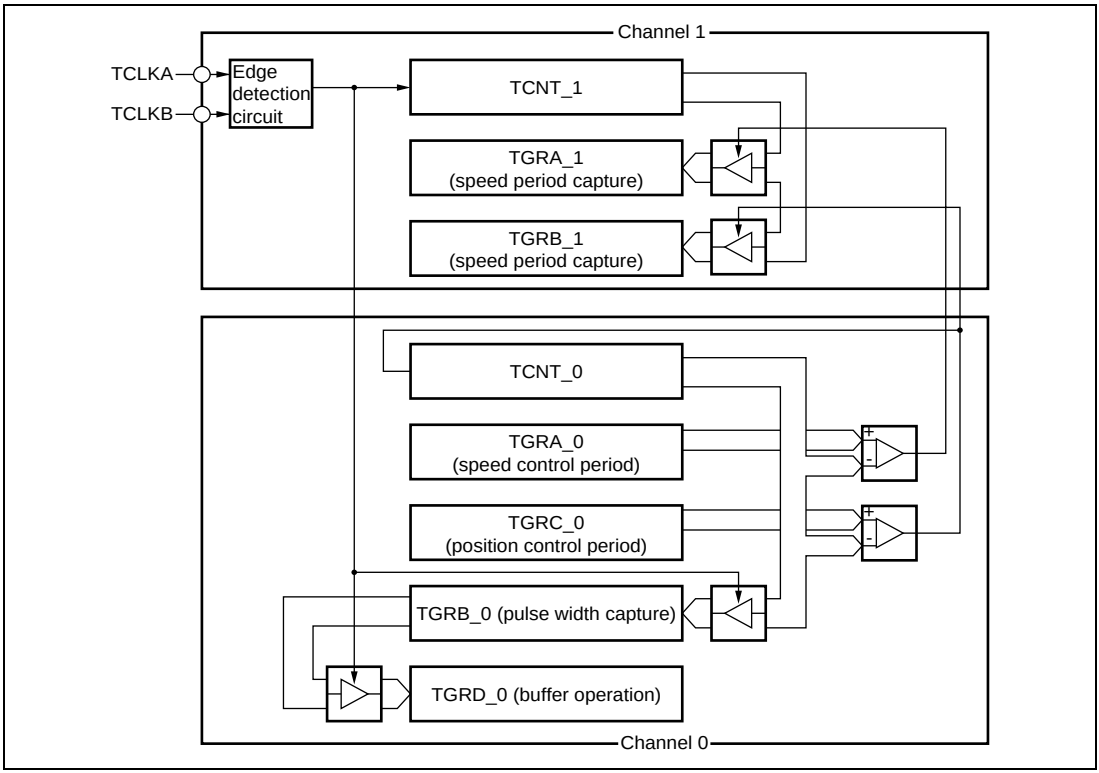



**Figure 10.29 Phase Counting Mode Application Example**

## 10.5 Interrupt Sources

There are three kinds of TPU interrupt source; TGR input capture/compare match, TCNT overflow, and TCNT underflow. Each interrupt source has its own status flag and enable/disabled bit, allowing the generation of interrupt request signals to be enabled or disabled individually.

When an interrupt request is generated, the corresponding status flag in TSR is set to 1. If the corresponding enable/disable bit in TIER is set to 1 at this time, an interrupt is requested. The interrupt request is cleared by clearing the status flag to 0.

Relative channel priorities can be changed by the interrupt controller, however the priority order within a channel is fixed. For details, see section 5, Interrupt Controller.

Table 10.36 lists the TPU interrupt sources.

**Table 10.36 TPU Interrupts**

| Channel | Name | Interrupt Source | Interrupt Flag | DTC Activation |
|---|---|---|---|---|
| 0 | TGIA_0 | TGRA_0 input capture/compare match | TGFA_0 | Possible |
| | TGIB_0 | TGRB_0 input capture/compare match | TGFB_0 | Possible |
| | TGIC_0 | TGRC_0 input capture/compare match | TGFC_0 | Possible |
| | TGID_0 | TGRD_0 input capture/compare match | TGFD_0 | Possible |
| | TCIV_0 | TCNT_0 overflow | TCFV_0 | Not possible |
| 1 | TGIA_1 | TGRA_1 input capture/compare match | TGFA_1 | Possible |
| | TGIB_1 | TGRB_1 input capture/compare match | TGFB_1 | Possible |
| | TCIV_1 | TCNT_1 overflow | TCFV_1 | Not possible |
| | TCIU_1 | TCNT_1 underflow | TCFU_1 | Not possible |
| 2 | TGIA_2 | TGRA_2 input capture/compare match | TGFA_2 | Possible |
| | TGIB_2 | TGRB_2 input capture/compare match | TGFB_2 | Possible |
| | TCIV_2 | TCNT_2 overflow | TCFV_2 | Not possible |
| | TCIU_2 | TCNT_2 underflow | TCFU_2 | Not possible |
| 3 | TGIA_3 | TGRA_3 input capture/compare match | TGFA_3 | Possible |
| | TGIB_3 | TGRB_3 input capture/compare match | TGFB_3 | Possible |
| | TGIC_3 | TGRC_3 input capture/compare match | TGFC_3 | Possible |
| | TGID_3 | TGRD_3 input capture/compare match | TGFD_3 | Possible |
| | TCIV_3 | TCNT_3 overflow | TCFV_3 | Not possible |
| 4 | TGIA_4 | TGRA_4 input capture/compare match | TGFA_4 | Possible |
| | TGIB_4 | TGRB_4 input capture/compare match | TGFB_4 | Possible |
| | TCIV_4 | TCNT_4 overflow | TCFV_4 | Not possible |
| | TCIU_4 | TCNT_4 underflow | TCFU_4 | Not possible |
| 5 | TGIA_5 | TGRA_5 input capture/compare match | TGFA_5 | Possible |
| | TGIB_5 | TGRB_5 input capture/compare match | TGFB_5 | Possible |
| | TCIV_5 | TCNT_5 overflow | TCFV_5 | Not possible |
| | TCIU_5 | TCNT_5 underflow | TCFU_5 | Not possible |

Note:* This table shows the initial state immediately after a reset. The relative channel priorities can be changed by the interrupt controller.

**Input Capture/Compare Match Interrupt:** An interrupt is requested if the TGIE bit in TIER is set to 1 when the TGF flag in TSR is set to 1 by the occurrence of a TGR input capture/compare match on a particular channel. The interrupt request is cleared by clearing the TGF flag to 0. The TPU has 16 input capture/compare match interrupts, four each for channels 0 and 3, and two each for channels 1, 2, 4, and 5.

**Overflow Interrupt:** An interrupt is requested if the TCIEV bit in TIER is set to 1 when the TCFV flag in TSR is set to 1 by the occurrence of TCNT overflow on a channel. The interrupt request is cleared by clearing the TCFV flag to 0. The TPU has six overflow interrupts, one for each channel.

**Underflow Interrupt:** An interrupt is requested if the TCIEU bit in TIER is set to 1 when the TCFU flag in TSR is set to 1 by the occurrence of TCNT underflow on a channel. The interrupt request is cleared by clearing the TCFU flag to 0. The TPU has four underflow interrupts, one each for channels 1, 2, 4, and 5.

## 10.6 DTC Activation

The DTC can be activated by the TGR input capture/compare match interrupt for a channel. For details, see section 8, Data Transfer Controller (DTC).

A total of 16 TPU input capture/compare match interrupts can be used as DTC activation sources, four each for channels 0 and 3, and two each for channels 1, 2, 4, and 5.

## 10.7 A/D Converter Activation

The A/D converter can be activated by the TGRA input capture/compare match for a channel.

If the TTGE bit in TIER is set to 1 when the TGFA flag in TSR is set to 1 by the occurrence of a TGRA input capture/compare match on a particular channel, a request to begin A/D conversion is sent to the A/D converter. If the TPU conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is begun.

In the TPU, a total of six TGRA input capture/compare match interrupts can be used as A/D converter conversion start sources, one for each channel.

## 10.8 Operation Timing

### 10.8.1 Input/Output Timing

**TCNT Count Timing:** Figure 10.30 shows TCNT count timing in internal clock operation, and figure 10.31 shows TCNT count timing in external clock operation.

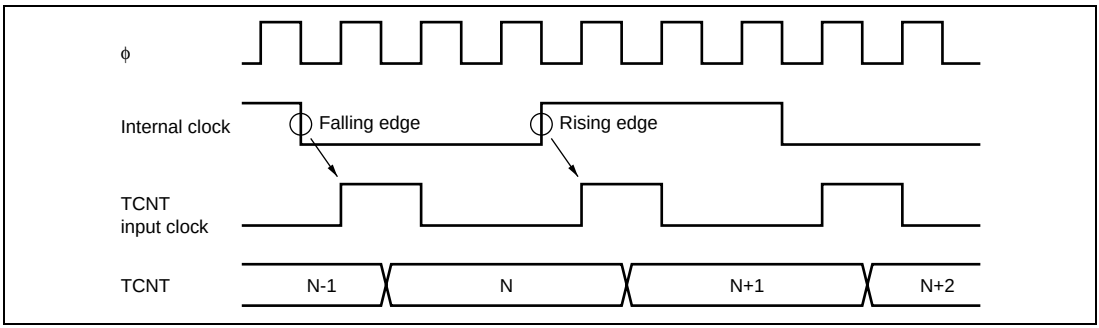



**Figure 10.30 Count Timing in Internal Clock Operation**

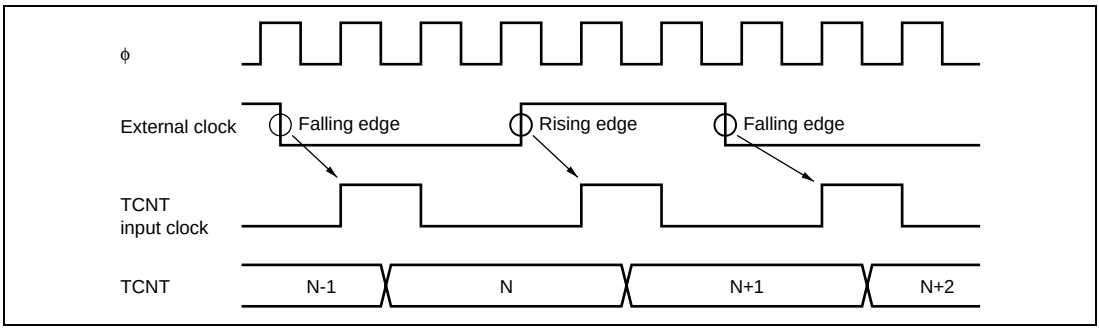



**Figure 10.31 Count Timing in External Clock Operation**

**Output Compare Output Timing:** A compare match signal is generated in the final state in which TCNT and TGR match (the point at which the count value matched by TCNT is updated). When a compare match signal is generated, the output value set in TIOR is output at the output compare output pin. After a match between TCNT and TGR, the compare match signal is not generated until the TCNT input clock is generated.

Figure 10.32 shows output compare output timing.

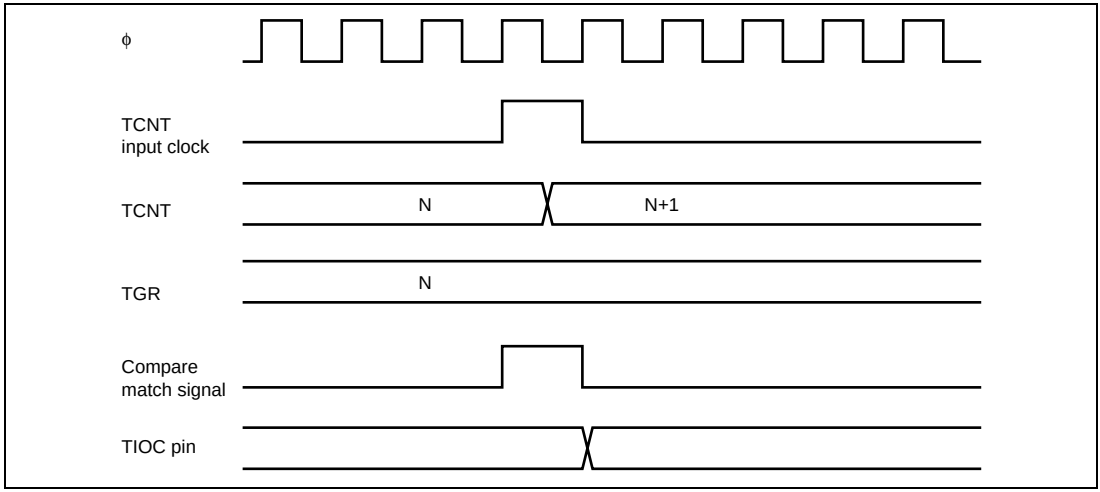



**Figure 10.32 Output Compare Output Timing**

**Input Capture Signal Timing:** Figure 10.33 shows input capture signal timing.

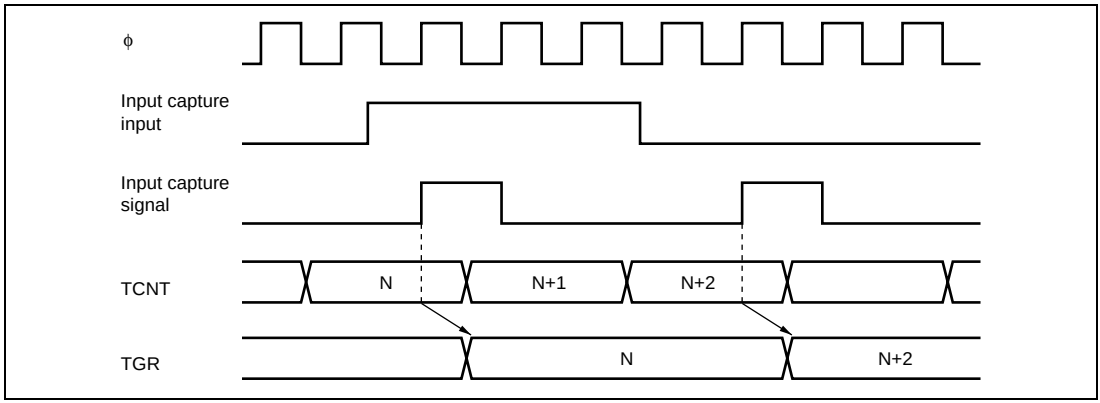



**Figure 10.33 Input Capture Input Signal Timing**

**Timing for Counter Clearing by Compare Match/Input Capture:** Figure 10.34 shows the timing when counter clearing on compare match is specified, and figure 10.35 shows the timing when counter clearing on input capture is specified.

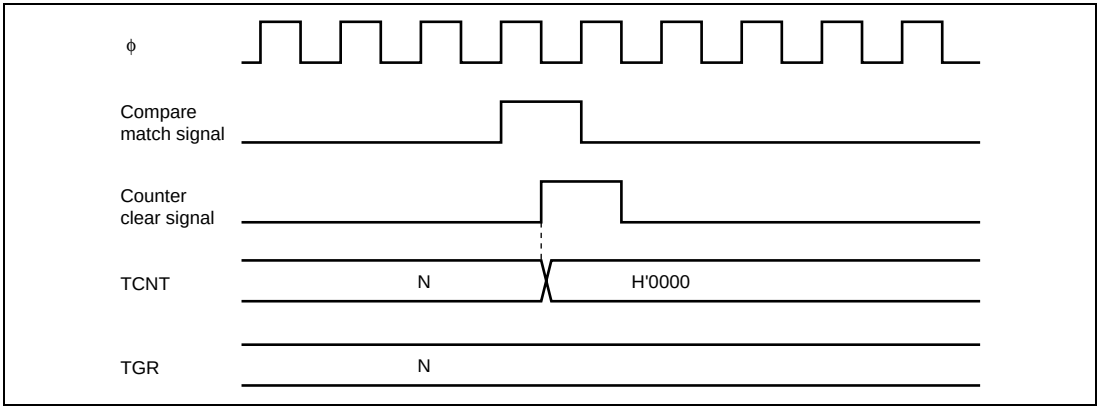



**Figure 10.34 Counter Clear Timing (Compare Match)**

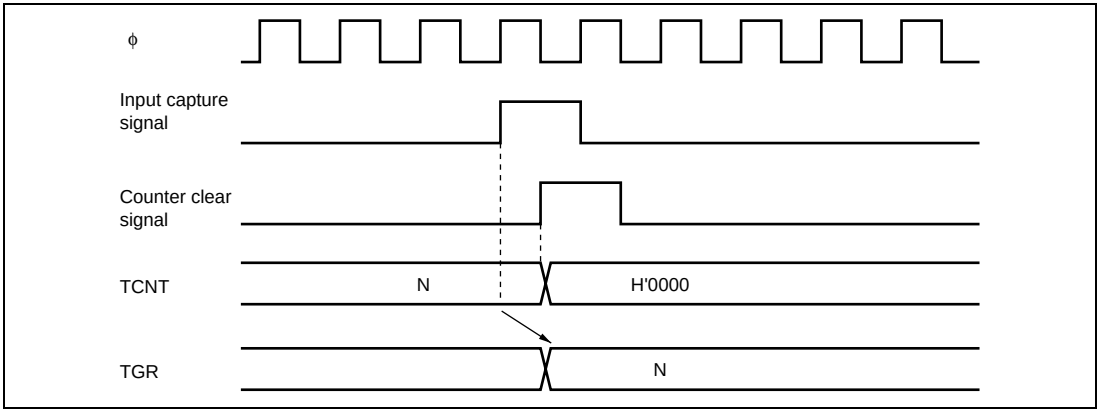



**Figure 10.35 Counter Clear Timing (Input Capture)**

**Buffer Operation Timing:** Figures 10.36 and 10.37 show the timing in buffer operation.

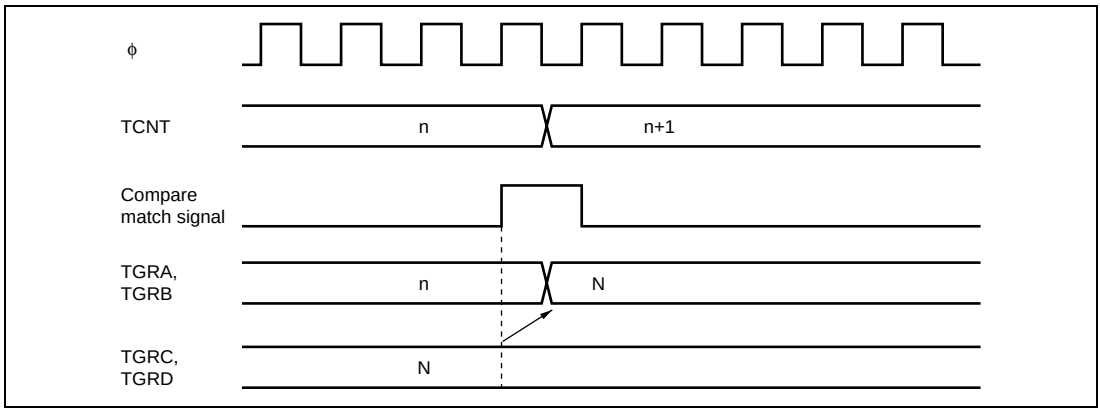



**Figure 10.36 Buffer Operation Timing (Compare Match)**

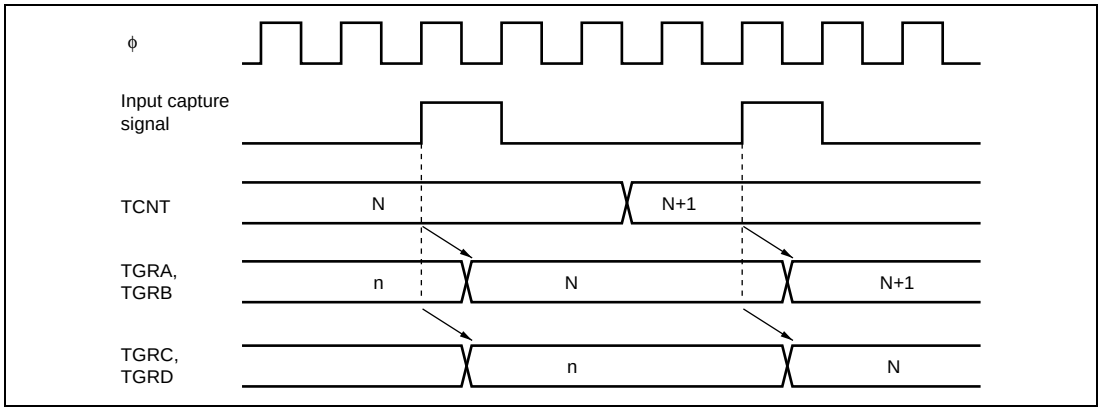



**Figure 10.37 Buffer Operation Timing (Input Capture)**

### 10.8.2 Interrupt Signal Timing

**TGF Flag Setting Timing in Case of Compare Match:** Figure 10.38 shows the timing for setting of the TGF flag in TSR on compare match, and TGI interrupt request signal timing.

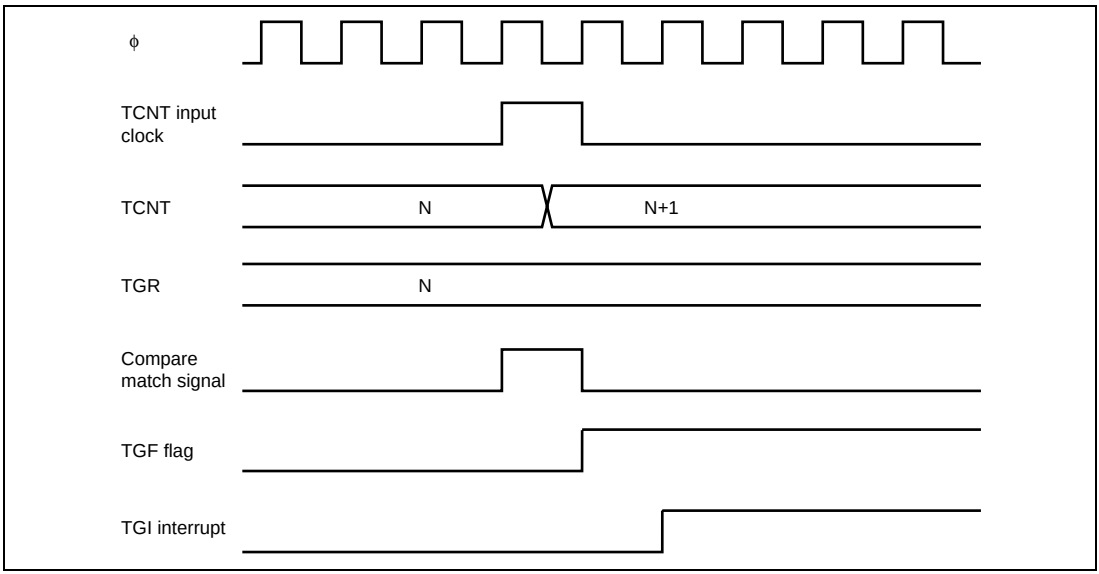



**Figure 10.38 TGI Interrupt Timing (Compare Match)**

**TGF Flag Setting Timing in Case of Input Capture:** Figure 10.39 shows the timing for setting of the TGF flag in TSR on input capture, and TGI interrupt request signal timing.

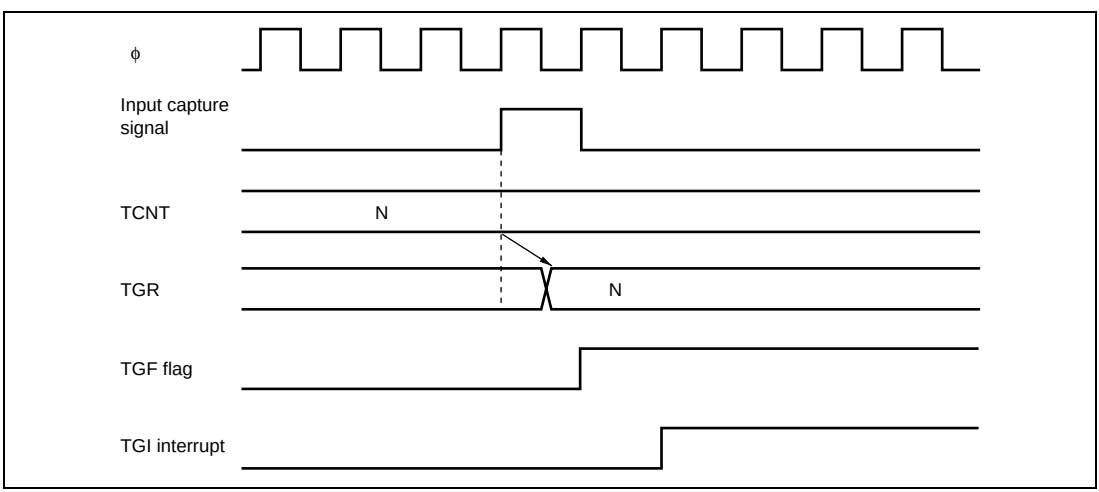



**Figure 10.39 TGI Interrupt Timing (Input Capture)**

**TCFV Flag/TCFU Flag Setting Timing:** Figure 10.40 shows the timing for setting of the TCFV flag in TSR on overflow, and TCIV interrupt request signal timing.

Figure 10.41 shows the timing for setting of the TCFU flag in TSR on underflow, and TCIU interrupt request signal timing.

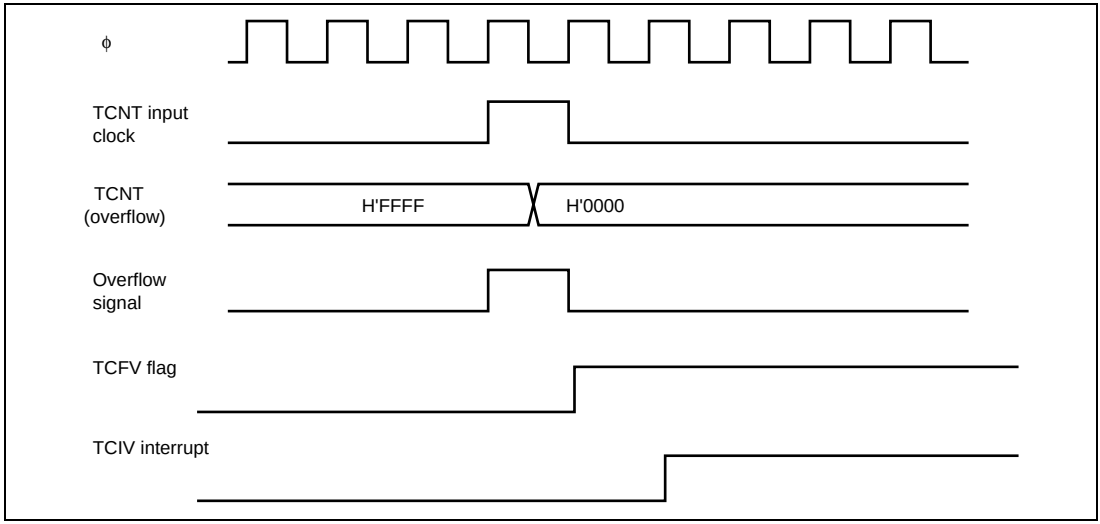



**Figure 10.40 TCIV Interrupt Setting Timing**

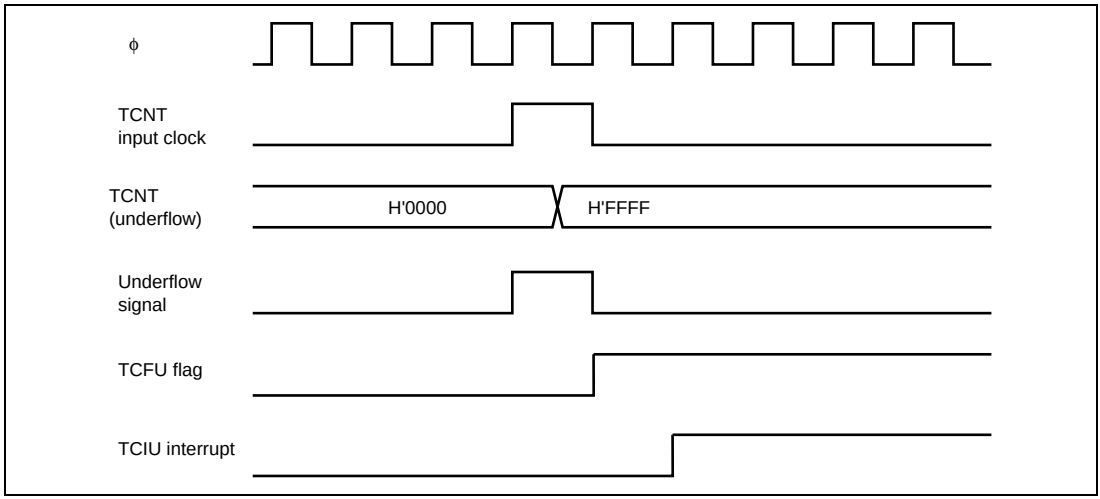



**Figure 10.41 TCIU Interrupt Setting Timing**

**Status Flag Clearing Timing:** After a status flag is read as 1 by the CPU, it is cleared by writing 0 to it. When the DTC is activated, the flag is cleared automatically. Figure 10.42 shows the timing for status flag clearing by the CPU, and figure 10.43 shows the timing for status flag clearing by the DTC.

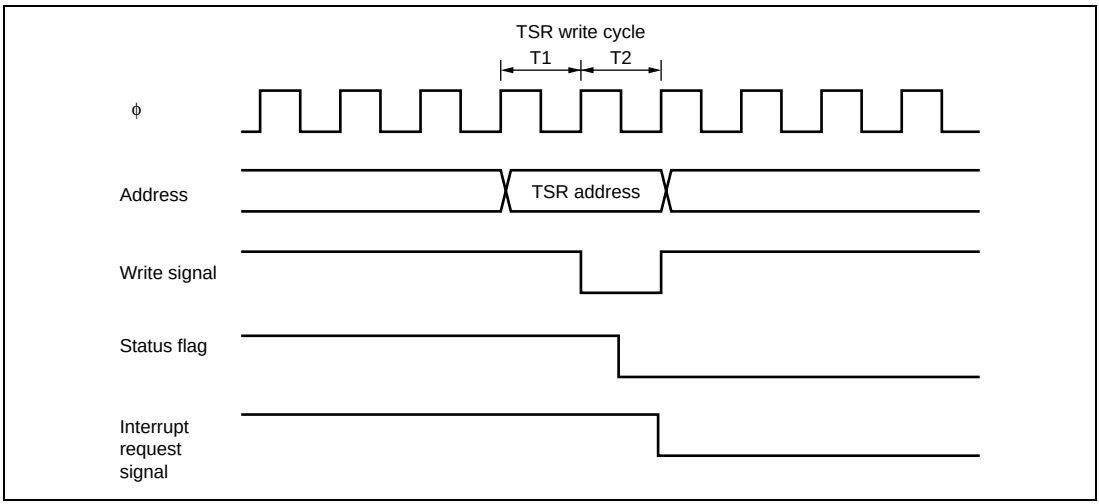



**Figure 10.42 Timing for Status Flag Clearing by CPU**

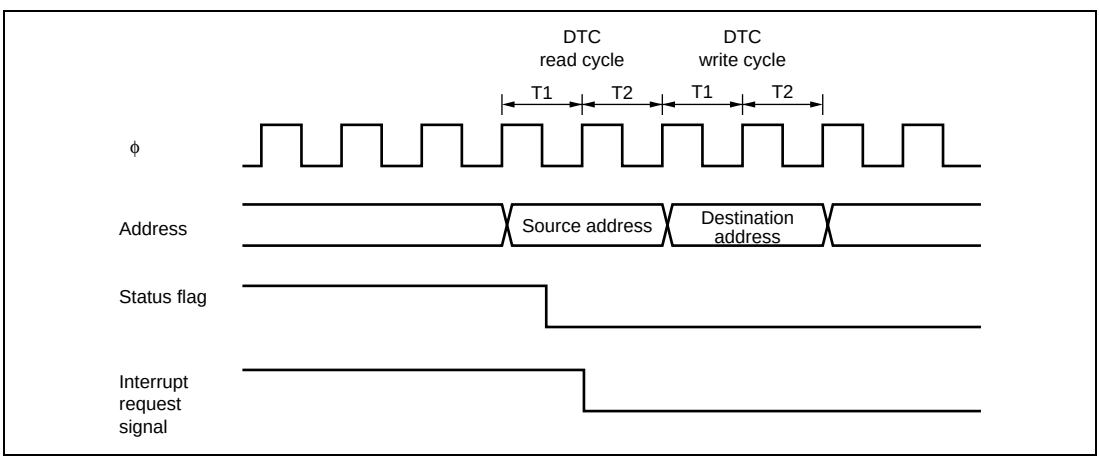



**Figure 10.43 Timing for Status Flag Clearing by DTC Activation**

## 10.9 Usage Notes

### 10.9.1 Module Stop Mode Setting

TPU operation can be disabled or enabled using the module stop control register. The initial setting is for TPU operation to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 10.9.2 Input Clock Restrictions

The input clock pulse width must be at least 1.5 states in the case of single-edge detection, and at least 2.5 states in the case of both-edge detection. The TPU will not operate properly at narrower pulse widths.

In phase counting mode, the phase difference and overlap between the two input clocks must be at least 1.5 states, and the pulse width must be at least 2.5 states. Figure 10.44 shows the input clock conditions in phase counting mode.

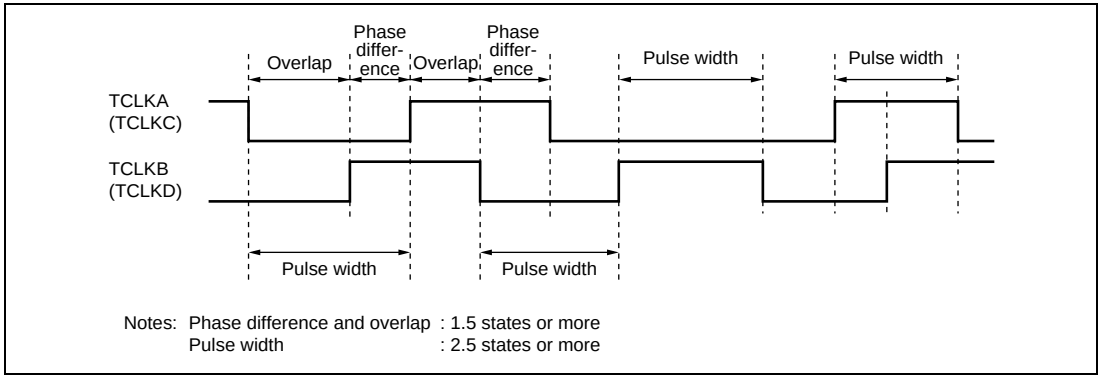



**Figure 10.44 Phase Difference, Overlap, and Pulse Width in Phase Counting Mode**

### 10.9.3 Caution on Period Setting

When counter clearing on compare match is set, TCNT is cleared in the final state in which it matches the TGR value (the point at which the count value matched by TCNT is updated). Consequently, the actual counter frequency is given by the following formula:

$$f = \frac{\phi}{(N + 1)}$$

Where f : Counter frequency
$\phi$ : Operating frequency
N : TGR set value

### 10.9.4 Conflict between TCNT Write and Clear Operations

If the counter clear signal is generated in the T2 state of a TCNT write cycle, TCNT clearing takes precedence and the TCNT write is not performed.

Figure 10.45 shows the timing in this case.

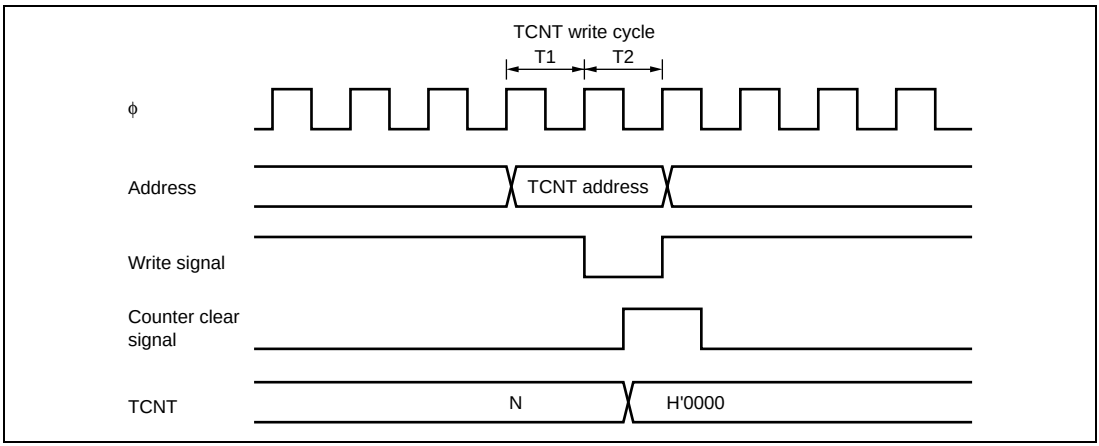



**Figure 10.45 Conflict between TCNT Write and Clear Operations**

### 10.9.5 Conflict between TCNT Write and Increment Operations

If incrementing occurs in the T2 state of a TCNT write cycle, the TCNT write takes precedence and TCNT is not incremented.

Figure 10.46 shows the timing in this case.

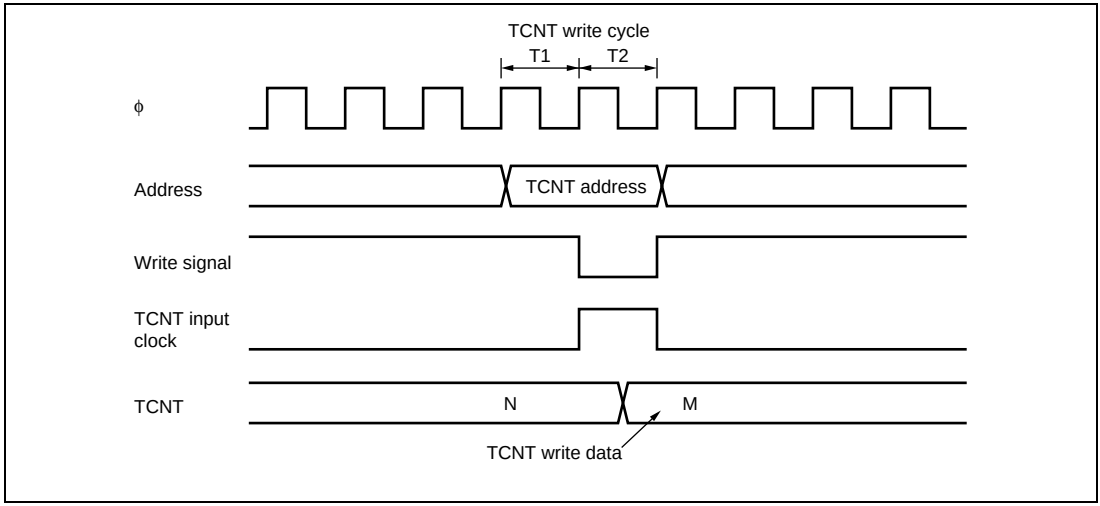



**Figure 10.46 Conflict between TCNT Write and Increment Operations**

### 10.9.6 Conflict between TGR Write and Compare Match

If a compare match occurs in the T2 state of a TGR write cycle, the TGR write takes precedence and the compare match signal is inhibited. A compare match does not occur even if the previous value is written.

Figure 10.47 shows the timing in this case.

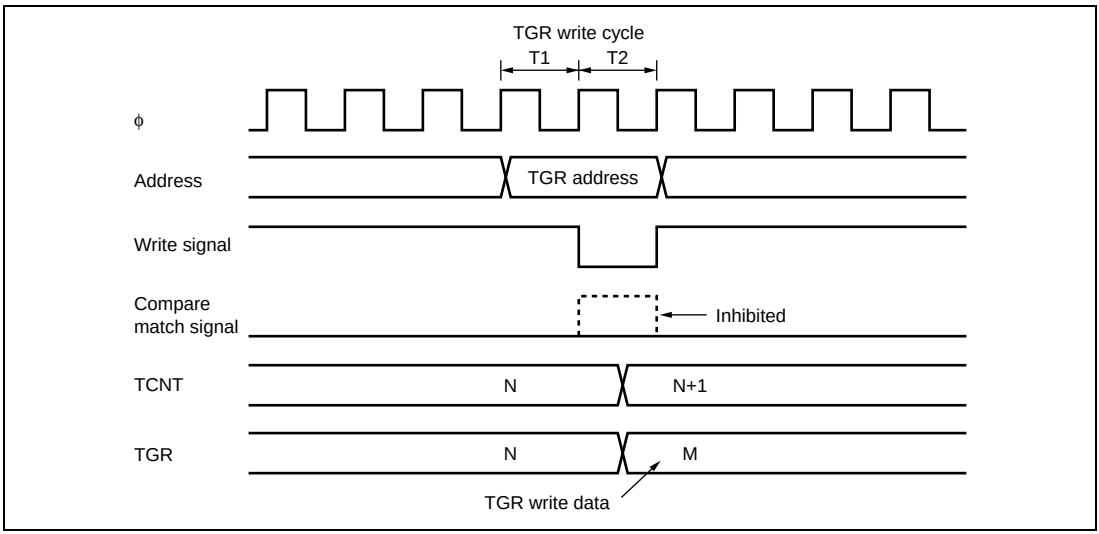



**Figure 10.47 Conflict between TGR Write and Compare Match**

### 10.9.7 Conflict between Buffer Register Write and Compare Match

If a compare match occurs in the T2 state of a TGR write cycle, the data that is transferred to TGR by the buffer operation will be that in the buffer prior to the write.

Figure 10.48 shows the timing in this case.

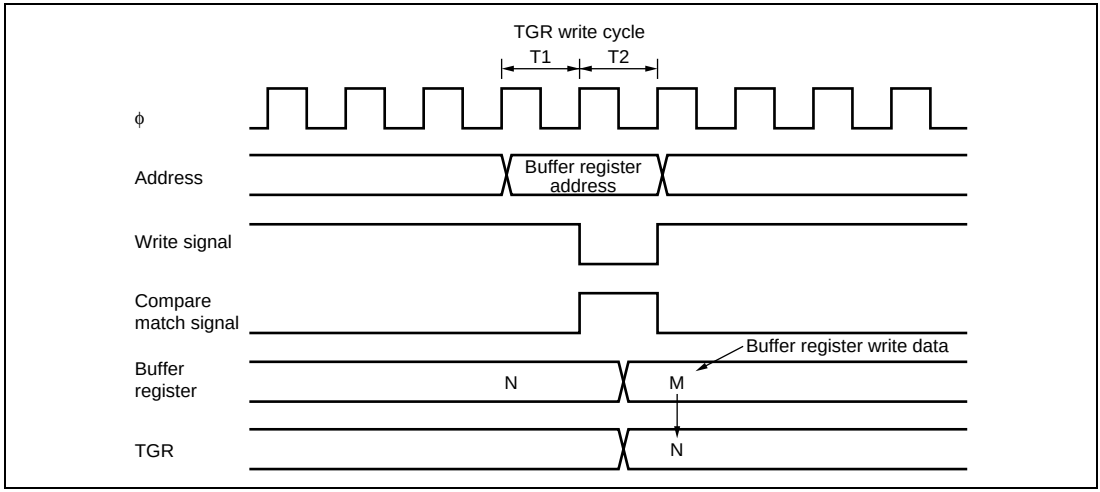



**Figure 10.48 Conflict between Buffer Register Write and Compare Match**

### 10.9.8 Conflict between TGR Read and Input Capture

If an input capture signal is generated in the T1 state of a TGR read cycle, the data that is read will be that in the buffer after input capture transfer.

Figure 10.49 shows the timing in this case.

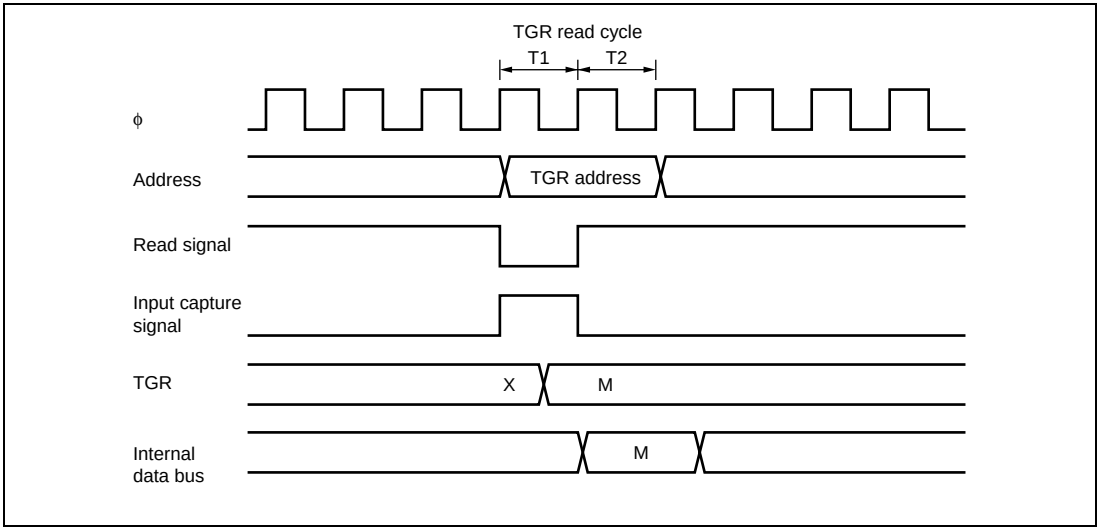



**Figure 10.49 Conflict between TGR Read and Input Capture**

### 10.9.9 Conflict between TGR Write and Input Capture

If an input capture signal is generated in the T2 state of a TGR write cycle, the input capture operation takes precedence and the write to TGR is not performed.

Figure 10.50 shows the timing in this case.

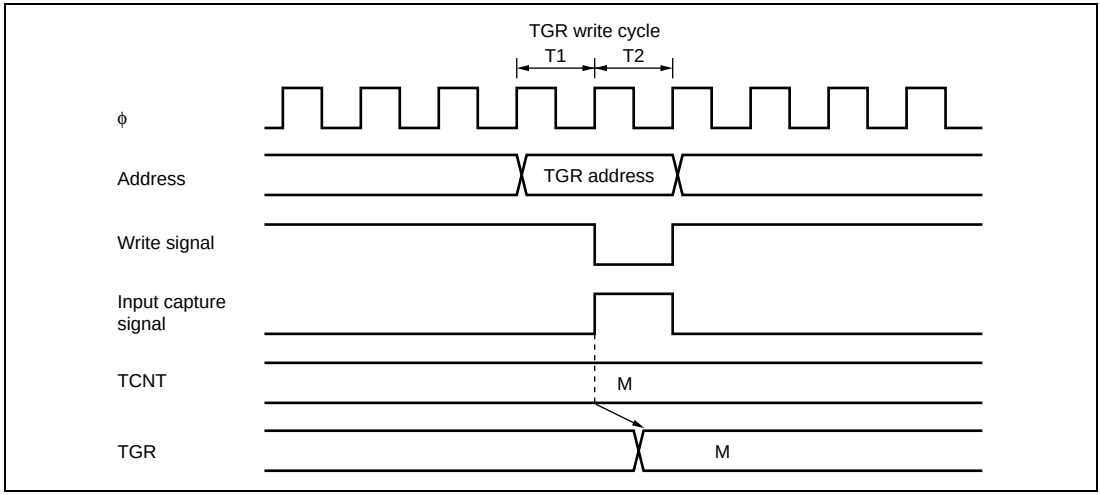



**Figure 10.50 Conflict between TGR Write and Input Capture**

### 10.9.10 Conflict between Buffer Register Write and Input Capture

If an input capture signal is generated in the T2 state of a buffer register write cycle, the buffer operation takes precedence and the write to the buffer register is not performed.

Figure 10.51 shows the timing in this case.

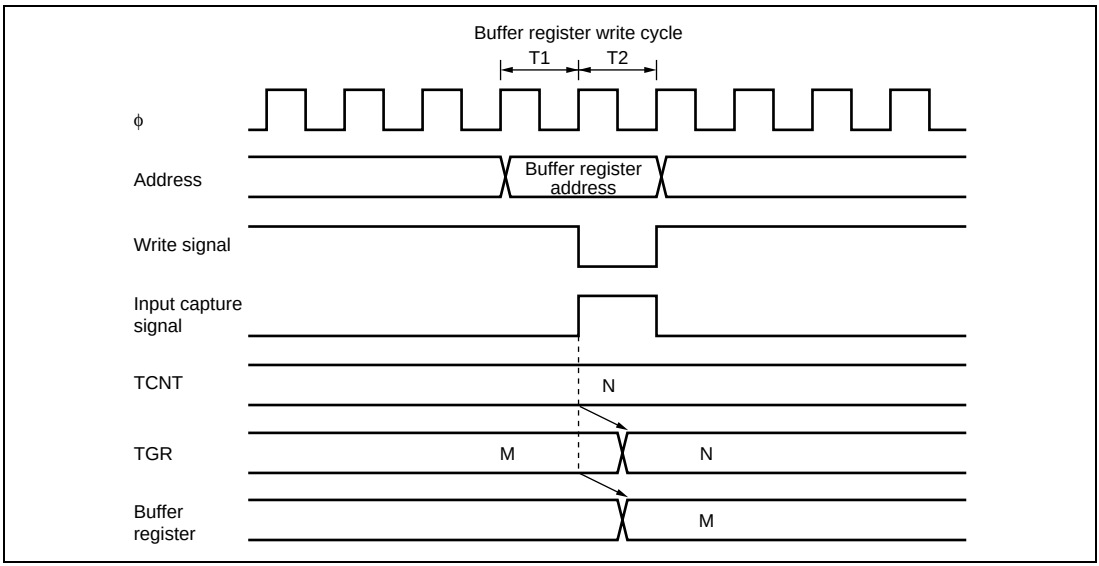



**Figure 10.51 Conflict between Buffer Register Write and Input Capture**

### 10.9.11 Conflict between Overflow/Underflow and Counter Clearing

If overflow/underflow and counter clearing occur simultaneously, the TCFV/TCFU flag in TSR is not set and TCNT clearing takes precedence.

Figure 10.52 shows the operation timing when a TGR compare match is specified as the clearing source, and when H′FFFF is set in TGR.

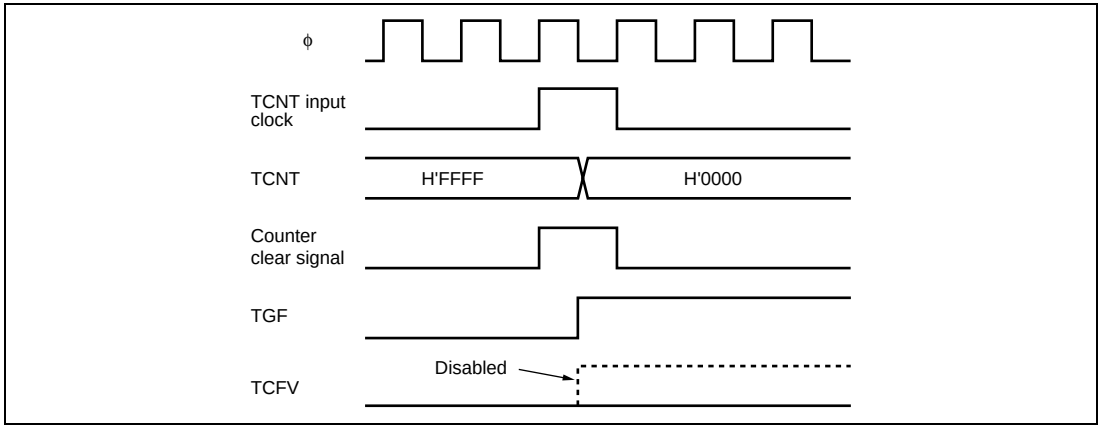



**Figure 10.52 Conflict between Overflow and Counter Clearing**

### 10.9.12 Conflict between TCNT Write and Overflow/Underflow

If there is an up-count or down-count in the T2 state of a TCNT write cycle, and overflow/underflow occurs, the TCNT write takes precedence and the TCFV/TCFU flag in TSR is not set.

Figure 10.53 shows the operation timing when there is conflict between TCNT write and overflow.

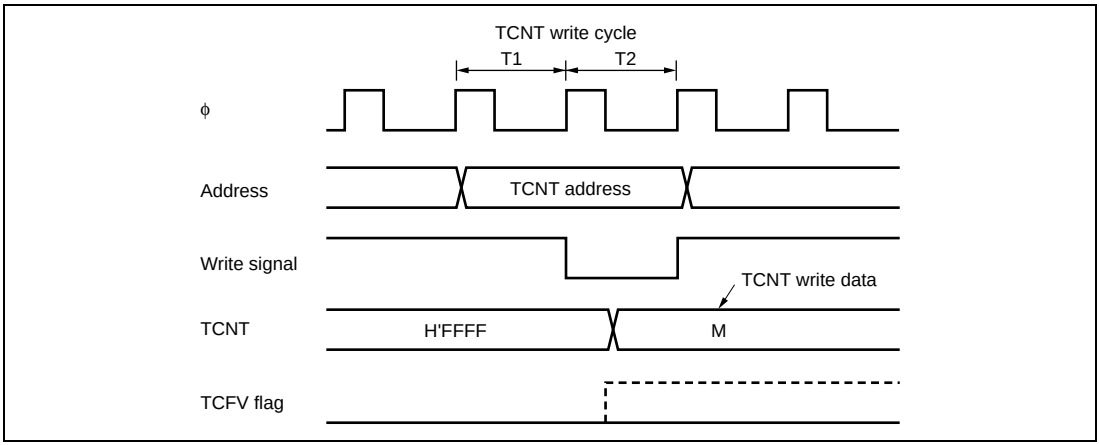



**Figure 10.53 Conflict between TCNT Write and Overflow**

### 10.9.13 Multiplexing of I/O Pins

In this LSI, the TCLKA input pin is multiplexed with the TIOCC0 I/O pin, the TCLKB input pin with the TIOCD0 I/O pin, the TCLKC input pin with the TIOCB1 I/O pin, and the TCLKD input pin with the TIOCB2 I/O pin. When an external clock is input, compare match output should not be performed from a multiplexed pin.

### 10.9.14 Interrupts in Module Stop Mode

If module stop mode is entered when an interrupt has been requested, it will not be possible to clear the CPU interrupt source or the DTC activation source. Interrupts should therefore be disabled before entering module stop mode.

# Section 11 8-Bit Timers

This LSI has an on-chip 8-bit timer module with four channels operating on the basis of an 8-bit counter.

The 8-bit timer module can be used to count external events and be used as a multifunction timer in a variety of applications, such as generation of counter reset, interrupt requests, and pulse output with an arbitrary duty cycle using a compare-match signal with two registers.

## 11.1 Features

- Selection of clock sources
  Selected from three internal clocks (ϕ/8, ϕ/64, and ϕ/8192) and an external clock.
- Selection of three ways to clear the counters
  The counters can be cleared on compare-match A or B, or by an external reset signal.
- Timer output controlled by two compare-match signals
  The timer output signal in each channel is controlled by two independent compare-match signals, enabling the timer to be used for various applications, such as the generation of pulse output or PWM output with an arbitrary duty cycle.
- Cascading of the two channels
  - Cascading of TMR_0 and TMR_1
    The module can operate as a 16-bit timer using TMR_0 as the upper half and TMR_1 as the lower half (16-bit count mode).
    TMR_1 can be used to count TMR_0 compare-match occurrences (compare-match count mode).
  - Cascading of TMR_2 and TMR_3
    The module can operate as a 16-bit timer using TMR_2 as the upper half and TMR_3 as the lower half (16-bit count mode).
    TMR_3 can be used to count TMR_2 compare-match occurrences (compare-match count mode).
- Multiple interrupt sources for each channel
  Two compare-match interrupts and one overflow interrupt can be requested independently.
- Generation of A/D conversion start trigger
  Channel 0 compare-match signal can be used as the A/D conversion start trigger.
- Module stop mode can be set
  At initialization, the 8-bit timer operation is halted. Register access is enabled by canceling the module stop mode.

Figure 11.1 shows a block diagram of the 8-bit timer module (TMR_0 and TMR_1).

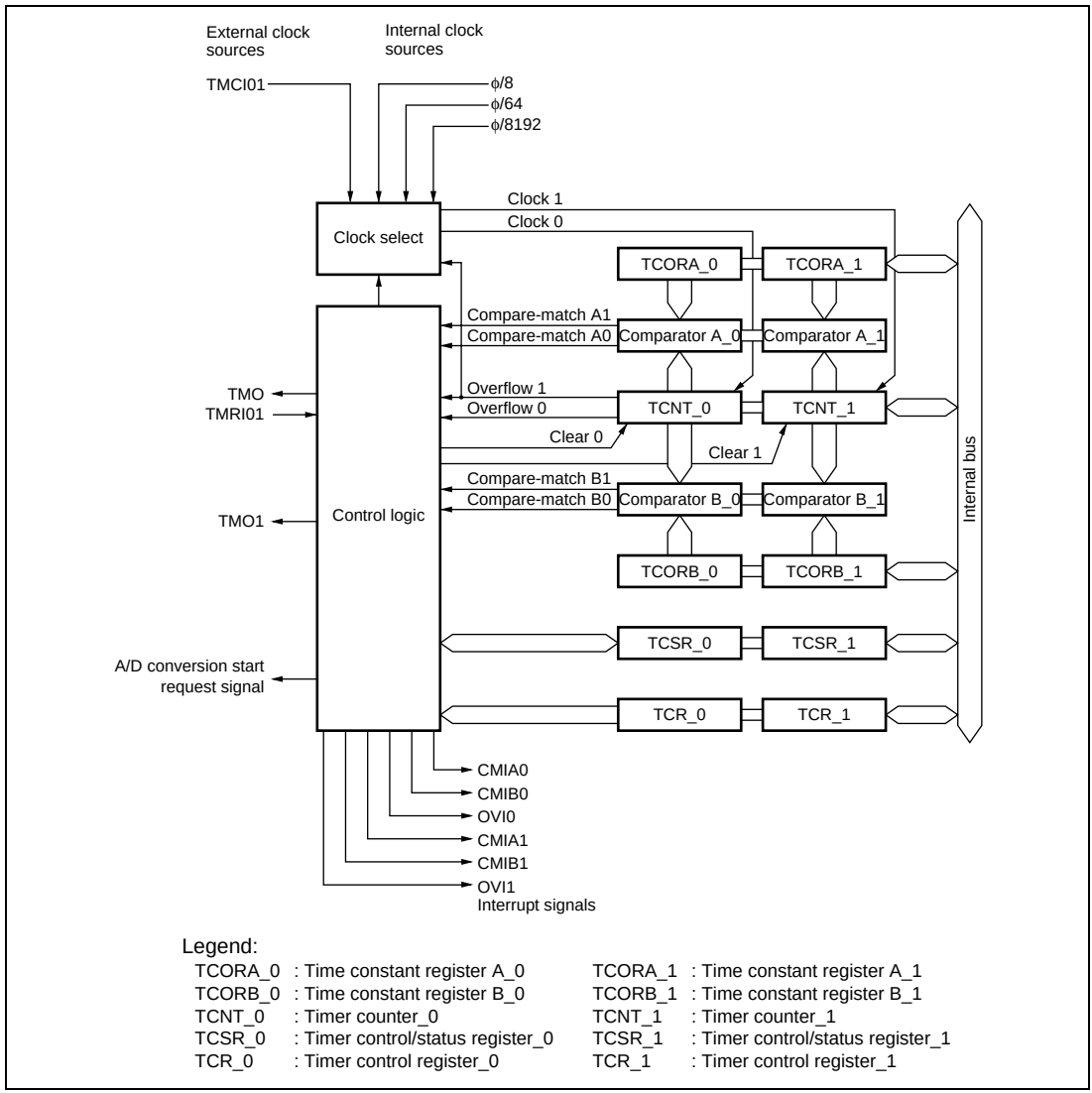



**Figure 11.1 Block Diagram of 8-Bit Timer Module**

## 11.2 Input/Output Pins

Table 11.1 summarizes the input and output pins of the 8-bit timer module.

**Table 11.1 Pin Configuration**

| Channel | Name | Symbol | I/O | Function |
|---|---|---|---|---|
| 0 | Timer output | TMO0 | Output | Output controlled by compare-match |
| 1 | Timer output | TMO1 | Output | Output controlled by compare-match |
| Common to 0 and 1 | Timer clock input | TMCI01 | Input | External clock input for the counter |
| | Timer reset input | TMRI01 | Input | External reset input for the counter |
| 2 | Timer output | TMO2 | Output | Output controlled by compare-match |
| 3 | Timer output | TMO3 | Output | Output controlled by compare-match |
| Common to 2 and 3 | Timer clock input | TMCI23 | Input | External clock input for the counter |
| | Timer reset input | TMRI23 | Input | External reset input for the counter |

## 11.3 Register Descriptions

The 8-bit timer has the following registers. For details on the module stop register, refer to section 21.1.2, Module Stop Registers A to C (MSTPCRA to MSTPCRC).

- Timer counter_0 (TCNT_0)
- Time constant register A_0 (TCORA_0)
- Time constant register B_0 (TCORB_0)
- Timer control register_0 (TCR_0)
- Timer control/status register_0 (TCSR_0)
- Timer counter_1 (TCNT_1)
- Time constant register A_1 (TCORA_1)
- Time constant register B_1 (TCORB_1)
- Timer control register_1 (TCR_1)
- Timer control/status register_1 (TCSR_1)
- Timer counter_2 (TCNT_2)
- Time constant register A_2 (TCORA_2)
- Time constant register B_2 (TCORB_2)
- Timer control register_2 (TCR_2)
- Timer control/status register_2 (TCSR_2)
- Timer counter_3 (TCNT_3)
- Time constant register A_3 (TCORA_3)
- Time constant register B_3 (TCORB_3)
- Timer control register_3 (TCR_3)
- Timer control/status register_3 (TCSR_3)

### 11.3.1 Timer Counters (TCNT)

Each TCNT is an 8-bit up-counter. TCNT_0 and TCNT_1, or TCNT_2 and TCNT_3 comprise a single 16-bit register, so they can be accessed together by word access.

This clock source is selected by clock select bits CKS2 to CKS0 in TCR. TCNT can be cleared by an external reset input signal or compare-match signals A and B. Counter clear bits CCLR1 and CCLR0 in TCR select the method of clearing.

When TCNT overflows from H′FF to H′00, the overflow flag (OVF) in TCSR is set to 1.

The initial value of TCNT is H′00.

### 11.3.2 Time Constant Registers A (TCORA)

TCORA is an 8-bit readable/writable register. TCORA_0, TCORA_1, TCORA_2 and TCORA_3 comprise a single 16-bit register, so they can be accessed together by word access.

TCORA is continually compared with the value in TCNT. When a match is detected, the corresponding compare-match flag A (CMFA) in TCSR is set. Note, however, that comparison is disabled during the T2 state of a TCORA write cycle.

The timer output from the TMO pin can be freely controlled by the compare-match signal A and the settings of output select bits OS1 and OS0 in TCSR.

The initial value of TCORA is H′FF.

### 11.3.3 Time Constant Registers B (TCORB)

TCORB is an 8-bit readable/writable register. TCORB_0, TCORB_1, TCORB_2 and TCORB_3 comprise a single 16-bit register, so they can be accessed together by word access.

TCORB is continually compared with the value in TCNT. When a match is detected, the corresponding compare-match flag B (CMFB) in TCSR is set. Note, however, that comparison is disabled during the T2 state of a TCORB write cycle.

The timer output from the TMO pin can be freely controlled by the compare-match signal B and the settings of output select bits OS1 and OS0 in TCSR.

The initial value of TCORB is H′FF.

### 11.3.4 Timer Control Registers (TCR)

TCR selects the TCNT clock source and the time at which TCNT is cleared, and controls interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMIEB | 0 | R/W | Compare-Match Interrupt Enable B<br>Selects whether the CMFB interrupt request (CMIB) is enabled or disabled when the CMFB flag in TCSR is set to 1.<br>0: CMFB interrupt request (CMIB) is disabled<br>1: CMFB interrupt request (CMIB) is enabled |
| 6 | CMIEA | 0 | R/W | Compare-Match Interrupt Enable A<br>Selects whether the CMFA interrupt request (CMIA) is enabled or disabled when the CMFA flag in TCSR is set to 1.<br>0: CMFA interrupt request (CMIA) is disabled<br>1: CMFA interrupt request (CMIA) is enabled |
| 5 | OVIE | 0 | R/W | Timer Overflow Interrupt Enable<br>Selects whether the OVF interrupt request (OVI) is enabled or disabled when the OVF flag in TCSR is set to 1.<br>0: OVF interrupt request (OVI) is disabled<br>1: OVF interrupt request (OVI) is enabled |
| 4 | CCLR1 | 0 | R/W | Counter Clear 1 and 0 |
| 3 | CCLR0 | 0 | R/W | These bits select the method by which TCNT is cleared<br>00: Clearing is disabled<br>01: Cleared on compare-match A<br>10: Cleared on compare-match B<br>11: Cleared on rising edge of external reset input |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | CKS2 | 0 | R/W | Clock Select 2 to 0 |
| 1 | CKS1 | 0 | R/W | The input clock can be selected from three clocks divided from the system clock (φ). When use of an external clock is selected, three types of count can be selected: at the rising edge, the falling edge, and both rising and falling edges. |
| 0 | CKS0 | 0 | R/W | |
| | | | | 000: Clock input disabled |
| | | | | 001: φ/8 internal clock source, counted on the falling edge |
| | | | | 010: φ/64 internal clock source, counted on the falling edge |
| | | | | 011: φ/8192 internal clock source, counted on the falling edge |
| | | | | 100: For channel 0: Counted on TCNT1 overflow signal* |
| | | | | For channel 1: Counted on TCNT0 overflow signal* |
| | | | | For channel 2: Counted on TCNT3 overflow signal* |
| | | | | For channel 3: Counted on TCNT2 overflow signal* |
| | | | | 101: External clock source, counted at rising edge |
| | | | | 110: External clock source, counted at falling edge |
| | | | | 111: External clock source, counted at both rising and falling edges |

Note:* If the count input of channel 0 (channel 2) is the TCNT1 (TCNT3) overflow signal and that of channel 1 (channel 3) is the TCNT1 (TCNT3) compare-match signal, no incrementing clock will be generated. Do not use this setting.

### 11.3.5 Timer Control/Status Registers (TCSR)

TCSR indicates status flags and controls compare-match output.

- TCSR_0

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMFB | 0 | R/(W)* | Compare-Match Flag B<br>[Setting condition]<br>When TCNT = TCORB<br>[Clearing condition]<br>• Read CMFB when CMFB = 1, then write 0 in CMFB.<br>• DTC is activated by the CMIB interrupt and the DISEL bit = 0 in MRB of TDC. |
| 6 | CMFA | 0 | R/(W)* | Compare-match Flag A<br>[Setting condition]<br>When TCNT = TCORA<br>[Clearing condition]<br>• Read CMFA when CMFA = 1, then write 0 in CMFA.<br>• DTC is activated by the CMIA interrupt and DISEL bit = 0 in MRB of DTC. |
| 5 | OVF | 0 | R/(W)* | Timer Overflow Flag<br>[Setting condition]<br>When TCNT overflows from H′FF to H′00<br>[Clearing condition]<br>Read OVF when OVF = 1, then write 0 in OVF |
| 4 | ADTE | 0 | R/W | A/D Trigger Enable<br>Enables or disables A/D converter start requests by compare-match A.<br>0: A/D converter start requests by compare-match A are disabled<br>1: A/D converter start requests by compare-match A are enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | OS3 | 0 | R/W | Output Select 3 and 2 |
| 2 | OS2 | 0 | R/W | These bits specify how the timer output level is to be changed by a compare-match B of TCORB and TCNT.<br>00: No change when compare-match B occurs<br>01: 0 is output when compare-match B occurs<br>10: 1 is output when compare-match B occurs<br>11: Output is inverted when compare-match B occurs (toggle output) |
| 1 | OS1 | 0 | R/W | Output Select 1 and 0 |
| 0 | OS0 | 0 | R/W | These bits specify how the timer output level is to be changed by a compare-match A of TCORA and TCNT.<br>00: No change when compare-match A occurs<br>01: 0 is output when compare-match A occurs<br>10: 1 is output when compare-match A occurs<br>11: Output is inverted when compare-match A occurs (toggle output) |

Note:* Only a 0 can be written to this bit, to clear the flag

- TCSR_1 and TCSR_3

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMFB | 0 | R/(W)* | Compare-Match Flag B<br>[Setting condition]<br>When TCNT = TCORB<br>[Clearing condition]<br>• Read CMFB when CMFB = 1, then write 0 in CMFB<br>• DTC is activated by the CMIB interrupt and the DISEL bit = 0 in MRB of DTC. |
| 6 | CMFA | 0 | R/(W)* | Compare-match Flag A<br>[Setting condition]<br>When TCNT = TCORA<br>[Clearing condition]<br>• Read CMFA when CMFA = 1, then write 0 in CMFA<br>• DTC is activated by the CMIA interrupt and the DISEL bit = 0 in MRB of DTC. |
| 5 | OVF | 0 | R/(W)* | Timer Overflow Flag<br>[Setting condition]<br>When TCNT overflows from H′FF to H′00<br>[Clearing condition]<br>Read OVF when OVF = 1, then write 0 in OVF |
| 4 | — | 1 | — | Reserved<br>This bit is always read as 1 and cannot be modified. |
| 3 | OS3 | 0 | R/W | Output Select 3 and 2<br>These bits specify how the timer output level is to be changed by a compare-match B of TCORB and TCNT.<br>00: No change when compare-match B occurs<br>01: 0 is output when compare-match B occurs<br>10: 1 is output when compare-match B occurs<br>11: Output is inverted when compare-match B occurs (toggle output) |
| 2 | OS2 | 0 | R/W | |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | OS1 | 0 | R/W | Output Select 1 and 0 |
| 0 | OS0 | 0 | R/W | These bits specify how the timer output level is to be changed by a compare-match A of TCORA and TCNT.<br>00: No change when compare-match A occurs<br>01: 0 is output when compare-match A occurs<br>10: 1 is output when compare-match A occurs<br>11: Output is inverted when compare-match A occurs (toggle output) |

Note:* Only a 0 can be written to this bit, to clear the flag.

- TCSR_2

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CMFB | 0 | R/(W)* | Compare-Match Flag B<br>[Setting condition]<br>When TCNT = TCORB<br>[Clearing condition]<br>• Read CMFB when CMFB = 1, then write 0 in CMFB<br>• DTC is activated by the CMIB interrupt and the DISEL bit = 0 in MRB of DTC. |
| 6 | CMFA | 0 | R/(W)* | Compare-match Flag A<br>[Setting condition]<br>When TCNT = TCORA<br>[Clearing condition]<br>• Read CMFA when CMFA = 1, then write 0 in CMFA<br>• DTC is activated by the CMIA interrupt and the DISEL bit = 0 in MRB of DTC. |
| 5 | OVF | 0 | R/(W)* | Timer Overflow Flag<br>[Setting condition]<br>When TCNT overflows from H′FF to H′00<br>[Clearing condition]<br>Read OVF when OVF = 1, then write 0 in OVF |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | — | 0 | R/W | Reserved<br>This bit is a readable/writable bit, but the write value should always be 0. |
| 3 | OS3 | 0 | R/W | Output Select 3 and 2<br>These bits specify how the timer output level is to be changed by a compare-match B of TCORB and TCNT.<br>00: No change when compare-match B occurs<br>01: 0 is output when compare-match B occurs<br>10: 1 is output when compare-match B occurs<br>11: Output is inverted when compare-match B occurs (toggle output) |
| 2 | OS2 | 0 | R/W | |
| 1 | OS1 | 0 | R/W | Output Select 1 and 0<br>These bits specify how the timer output level is to be changed by a compare-match A of TCORA and TCNT.<br>00: No change when compare-match A occurs<br>01: 0 is output when compare-match A occurs<br>10: 1 is output when compare-match A occurs<br>11: Output is inverted when compare-match A occurs (toggle output) |
| 0 | OS0 | 0 | R/W | |

Note:* Only a 0 can be written to this bit, to clear the flag.

## 11.4 Operation

### 11.4.1 Pulse Output

Figure 11.2 shows an example of arbitrary duty cycle pulse output.

1. Set TCR in CCR1 to 0 and CCLR0 to 1 to clear TCNT by a TCORA compare-match.
2. Set OS3 to OS0 bits in TCSR to B′0110 to output 1 by a compare-match A and 0 by compare-match B.

By the above settings, waveforms with the cycle of TCORA and the pulse width of TCRB can be output without software intervention.

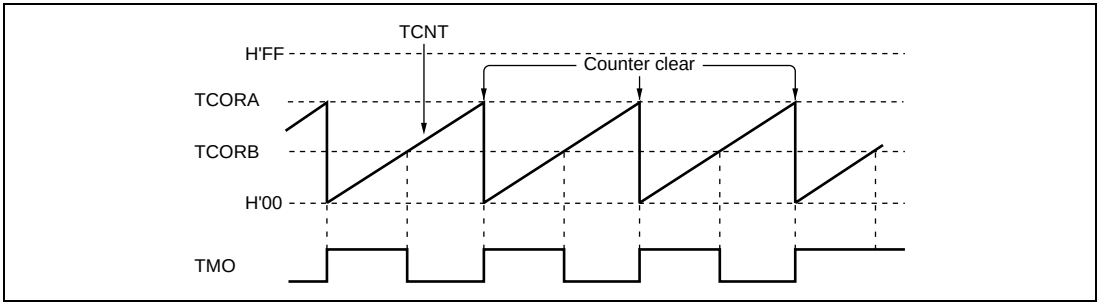



**Figure 11.2 Example of Pulse Output**

## 11.5 Operation Timing

### 11.5.1 TCNT Incrementation Timing

Figure 11.3 shows the TCNT count timing with internal clock source. Figure 11.4 shows the TCNT incrementation timing with external clock source. The pulse width of the external clock for incrementation at signal edge must be at least 1.5 system clock (ϕ) periods, and at least 2.5 states for incrementation at both edges. The counter will not increment correctly if the pulse width is less than these values.

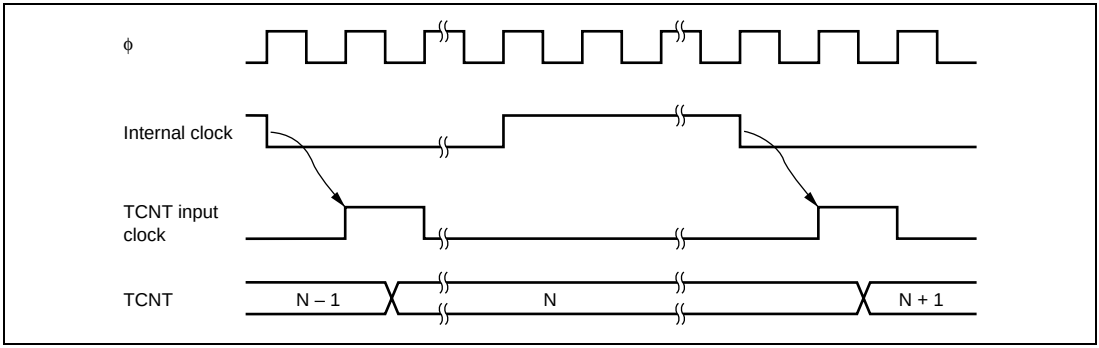



**Figure 11.3 Count Timing for Internal Clock Input**

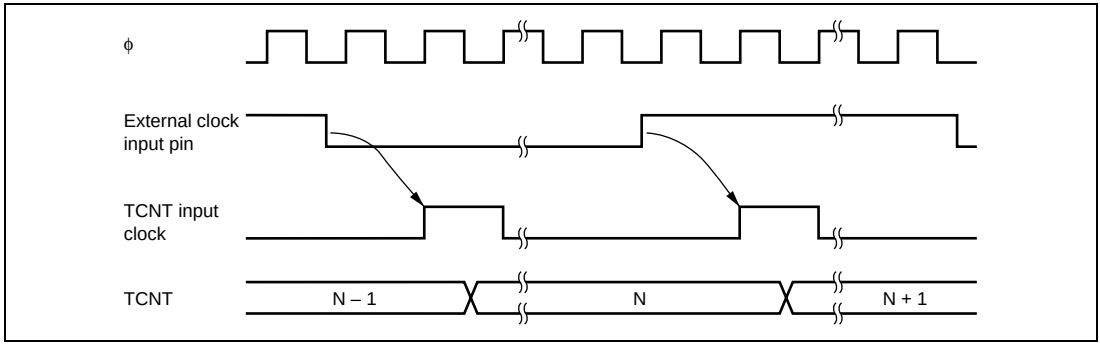



**Figure 11.4 Count Timing for External Clock Input**

### 11.5.2 Timing of CMFA and CMFB Setting When a Compare-Match Occurs

The CMFA and CMFB flags in TCSR are set to 1 by a compare-match signal generated when the TCOR and TCNT values match. The compare-match signal is generated at the last state in which the match is true, just before the timer counter is updated. Therefore, when TCOR and TCNT match, the compare-match signal is not generated until the next incrementation clock input. Figure 11.5 shows the timing of CMF flag setting.

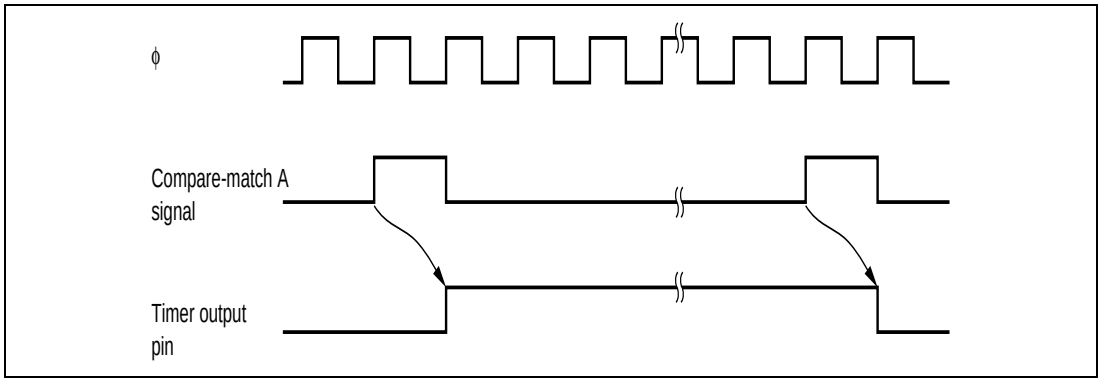



**Figure 11.5 Timing of CMF Setting**

### 11.5.3 Timing of Timer Output When a Compare-Match Occurs

When a compare-match occurs, the timer output changes as specified by the output select bits (OS3 to OS0) in TCSR. Figure 11.6 shows the timing when the output is set to toggle at compare-match A.

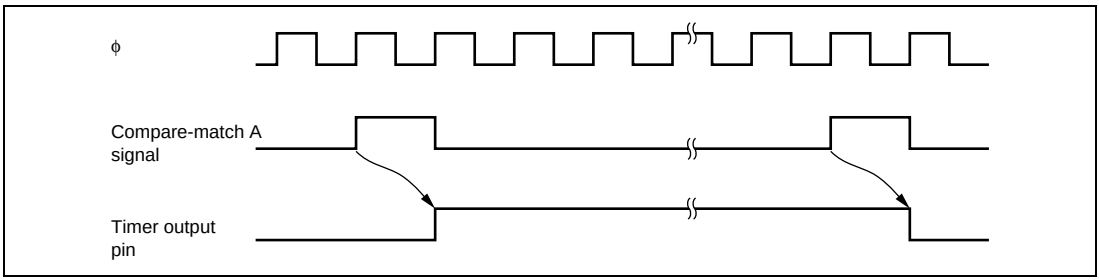



**Figure 11.6 Timing of Timer Output**

### 11.5.4 Timing of Compare-Match Clear When a Compare-Match Occurs

TCNT is cleared when compare-match A or B occurs, depending on the setting of the CCLR1 and CCLR0 bits in TCR. Figure 11.7 shows the timing of this operation.

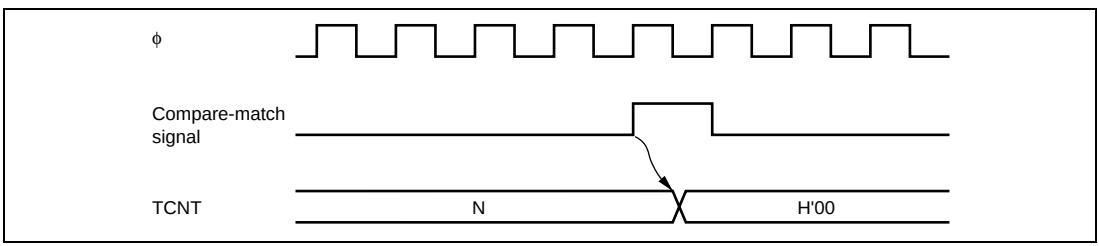



**Figure 11.7 Timing of Compare-Match Clear**

### 11.5.5 TCNT External Reset Timing

TCNT is cleared at the rising edge of an external reset input, depending on the settings of the CCLR1 and CCLR0 bits in TCR. The width of the clearing pulse must be at least 1.5 states. Figure 11.8 shows the timing of this operation.

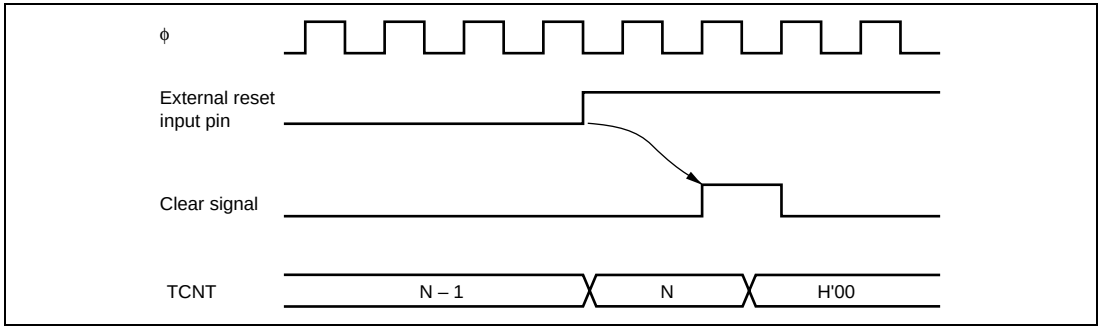



**Figure 11.8 Timing of Clearing by External Reset Input**

### 11.5.6 Timing of Overflow Flag (OVF) Setting

OVF in TCSR is set to 1 when the timer count overflows (changes from H′FF to H′00). Figure 11.9 shows the timing of this operation.

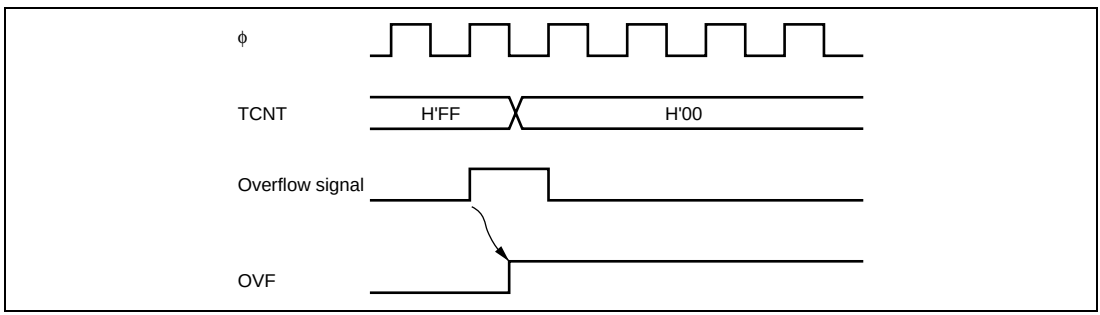



**Figure 11.9 Timing of OVF Setting**

## 11.6 Operation with Cascaded Connection

If bits CKS2 to CKS0 in one of TCR_0 and TCR_1, or TCR_2 and TCR_3 are set to B′100, the 8-bit timers of the two channels are cascaded. With this configuration, a single 16-bit timer can be used (16-bit timer mode) or compare-matches of 8-bit channel 0 (Channel 2) can be counted by the timer of channel 1 (Channel 3) (compare-match count mode). In the case that channel 0 is connected to channel 1 in cascade, the timer operates as described below.

### 11.6.1 16-Bit Count Mode

When bits CKS2 to CKS0 in TCR_0 are set to B′100, the timer functions as a single 16-bit timer with channel 0 occupying the upper 8 bits and channel 1 occupying the lower 8 bits.

- Setting of compare-match flags
  — The CMF flag in TCSR_0 is set to 1 when a 16-bit compare-match occurs.

— The CMF flag in TCSR_1 is set to 1 when a lower 8-bit compare-match occurs.

- Counter clear specification
  — If the CCLR1 and CCLR0 bits in TCR_0 have been set for counter clear at compare-match, the 16-bit counter (TCNT_0 and TCNT_1 together) is cleared when a 16-bit compare-match occurs. The 16-bit counter (TCNT_0 and TCNT_1 together) is cleared even if counter clear by the TMRI01 pin has also been set.
  — The settings of the CCLR1 and CCLR0 bits in TCR_1 are ignored. The lower 8 bits cannot be cleared independently.
- Pin output
  — Control of output from the TMO0 pin by bits OS3 to OS0 in TCSR_0 is in accordance with the 16-bit compare-match conditions.
  — Control of output from the TMO1 pin by bits OS3 to OS0 in TCSR_1 is in accordance with the lower 8-bit compare-match conditions.

### 11.6.2 Compare-Match Count Mode

When bits CKS2 to CKS0 in TCR_1 are B′100, TCNT_1 counts compare-match A for channel 0. Channels 0 and 1 are controlled independently. Conditions such as setting of the CMF flag, generation of interrupts, output from the TMO pin, and counter clearing are in accordance with the settings for each channel.

## 11.7 Interrupt Sources

### 11.7.1 Interrupt Sources and DTC Activation

The 8-bit timer can generate three types of interrupt: CMIA, CMIB, and OVI. Table 11.2 shows the interrupt sources and priority. Each interrupt source can be enabled or disabled independently by interrupt enable bits in TCR. Independent signals are sent to the interrupt controller for each interrupt. It is also possible to activate the DTC by means of CMIA and CMIB interrupts.

**Table 11.2 8-Bit Timer Interrupt Sources**

| Interrupt source | Description | Flag | DTC Activation | Interrupt Priority |
|---|---|---|---|---|
| CMIA0 | TCORA_0 compare-match | CMFA | Possible | High |
| CMIB0 | TCORA_0 compare-match | CMFB | Possible | |
| OVI0 | TCNT_0 overflow | OVF | Not possible | |
| CMIA1 | TCORA_1 compare-match | CMFA | Possible | |
| CMIB1 | TCORA_1 compare-match | CMFB | Possible | |
| OVI1 | TCNT_1 overflow | OVF | Not possible | |
| CMIA2 | TCORA_2 compare-match | CMFA | Possible | |
| CMIB2 | TCORA_2 compare-match | CMFB | Possible | |
| OVI2 | TCNT_2 overflow | OVF | Not possible | |
| CMIA3 | TCORA_3 compare-match | CMFA | Possible | |
| CMIB3 | TCORA_3 compare-match | CMFB | Possible | |
| OVI3 | TCNT_3 overflow | OVF | Not possible | Low |

### 11.7.2 A/D Converter Activation

The A/D converter can be activated only by channel 0 compare match A.

If the ADTE bit in TCSR0 is set to 1 when the CMFA flag is set to 1 by the occurrence of channel 0 compare match A, a request to start A/D conversion is sent to the A/D converter. If the 8-bit timer conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is started.

## 11.8 Usage Notes

### 11.8.1 Conflict between TCNT Write and Clear

If a timer counter clock pulse is generated during the T2 state of a TCNT write cycle, the clear takes priority, so that the counter is cleared and the write is not performed. Figure 11.10 shows this operation.

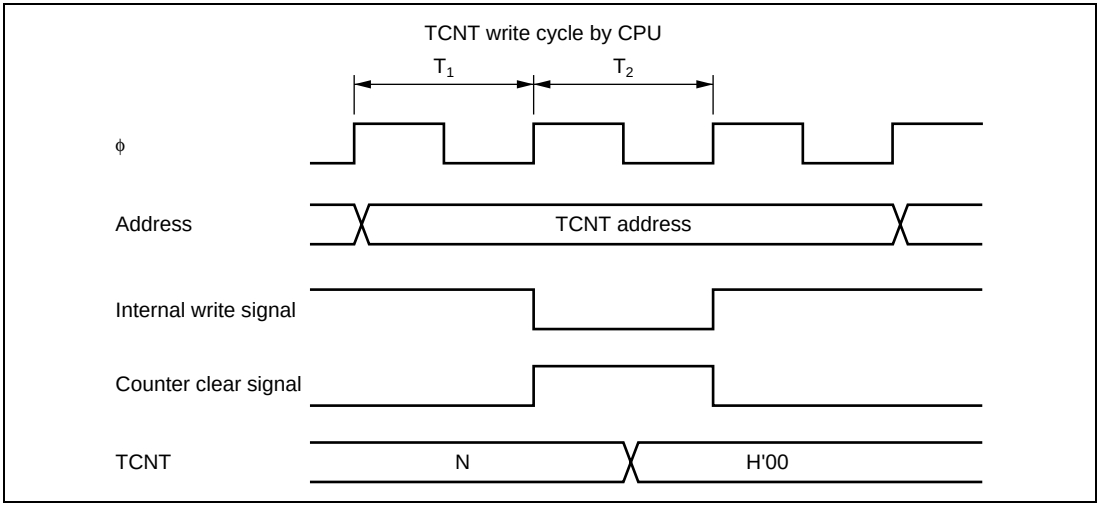



**Figure 11.10 Conflict between TCNT Write and Clear**

### 11.8.2 Conflict between TCNT Write and Increment

If a timer counter clock pulse is generated during the T2 state of a TCNT write cycle, the write takes priority and the counter is not incremented. Figure 11.11 shows this operation.

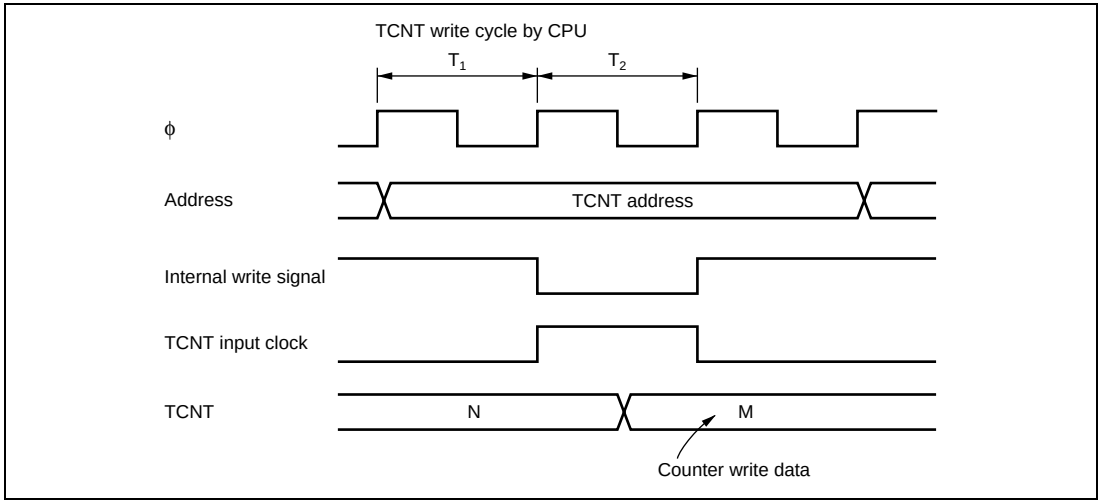



**Figure 11.11 Conflict between TCNT Write and Increment**

### 11.8.3 Conflict between TCOR Write and Compare-Match

During the T2 state of a TCOR write cycle, the TCOR write has priority even if a compare-match occurs and the compare-match signal is disabled. Figure 11.12 shows this operation.

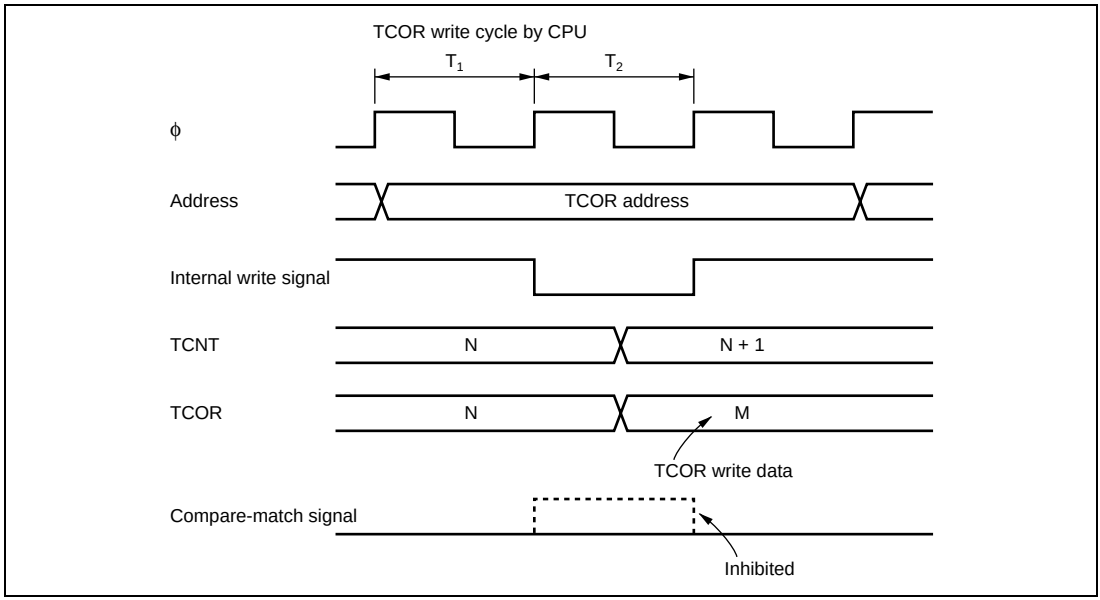



**Figure 11.12 Conflict between TCOR Write and Compare-Match**

### 11.8.4 Conflict between Compare-Matches A and B

If compare-matches A and B occur at the same time, the 8-bit timer operates in accordance with the priorities for the output states set for compare-match A and compare-match B, as shown in table 11.3.

**Table 11.3 Timer Output Priorities**

| Output Setting | Priority |
|---|---|
| Toggle output | High |
| 1 output | |
| 0 output | |
| No change | Low |

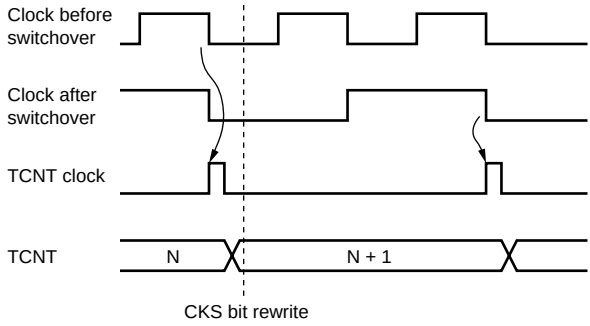
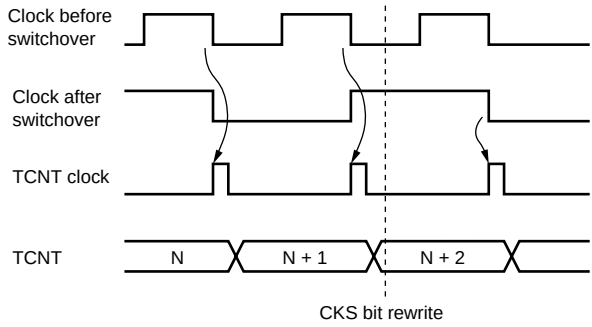
### 11.8.5 Switching of Internal Clocks and TCNT Operation

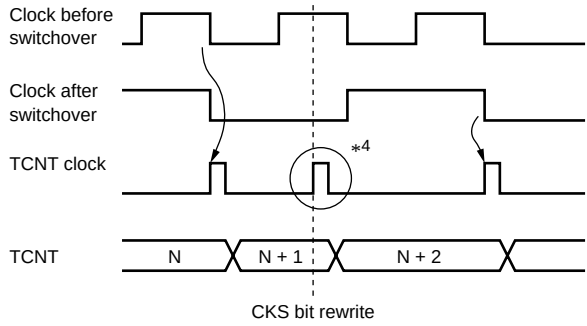
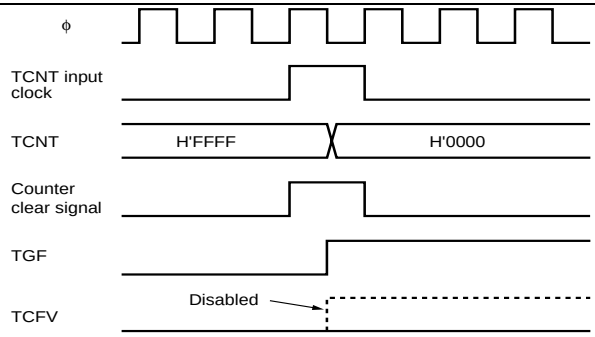
TCNT may increment erroneously when the internal clock is switched over. Table 11.4 shows the relationship between the timing at which the internal clock is switched (by writing to the CKS1 and CKS0 bits) and the TCNT operation.

When the TCNT clock is generated from an internal clock, the falling edge of the internal clock pulse is detected. If clock switching causes a change from high to low level, as shown in no. 3 in table 11.4, a TCNT clock pulse is generated on the assumption that the switchover is a falling edge. This increments TCNT.

Erroneous incrementation can also happen when switching between internal and external clocks.

**Table 11.4 Switching of Internal Clock and TCNT Operation**

| No. | Timing of Switchover by Means of CKS1 and CKS0 Bits | TCNT Clock Operation |
|---|---|---|
| 1 | Switching from low to low*[1] |  |
| 2 | Switching from low to high*[2] |  |

| No. | Timing of Switchover by Means of CKS1 and CKS0 Bits | TCNT Clock Operation |
|---|---|---|
| 3 | Switching from high to low*[3] |  |
| 4 | Switching from high to high |  |

Notes: 1. Includes switching from low to stop, and from stop to low.
2. Includes switching from stop to high.
3. Includes switching from high to stop.
4. Generated on the assumption that the switchover is a falling edge; TCNT is incremented.

### 11.8.6 Conflict between Interrupts and Module Stop Mode

If module stop mode is entered when an interrupt has been requested, it will not be possible to clear the CPU interrupt source or the DTC activation source. Interrupts should therefore be disabled before entering module stop mode.

### 11.8.7 Notes on Cascaded Connection

If 16-bit count mode and compare-match count mode are set simultaneously, the counter stops and does not operate since input clocks of TCNT_0 and TCNT_1 (TCNT_2 and TCNT_3) are not generated. This setting is inhibited.

# Section 12 Programmable Pulse Generator (PPG)

The programmable pulse generator provides pulse outputs using the 16-bit timer pulse unit (TPU) as a time base. The PPG pulse outputs are divided into 4-bit groups (group 2 and group 3) that can operate both simultaneously and independently. The block diagram of the PPG is shown in figure 12.1.

## 12.1 Features

- 8-bit output data
- Two output groups
- Selectable output trigger signals
- Non-overlap mode
- Can operate in tandem with the data transfer controller (DTC)
- Settable inverted output
- Module stop mode can be set

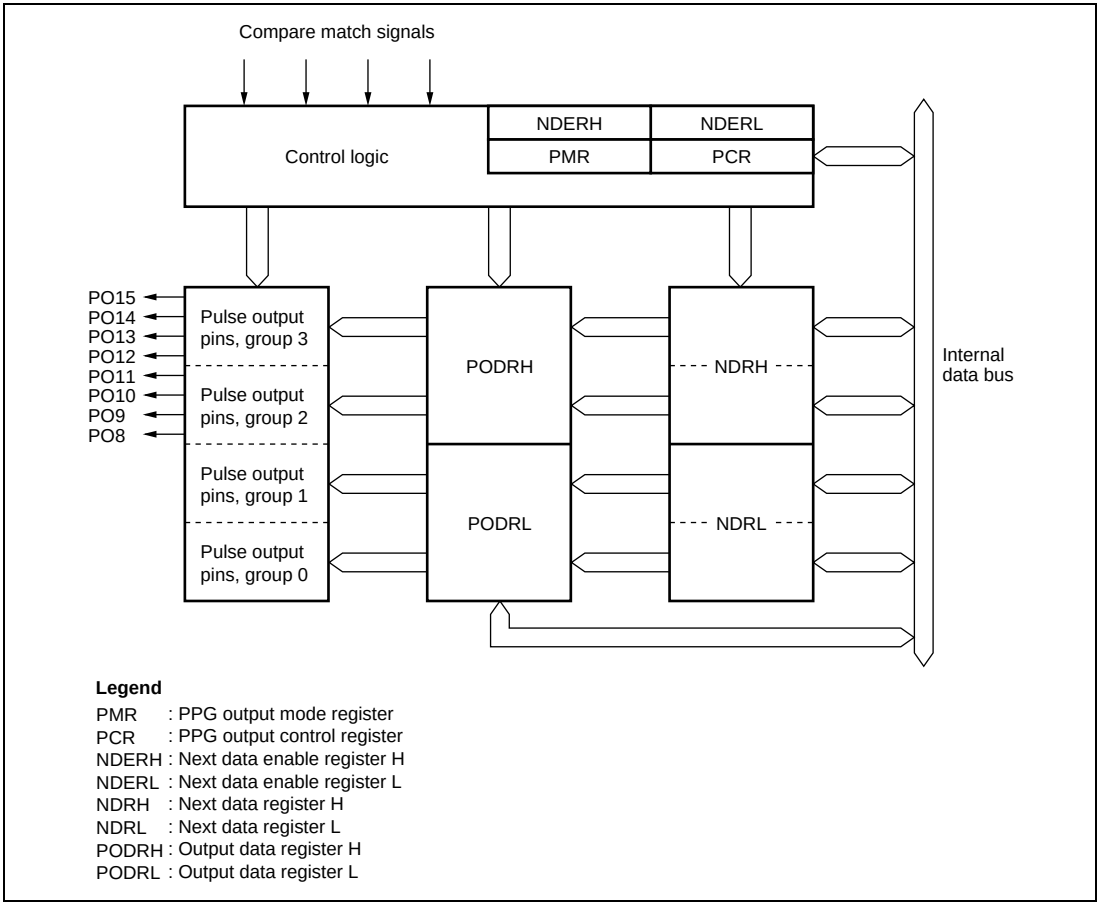



**Figure 12.1 Block Diagram of PPG**

## 12.2 Input/Output Pins

Table 12.1 summarizes the pin configuration of the PPG.

**Table 12.1 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| PO15 | Output | Group 3 pulse output |
| PO14 | Output | |
| PO13 | Output | |
| PO12 | Output | |
| PO11 | Output | Group 2 pulse output |
| PO10 | Output | |
| PO9 | Output | |
| PO8 | Output | |

## 12.3 Register Descriptions

The PPG has the following registers.

- PPG output control register (PCR)
- PPG output mode register (PMR)
- Next data enable register H (NDERH)
- Next data enable register L (NDERL)
- Output data register H (PODRH)
- Output data register L (PODRL)
- Next data register H (NDRH)
- Next data register L (NDRL)

### 12.3.1 Next Data Enable Registers H, L (NDERH, NDERL)

NDERH and NDERL are 8-bit readable/writable registers that enable or disable pulse output on a bit-by-bit basis. The corresponding DDR also needs to be set to 1 in order to enable pulse output by the PPG.

- NDERH

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDER15 | 0 | R/W | Next Data Enable 8 to 15 |
| 6 | NDER14 | 0 | R/W | When a bit is set to 1 for pulse output by NDRH, the value in the corresponding NDRH bit is transferred to the PODRH bit by the selected output trigger. Values are not transferred from NDRH to PODRH for cleared bits. |
| 5 | NDER13 | 0 | R/W | |
| 4 | NDER12 | 0 | R/W | |
| 3 | NDER11 | 0 | R/W | |
| 2 | NDER10 | 0 | R/W | |
| 1 | NDER9 | 0 | R/W | |
| 0 | NDER8 | 0 | R/W | |

- NDERL

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDER7 | 0 | R/W | Next Data Enable 0 to 7 |
| 6 | NDER6 | 0 | R/W | When a bit is set to 1 for pulse output by NDRL, the value in the corresponding NDRL bit is transferred to the PODRL bit by the selected output trigger. Values are not transferred from NDRL to PODRL for cleared bits. |
| 5 | NDER5 | 0 | R/W | |
| 4 | NDER4 | 0 | R/W | |
| 3 | NDER3 | 0 | R/W | |
| 2 | NDER2 | 0 | R/W | |
| 1 | NDER1 | 0 | R/W | |
| 0 | NDER0 | 0 | R/W | |

### 12.3.2 Output Data Registers H, L (PODRH, PODRL)

PODRH and PODRL are 8-bit readable/writable registers that store output data for use in pulse output. A bit that has been set for pulse output by NDER is read-only and cannot be modified.

- PODRH

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | POD15 | 0 | R/W | Output Data Register 8 to 15 |
| 6 | POD14 | 0 | R/W | For bits that have been set to pulse output by NDERH, the output trigger transfers NDRH values to this register during PPG operation. While NDERH is set to 1, the CPU cannot write to this register. While NDERH is cleared, the initial output value of the pulse can be set. |
| 5 | POD13 | 0 | R/W | |
| 4 | POD12 | 0 | R/W | |
| 3 | POD11 | 0 | R/W | |
| 2 | POD10 | 0 | R/W | |
| 1 | POD9 | 0 | R/W | |
| 0 | POD8 | 0 | R/W | |

- PODRL

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | POD7 | 0 | R/W | Output Data Register 0 to 7 |
| 6 | POD6 | 0 | R/W | For bits which have been set to pulse output by NDERL, the output trigger transfers NDRL values to this register during PPG operation. While NDERL is set to 1, the CPU cannot write to this register. While NDERL is cleared, the initial output value of the pulse can be set. |
| 5 | POD5 | 0 | R/W | |
| 4 | POD4 | 0 | R/W | |
| 3 | POD3 | 0 | R/W | |
| 2 | POD2 | 0 | R/W | |
| 1 | POD1 | 0 | R/W | |
| 0 | POD0 | 0 | R/W | |

### 12.3.3 Next Data Registers H, L (NDRH, NDRL)

NDRH and NDRL are 8-bit readable/writable registers that store the data for the next pulse output. The NDR addresses differ depending on whether pulse output groups have the same output trigger or different output triggers.

- NDRH

If pulse output groups 2 and 3 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time, as shown below.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDR15 | 0 | R/W | Next Data Register 8 to 15 |
| 6 | NDR14 | 0 | R/W | The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR. |
| 5 | NDR13 | 0 | R/W | |
| 4 | NDR12 | 0 | R/W | |
| 3 | NDR11 | 0 | R/W | |
| 2 | NDR10 | 0 | R/W | |
| 1 | NDR9 | 0 | R/W | |
| 0 | NDR8 | 0 | R/W | |

If pulse output groups 2 and output pulse groups 3 have different output triggers, the upper 4 bits and the lower 4 bits are mapped to different addresses, as shown below.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDR15 | 0 | R/W | Next Data Register 12 to 15 |
| 6 | NDR14 | 0 | R/W | The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR. |
| 5 | NDR13 | 0 | R/W | |
| 4 | NDR12 | 0 | R/W | |
| 3 to 0 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |
| 3 | NDR11 | 0 | R/W | Next Data Register 8 to11<br>The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR. |
| 2 | NDR10 | 0 | R/W | |
| 1 | NDR9 | 0 | R/W | |
| 0 | NDR8 | 0 | R/W | |

- NDRL

If pulse output groups 0 and 1 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time, as shown below.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDR7 | 0 | R/W | Next Data Register 0 to 7<br>The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR. |
| 6 | NDR6 | 0 | R/W | |
| 5 | NDR5 | 0 | R/W | |
| 4 | NDR4 | 0 | R/W | |
| 3 | NDR3 | 0 | R/W | |
| 2 | NDR2 | 0 | R/W | |
| 1 | NDR1 | 0 | R/W | |
| 0 | NDR0 | 0 | R/W | |

If pulse output groups 0 and output pulse groups 1 have different output triggers, upper 4 bits and lower 4 bits are mapped to the different addresses as shown below.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | NDR7 | 0 | R/W | Next Data Register 4 to 7<br>The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR. |
| 6 | NDR6 | 0 | R/W | |
| 5 | NDR5 | 0 | R/W | |
| 4 | NDR4 | 0 | R/W | |
| 3 to 0 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |
| 3 | NDR3 | 0 | R/W | Next Data Register 3 to 0<br>The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR. |
| 2 | NDR2 | 0 | R/W | |
| 1 | NDR1 | 0 | R/W | |
| 0 | NDR0 | 0 | R/W | |

### 12.3.4 PPG Output Control Register (PCR)

PCR is an 8-bit readable/writable register that selects output trigger signals on a group-by-group basis. For details on output trigger selection, refer to section 12.3.5, PPG Output Mode Register (PMR).

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | G3CMS1 | 1 | R/W | Group 3 Compare Match Select 0 and 1 |
| 6 | G3CMS0 | 1 | R/W | Select output trigger of pulse output group 3.<br>00: Compare match in TPU channel 0<br>01: Compare match in TPU channel 1<br>10: Compare match in TPU channel 2<br>11: Compare match in TPU channel 3 |
| 5 | G2CMS1 | 1 | R/W | Group 2 Compare Match Select 0 and 1 |
| 4 | G2CMS0 | 1 | R/W | Select output trigger of pulse output group 2.<br>00: Compare match in TPC channel 0<br>01: Compare match in TPC channel 1<br>10: Compare match in TPC channel 2<br>11: Compare match in TPC channel 3 |
| 3 | G1CMS1 | 1 | R/W | Reserved |
| 2 | G1CMS0 | 1 | R/W | |
| 1 | G0CMS1 | 1 | R/W | Reserved |
| 0 | G0CMS0 | 1 | R/W | |

### 12.3.5 PPG Output Mode Register (PMR)

The PMR is an 8-bit readable/writable register that selects the pulse output mode of the PPG for each group. If inverted output is selected, a low-level pulse is output when PODRH is 1 and a high-level pulse is output when PODRH is 0. If non-overlapping operation is selected, PPG updates its output values on compare match A or B of the TPU that becomes the output trigger. For details, refer to section 12.4.5, Non-Overlapping Pulse Output.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | G3INV | 1 | R/W | Group 3 Inversion<br>Selects direct output or inverted output for pulse output group 3.<br>0: Inverted output<br>1: Direct output |
| 6 | G2INV | 1 | R/W | Group 2 Inversion<br>Selects direct output or inverted output for pulse output group 2.<br>0: Inverted output<br>1: Direct output |
| 5, 4 | — | All 1 | R/W | Reserved |
| 3 | G3NOV | 0 | R/W | Group 3 Non-Overlap<br>Selects normal or non-overlapping operation for pulse output group 3.<br>0: Normal operation (output values updated at compare match A in the selected TPU channel)<br>1: Non-overlapping operation (output values at compare match A or B in the selected TPU channel) |
| 2 | G2NOV | 0 | R/W | Group 2 Non-Overlap<br>Selects normal or non-overlapping operation for pulse output group 2.<br>0: Normal operation (output values updated at compare match A in the selected TPU channel)<br>1: Non-overlapping operation (output values at compare match A or B in the selected TPU channel) |
| 1, 0 | — | All 0 | R/W | Reserved |

## 12.4 Operation

### 12.4.1 Overview

Figure 12.2 shows a block diagram of the PPG. PPG pulse output is enabled when the corresponding bits in P1DDR and NDER are set to 1. An initial output value is determined by its corresponding PODR initial setting. When the compare match event specified by PCR occurs, the corresponding NDR bit contents are transferred to PODR to update the output values.

The sequential output of up to 8 bits of data is possible by writing new output data to NDR before the next compare match.

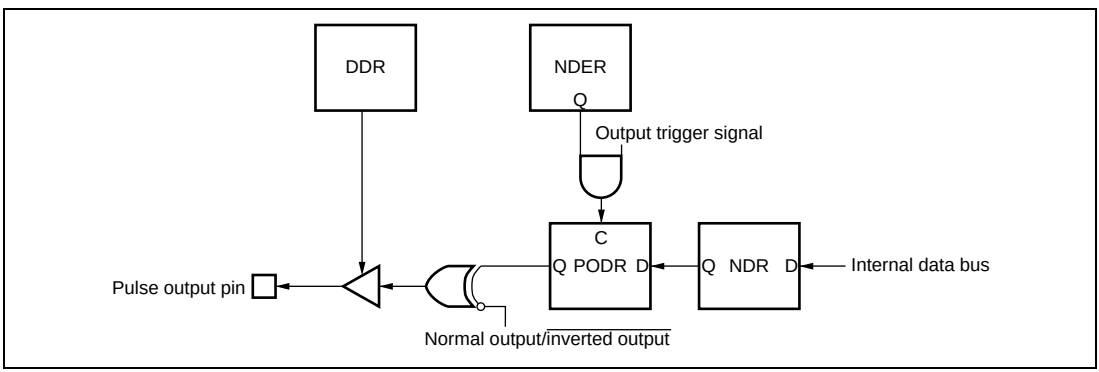



**Figure 12.2 PPG Output Operation**

### 12.4.2 Output Timing

If pulse output is enabled, the contents of NDR contents are transferred to PODR and output when the specified compare match event occurs. Figure 12.3 shows the timing of these operations for the case of normal output in groups 2 and 3, triggered by compare match A.

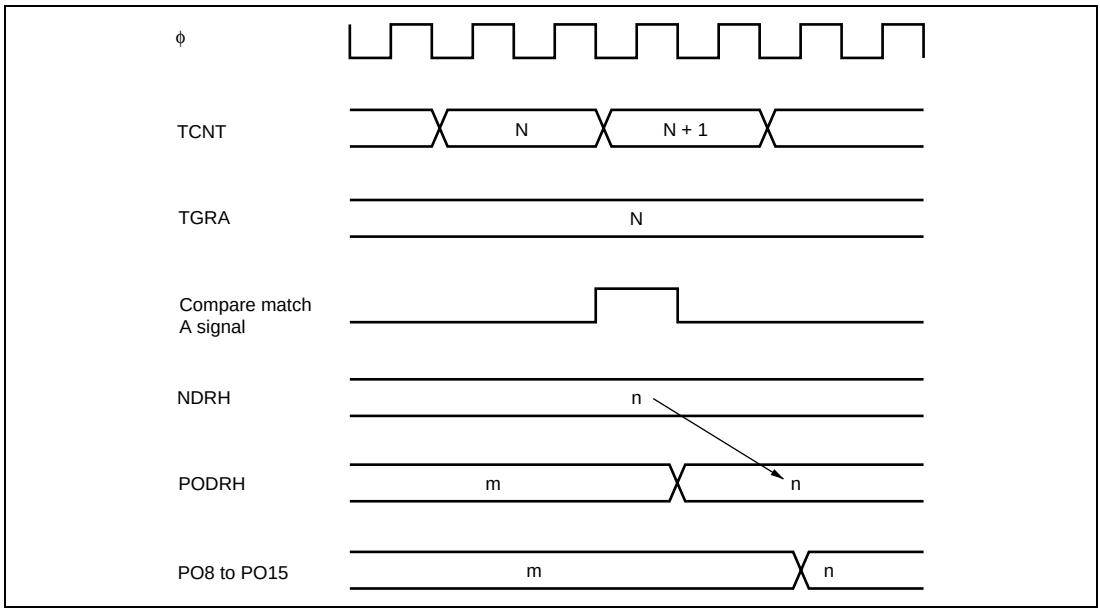



**Figure 12.3 Timing of Transfer and Output of NDR Contents (Example)**

### 12.4.3 Sample Setup Procedure for Normal Pulse Output

Figure 12.4 shows a sample procedure for setting up normal pulse output.

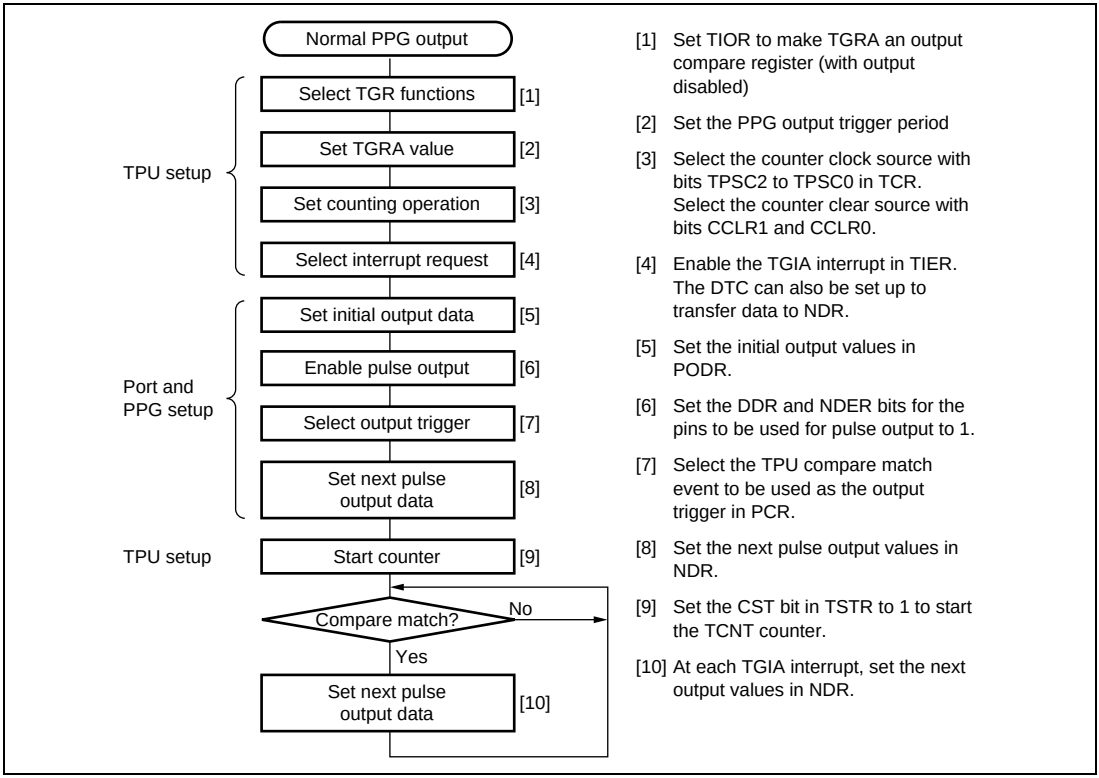



**Figure 12.4 Setup Procedure for Normal Pulse Output (Example)**

### 12.4.4 Example of Normal Pulse Output (Example of Five-Phase Pulse Output)

Figure 12.5 shows an example in which pulse output is used for cyclic five-phase pulse output.

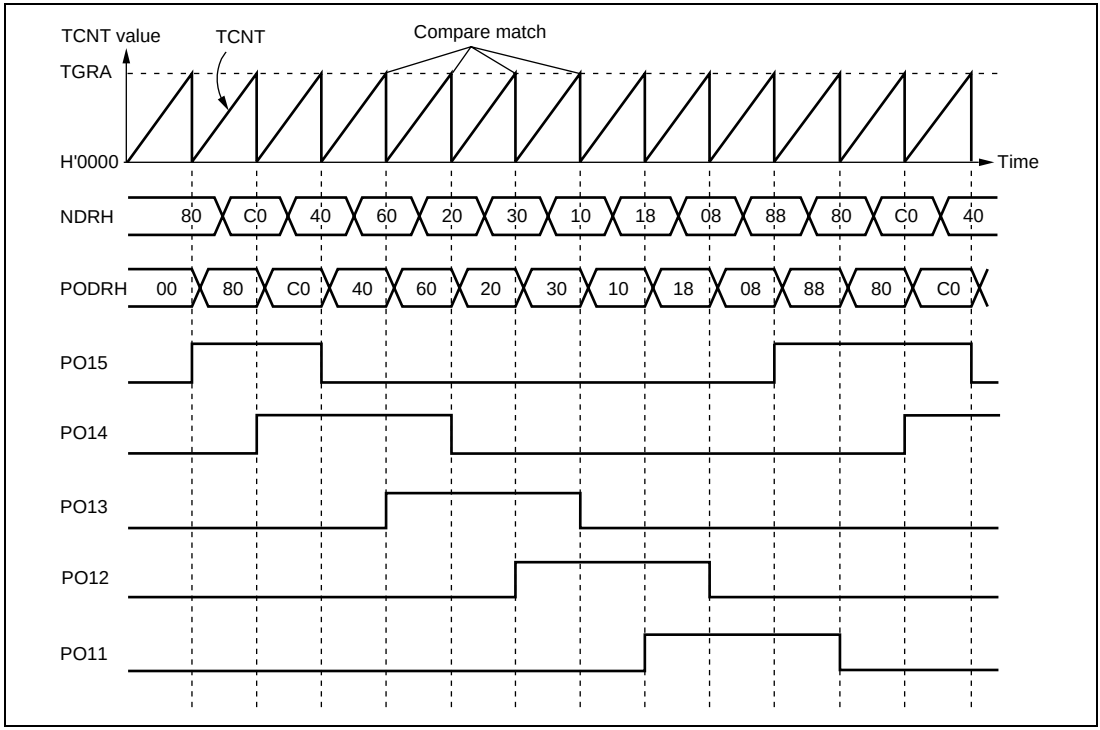



**Figure 12.5 Normal Pulse Output Example (Five-Phase Pulse Output)**

1. Set up TGRA of the TPU that is used as the output trigger to be an output compare register. Set a frequency in TGRA so the counter will be cleared on compare match A. Set the TGIEA bit of TIER to 1 to enable the compare match/input capture A (TGIA) interrupt.
2. Write H′F8 in P1DDR and NDERH, and set the G3CMS0, G3CMS1, G2CMS0, and G2CMS1 bits in PCR to select compare match in the TPU channel set up in the previous step to be the output trigger. Write output data H′80 in NDRH.
3. When compare match A occurs, the NDRH contents are transferred to PODRH and output. The TGIA interrupt handling routine writes the next output data (H′C0) in NDRH.
4. Five-phase overlapping pulse output (one or two phases active at a time) can be obtained subsequently by writing H′40, H′60, H′20, H′30. H′10, H′18, H′08, H′88... at successive TGIA interrupts. If the DTC is set for activation by this interrupt, pulse output can be obtained without imposing a load on the CPU.

### 12.4.5 Non-Overlapping Pulse Output

During non-overlapping operation, transfer from NDR to PODR is performed as follows:

- NDR bits are always transferred on PODR bits on compare match A.
- On compare match B, NDR bits are transferred only if their value is 0. Bits are not transferred if their value is 1.

Figure 12.6 illustrates the non-overlapping pulse output operation.

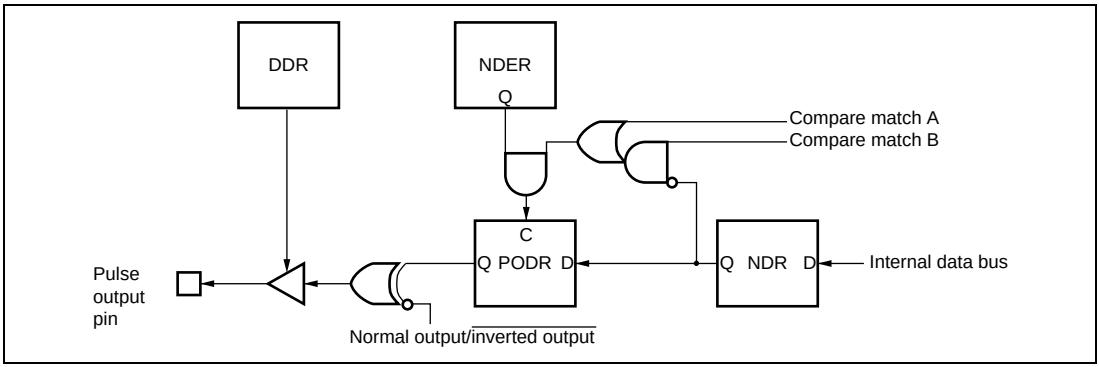



**Figure 12.6 Non-Overlapping Pulse Output**

Therefore, 0 data can be transferred ahead of 1 data by making compare match B occur before compare match A. The NDR contents should not be altered during the interval between compare match B and compare match A (the non-overlap margin).

This can be accomplished by having the TGIA interrupt handling routine write the next data in NDR, or by having the TGIA interrupt activate the DTC. Note, however, that the next data must be written before the next compare match B occurs.

Figure 12.7 shows the timing of this operation.

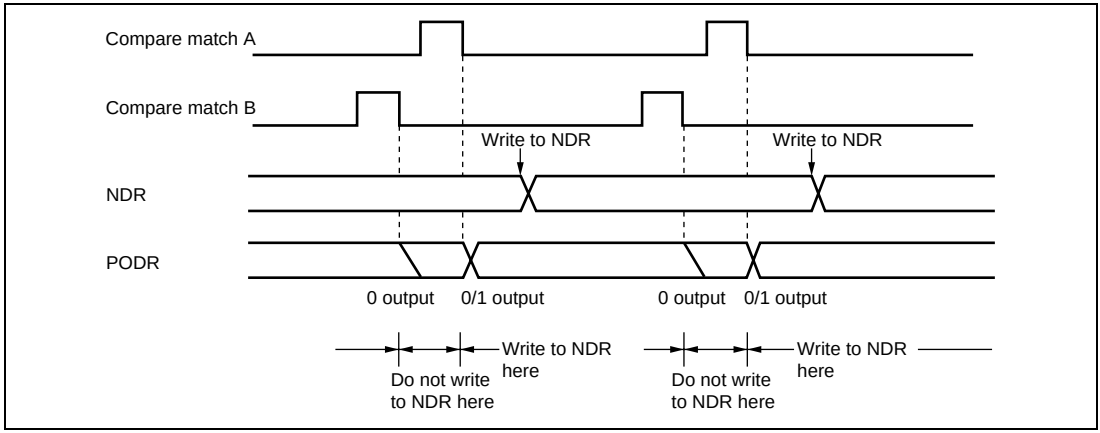



**Figure 12.7 Non-Overlapping Operation and NDR Write Timing**

### 12.4.6 Sample Setup Procedure for Non-Overlapping Pulse Output

Figure 12.8 shows a sample procedure for setting up non-overlapping pulse output.

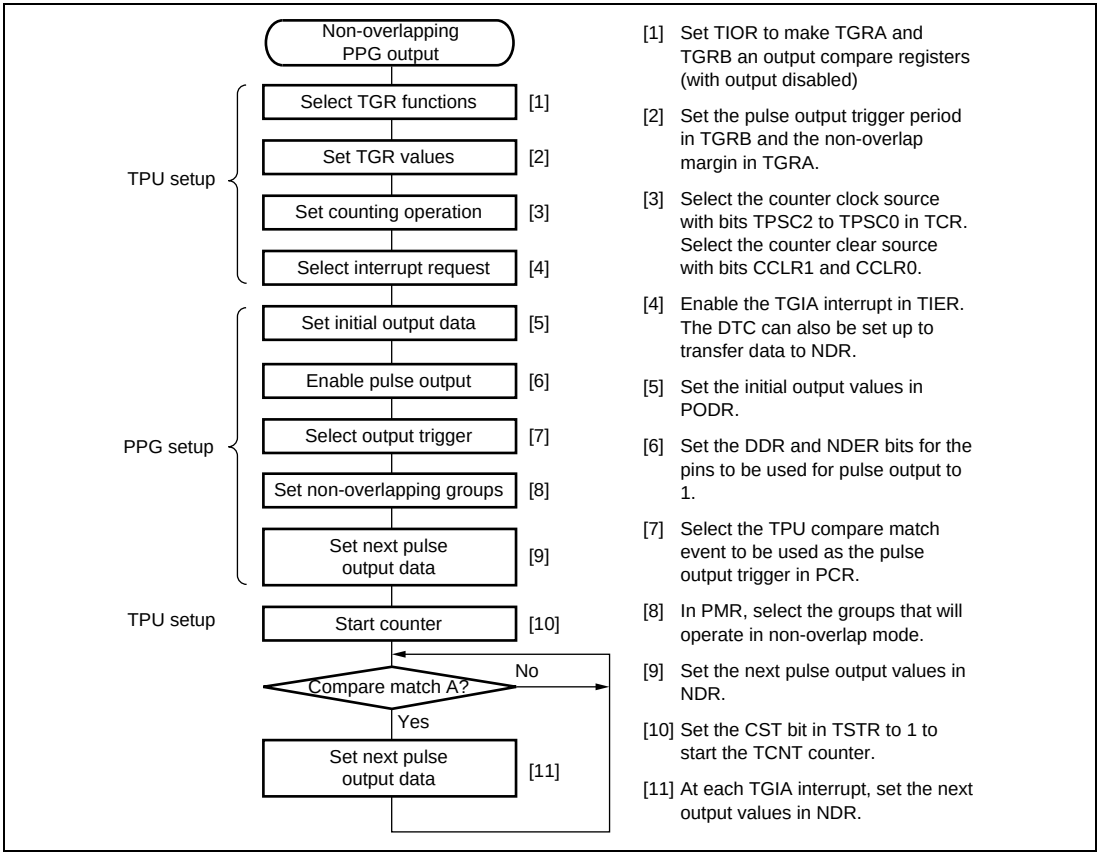



**Figure 12.8 Setup Procedure for Non-Overlapping Pulse Output (Example)**

### 12.4.7 Example of Non-Overlapping Pulse Output (Example of Four-Phase Complementary Non-Overlapping Output)

Figure 12.9 shows an example in which pulse output is used for four-phase complementary non-overlapping pulse output.

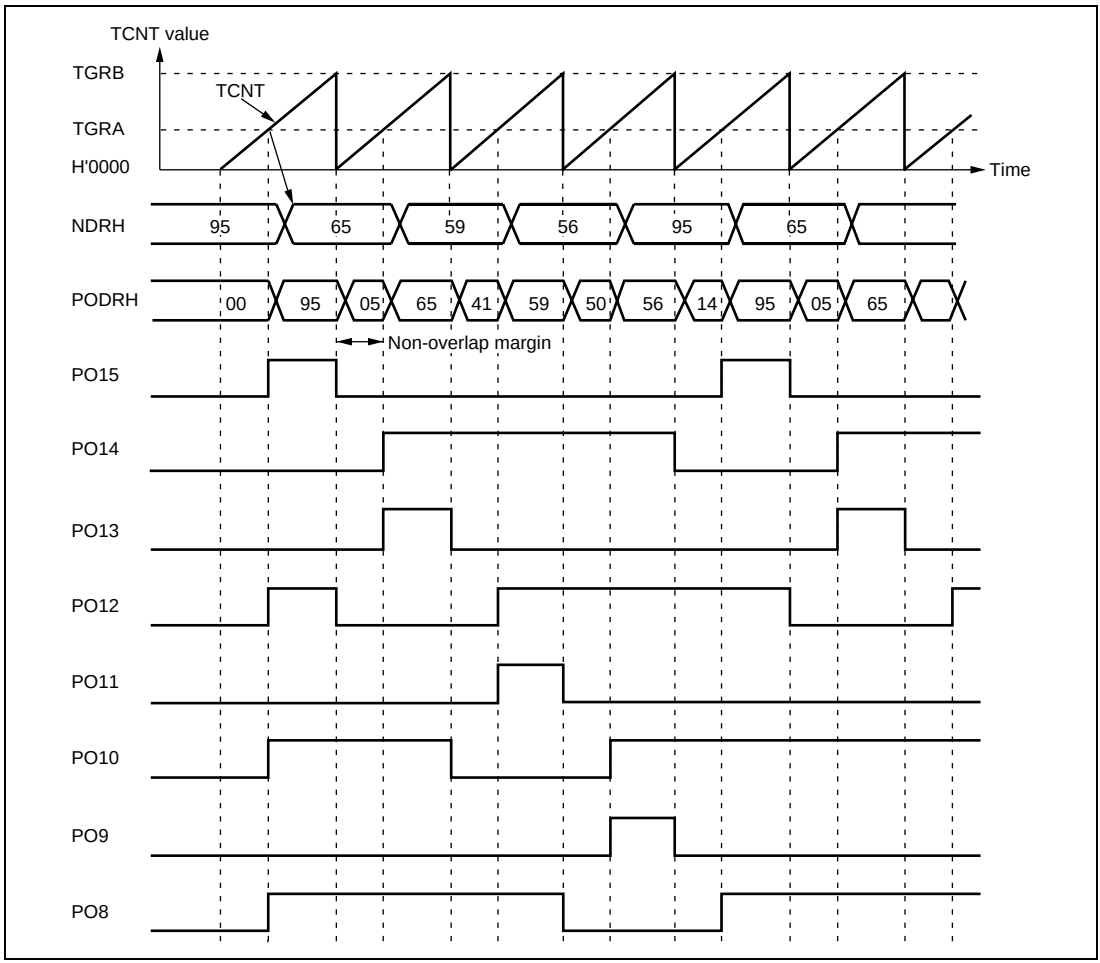



**Figure 12.9 Non-Overlapping Pulse Output Example (Four-Phase Complementary)**

1. Set up the TPU channel to be used as the output trigger channel such that TGRA and TGRB are output compare registers. Set the trigger period in TGRB and the non-overlap margin in TGRA, and set the counter to be cleared on compare match B. Set the TGIEA bit in TIER to 1 to enable the TGIA interrupt.
2. Write H′FF in P1DDR and NDERH, and set the G3CMS1, G3CMS0, G2CMS1, and G2CMS0 bits in PCR to select compare match in the TPU channel set up in the previous step to be the output trigger. Set the G3NOV and G2NOV bits in PMR to 1 to select non-overlapping output. Write output data H′95 in NDRH.
3. The timer counter in the TPU channel starts. When a compare match with TGRB occurs, outputs change from 1 to 0. When a compare match with TGRA occurs, outputs change from 0 to 1 (the change from 0 to 1 is delayed by the value set in TGRA). The TGIA interrupt handling routine writes the next output data (H′65) in NDRH.
4. Four-phase complementary non-overlapping pulse output can be obtained subsequently by writing H′59, H′56, H′95... at successive TGIA interrupts. If the DTC is set for activation by this interrupt, pulse output can be obtained without imposing a load on the CPU.

### 12.4.8 Inverted Pulse Output

If the G3INV, G2INV, G1INV, and G0INV bits in PMR are cleared to 0, values that are the inverse of the PODR contents can be output.

Figure 12.10 shows the outputs when G3INV and G2INV are cleared to 0, in addition to the settings of figure 12.9.

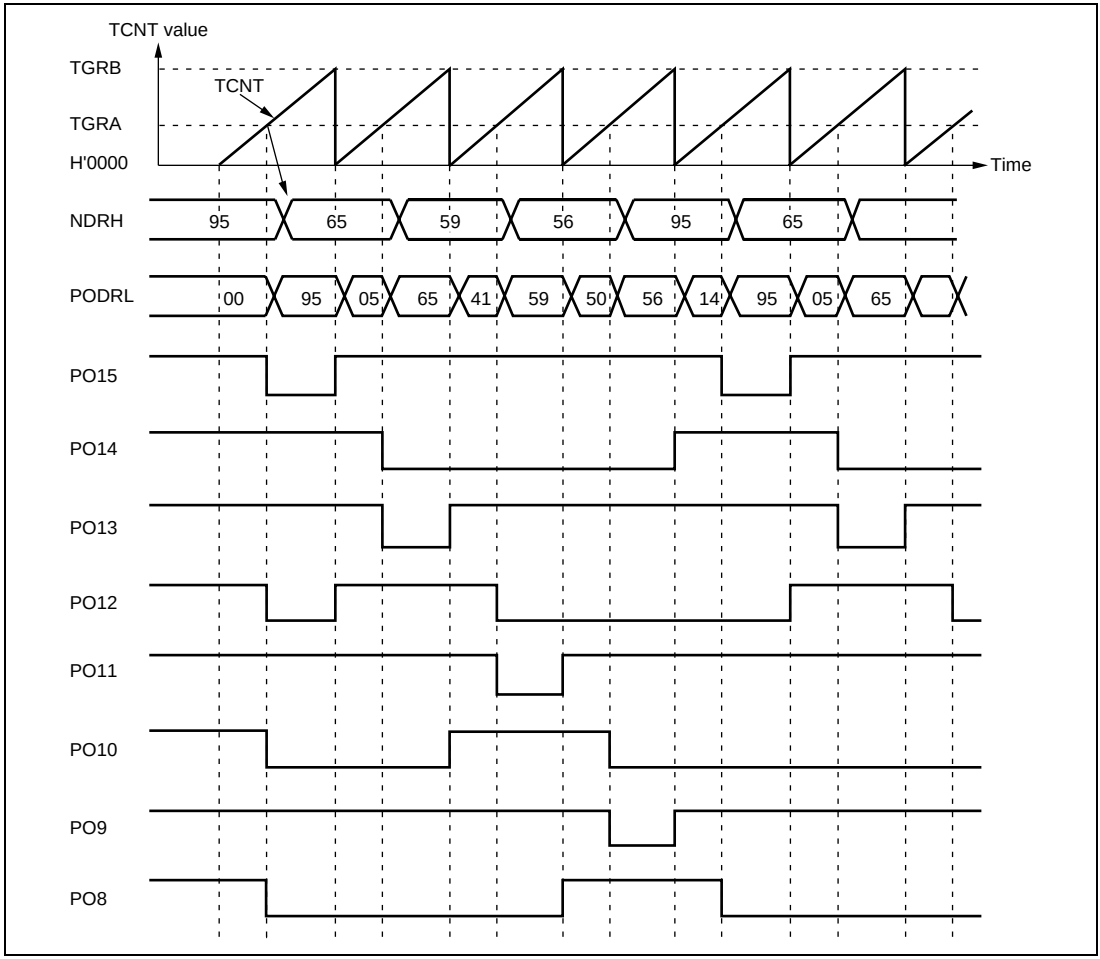



**Figure 12.10 Inverted Pulse Output (Example)**

### 12.4.9 Pulse Output Triggered by Input Capture

Pulse output can be triggered by TPU input capture as well as by compare match. If TGRA functions as an input capture register in the TPU channel selected by PCR, pulse output will be triggered by the input capture signal.

Figure 12.11 shows the timing of this output.

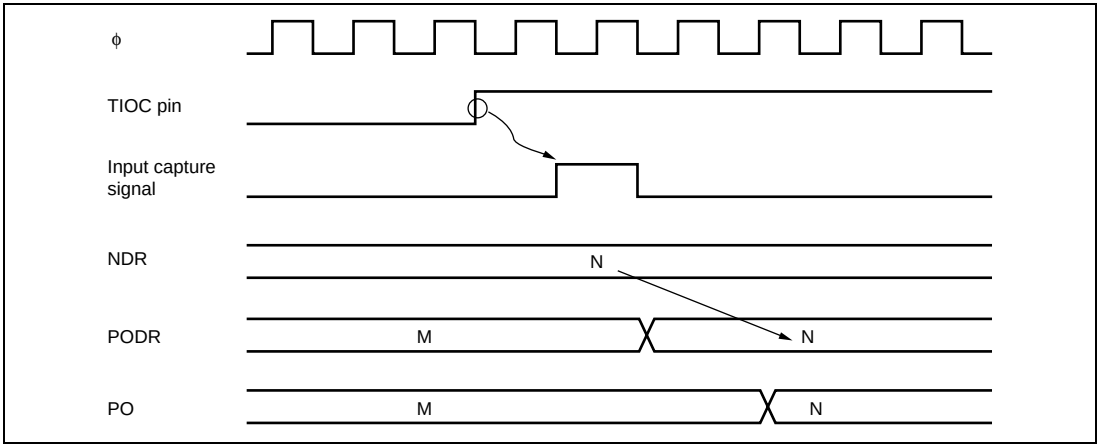



**Figure 12.11 Pulse Output Triggered by Input Capture (Example)**

## 12.5 Usage Notes

### 12.5.1 Module Stop Mode Setting

PPG operation can be disabled or enabled using the module stop control register. The initial setting is for PPG operation to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 12.5.2 Operation of Pulse Output Pins

Pins PO8 to PO15 are also used for other peripheral functions such as the TPU. When output by another peripheral function is enabled, the corresponding pins cannot be used for pulse output. Note, however, that data transfer from NDR bits to PODR bits takes place, regardless of the usage of the pins.

Pin functions should be changed only under conditions in which the output trigger event will not occur.

# Section 13 Watchdog Timer

The watchdog timer (WDT) 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.

When this watchdog function is not needed, the WDT can be used as an interval timer. In interval timer operation, an interval timer interrupt is generated each time the counter overflows.

The block diagram of the WDT is shown in figure 13.1.

## 13.1 Features

- Selectable from eight counter input clocks.
- Switchable between watchdog timer mode and interval timer mode

In watchdog timer mode

- If the counter overflows, it is possible to select whether this LSI is internally reset or not.

In interval timer mode

- If the counter overflows, the WDT generates an interval timer interrupt (WOVI).

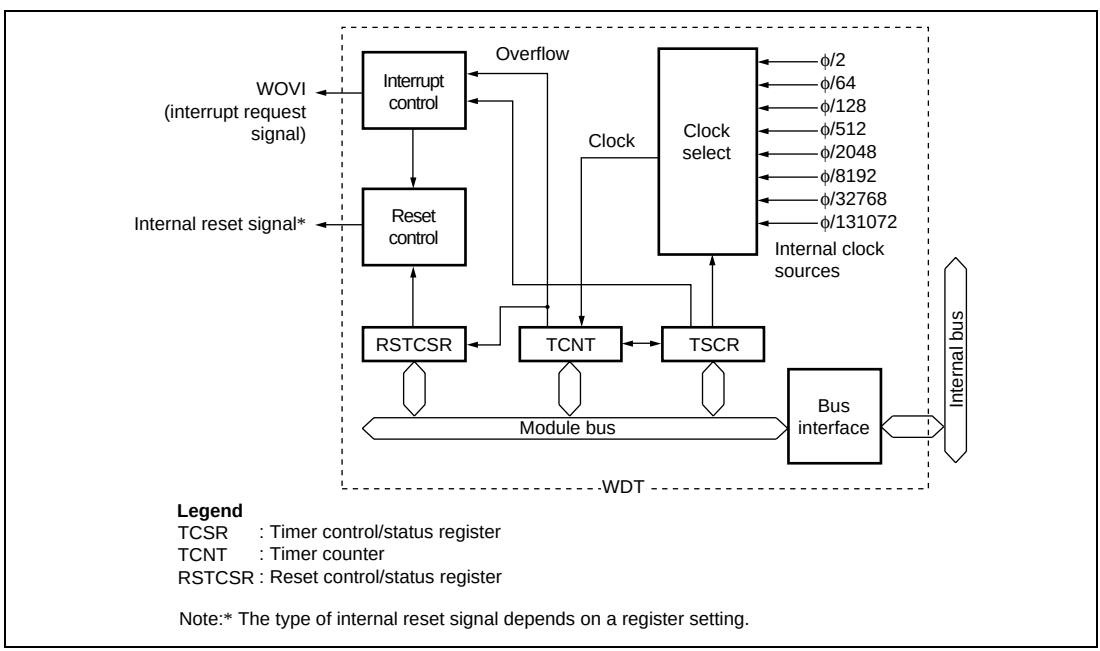



**Figure 13.1 Block Diagram of WDT**

## 13.2 Register Descriptions

The WDT has the following three registers. To prevent accidental overwriting, TCSR, TCNT, and RSTCSR have to be written to by a different method to normal registers. For details, refer to section 13.5.1, Notes on Register Access.

- Timer control/status register (TCSR)
- Timer counter (TCNT)
- Reset control/status register (RSTCSR)

### 13.2.1 Timer Counter (TCNT)

TCNT is an 8-bit readable/writable up-counter. TCNT is initialized to H′00 by a reset, when the TME bit in TCSR is cleared to 0.

### 13.2.2 Timer Control/Status Register (TCSR)

TCSR is an 8-bit readable/writable register. Its functions include selecting the clock source to be input to TCNT, and selecting the timer mode.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | OVF | 0 | R/(W)* | Overflow Flag<br>Indicates that TCNT has overflowed. Only a write of 0 is permitted, to clear the flag.<br>[Setting condition]<br>When TCNT overflows (changes from H′FF to H′00)<br>When internal reset request generation is selected in watchdog timer mode, OVF is cleared automatically by the internal reset.<br>[Clearing conditions]<br>Cleared by reading TCSR when OVF = 1, then writing 0 to OVF |
| 6 | WT/$\overline{IT}$ | 0 | R/W | Timer Mode Select<br>Selects whether the WDT is used as a watchdog timer or interval timer.<br>0: Interval timer mode<br>1: Watchdog timer mode |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | TME | 0 | R/W | Timer Enable<br>When this bit is set to 1, TCNT starts counting. When this bit is cleared, TCNT stops counting and is initialized to H′00. |
| 4, 3 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |
| 2 | CKS2 | 0 | R/W | Clock Select 0 to 2<br>Selects the clock source to be input to TCNT. The overflow frequency for $\phi$ = 20 MHz is enclosed in parentheses.<br>000: Clock $\phi$/2 (frequency: 25.6 μs)<br>001: Clock $\phi$/64 (frequency: 819.2 μs)<br>010: Clock $\phi$/128 (frequency: 1.6 ms)<br>011: Clock $\phi$/512 (frequency: 6.6 ms)<br>100: Clock $\phi$/2048 (frequency: 26.2 ms)<br>101: Clock $\phi$/8192 (frequency: 104.9 ms)<br>110: Clock $\phi$/32768 (frequency: 419.4 ms)<br>111: Clock $\phi$/131072 (frequency: 1.68 s) |
| 1 | CKS1 | 0 | R/W | |
| 0 | CKS0 | 0 | R/W | |

Note:* Only 0 can be written, for flag clearing.

### 13.2.3 Reset Control/Status Register (RSTCSR)

RSTCSR is an 8-bit readable/writable register that controls the generation of the internal reset signal when TCNT overflows, and selects the type of internal reset signal. RSTCSR is initialized to H′1F by a reset signal from the $\overline{\text{RES}}$ pin, and not by the WDT internal reset signal caused by overflows.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | WOVF | 0 | R/(W)* | Watchdog Overflow Flag<br>This bit is set when TCNT overflows in watchdog timer mode. This bit cannot be set in interval timer mode, and only 0 can be written.<br>[Setting condition]<br>Set when TCNT overflows (changed from H′FF to H′00) in watchdog timer mode<br>[Clearing condition]<br>Cleared by reading RSTCSR when WOVF = 1, and then writing 0 to WOVF |
| 6 | RSTE | 0 | R/W | Reset Enable<br>Specifies whether or not a reset signal is generated in the chip if TCNT overflows during watchdog timer operation.<br>0: Reset signal is not generated even if TCNT overflows<br>(Though this LSI is not reset, TCNT and TCSR in WDT are reset)<br>1: Reset signal is generated if TCNT overflows |
| 5 | RSTS | 0 | R/W | Reset Select<br>Selects the type of internal reset generated if TCNT overflows during watchdog timer operation.<br>0: Power-on reset<br>1: Setting prohibited |
| 4 to 0 | — | All 1 | — | Reserved<br>These bits are always read as 1 and cannot be modified. |

Note:* Only 0 can be written, for flag clearing.

## 13.3 Operation

### 13.3.1 Watchdog Timer Mode Operation

To use the WDT as a watchdog timer, set the WT/$\overline{IT}$ bit in TCSR and the TME bit to 1. Software must prevent TCNT overflows by rewriting the TCNT value (normally by writing H′00) before overflow occurs. This ensures that TCNT does not overflow while the system is operating normally. If TCNT overflows without being rewritten because of a system malfunction or other error, the WOVF bit in RSTCSR is set to 1. If the RSTE bit in RSTCSR is set to 1, an internal reset is issued. This is shown in figure 13.2. At this time, select the power-on reset by clearing the RSTS bit in RSTCSR to 0. The internal reset signal is output for 518 states.

If a reset caused by a signal input to the $\overline{RES}$ pin occurs at the same time as a reset caused by a WDT overflow, the reset by the $\overline{RES}$ pin has priority and the WOVF bit in RSTCSR is cleared to 0.

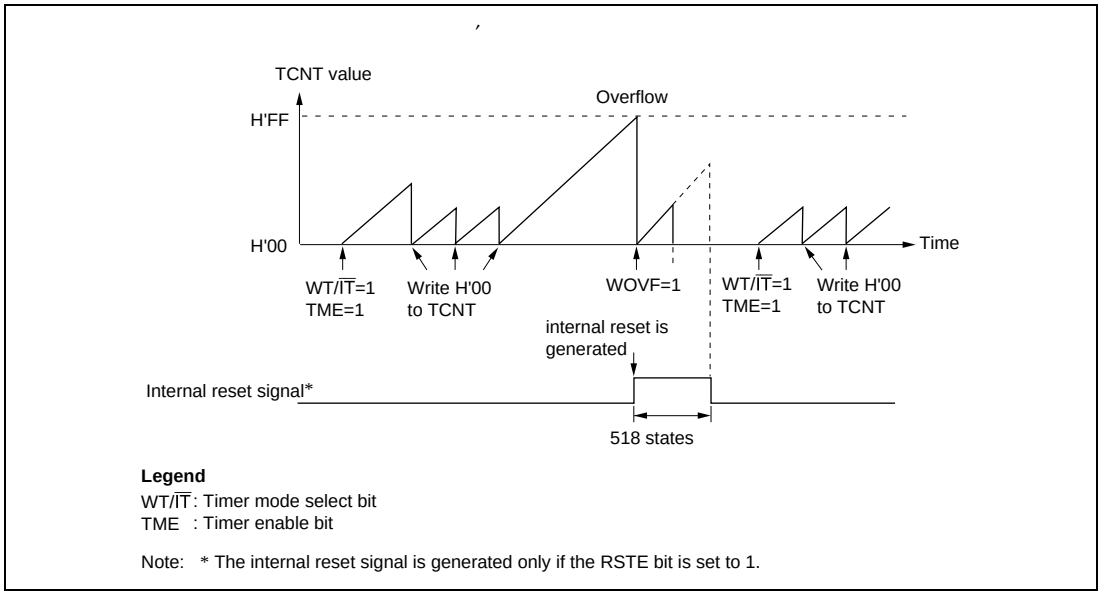



**Figure 13.2 Example of WDT0 Watchdog Timer Operation**

### 13.3.2 Interval Timer Mode

When the WDT is used as an interval timer, an interval timer interrupt (WOVI) is generated each time the TCNT overflows. Therefore, an interrupt can be generated at intervals.

When the TCNT overflows in interval timer mode, an interval timer interrupt (WOVI) is requested at the time the OVF bit of the TCSR is set to 1.

## 13.4 Interrupts

During interval timer mode operation, an overflow generates an interval timer interrupt (WOVI). The interval timer interrupt is requested whenever the OVF flag is set to 1 in TCSR. OVF must be cleared to 0 in the interrupt handling routine.

**Table 13.1 WDT Interrupt Source**

| Name | Interrupt Source | Interrupt Flag | DTC Activation |
|---|---|---|---|
| WOVI | TCNT overflow | WOVF | Impossible |

## 13.5 Usage Notes

### 13.5.1 Notes on Register Access

The watchdog timer's TCNT, TCSR, and RSTCSR registers differ from other registers in being more difficult to write to. The procedures for writing to and reading these registers are given below.

**Writing to TCNT, TCSR, and RSTCSR:** To write to TCNT and TCSR, execute a word transfer instruction. They cannot be written to by a byte transfer instruction.

TCNT and TCSR both have the same write address. Therefore, the relative condition shown in figure 13.3 needs to be satisfied in order to write to TCNT or TCSR. The transfer instruction writes the lower byte data to TCNT or TCSR according to the satisfied condition.

To write to RSTCSR, execute a word transfer instruction for address H′FF76. A byte transfer instruction cannot write to RSTCSR.

The method of writing 0 to the WOVF bit differs from that of writing to the RSTE and RSTS bits. To write 0 to the WOVF bit, satisfy the condition shown in figure 13.3. If satisfied, the transfer instruction clears the WOVF bit to 0, but has no effect on the RSTE and RSTS bits. To write to the RSTE and RSTS bits, satisfy the condition shown in figure 13.3. If satisfied, the transfer instruction writes the values in bits 5 and 6 of the lower byte into the RSTE and RSTS bits, respectively, but has no effect on the WOVF bit.

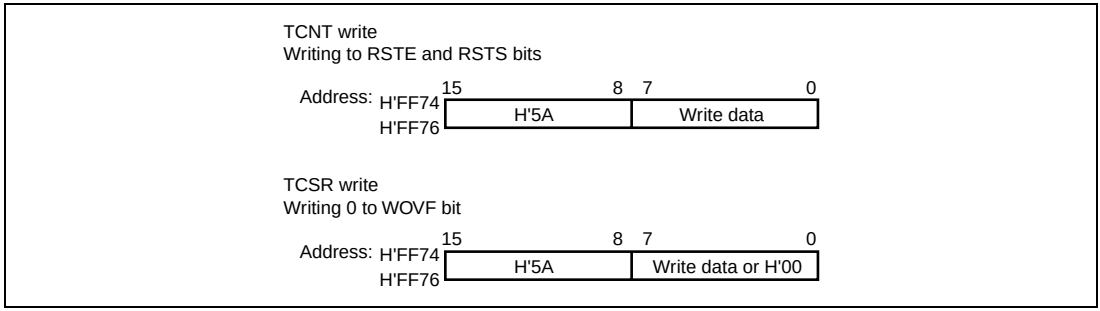



**Figure 13.3 Writing to TCNT, TCSR, and RSTCSR (example for WDT0)**

**Reading TCNT, TCSR, and RSTCSR (WDT0):** These registers are read in the same way as other registers. The read addresses are H′FF74 for TCSR, H′FF75 for TCNT, and H′FF77 for RSTCSR.

### 13.5.2 Conflict between Timer Counter (TCNT) Write and Increment

If a timer counter clock pulse is generated during the T2 state of a TCNT write cycle, the write takes priority and the timer counter is not incremented. Figure 13.4 shows this operation.

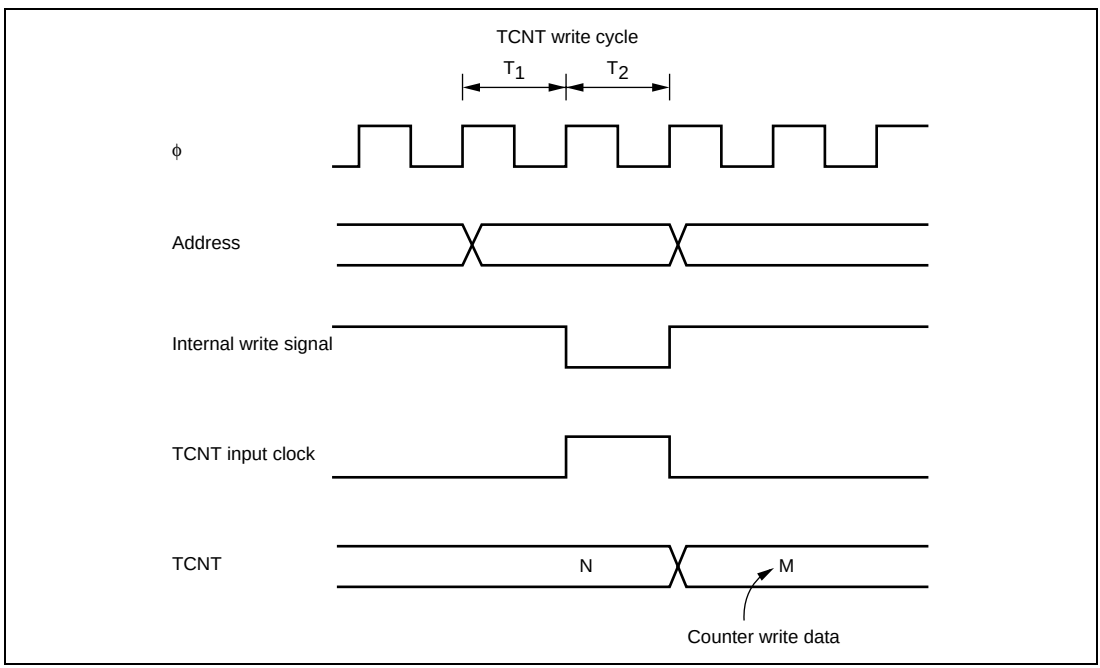



**Figure 13.4 Conflict between TCNT Write and Increment**

### 13.5.3 Changing Value of CKS2 to CKS0

If bits CKS0 to CKS2 in TCSR are written to while the WDT is operating, errors could occur in the incrementation. Software must be used to stop the watchdog timer (by clearing the TME bit to 0) before changing the value of bits CKS0 to CKS2.

### 13.5.4 Switching between Watchdog Timer Mode and Interval Timer Mode

If the mode is switched from watchdog timer to interval timer while the WDT is operating, errors could occur in the incrementation. Software must be used to stop the watchdog timer (by clearing the TME bit to 0) before switching the mode.

### 13.5.5 Internal Reset in Watchdog Timer Mode

This LSI is not reset internally if TCNT overflows while the RSTE bit is cleared to 0 during watchdog timer operation, however TCNT and TCSR of the WDT are reset.

TCNT, TCSR, or RSTCR cannot be written to for 132 states following an overflow. During this period, any attempt to read the WOVF flag is not acknowledged. Accordingly, wait 132 states after overflow to write 0 to the WOVF flag for clearing.

### 13.5.6 OVF Flag Clearing in Interval Timer Mode

When the OVF flag setting conflicts with the OVF flag reading in interval timer mode, writing 0 to the OVF bit may not clear the flag even though the OVF bit has been read while it is 1. If there is a possibility that the OVF flag setting and reading will conflict, such as when the OVF flag is polled with the interval timer interrupt disabled, read the OVF bit while it is 1 at least twice before writing 0 to the OVF bit to clear the flag.

# Section 14 Serial Communication Interface (SCI)

This LSI has two independent serial communication interface (SCI) channels. The SCI can handle both asynchronous and clocked synchronous serial communication. 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). The SCI also supports an IC card (Smart Card) interface conforming to ISO/IEC 7816-3 (Identification Card) as a serial communication interface extension function.

Figure 14.1 shows a block diagram of the SCI.

## 14.1 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 can be selected as a transfer clock source (except for in Smart Card interface mode).
- Choice of LSB-first or MSB-first transfer (except in the case of asynchronous mode 7-bit data)
- Four interrupt sources

  Transmit-end, transmit-data-empty, receive-data-full, and receive error — that can issue requests.

  The transmit-data-empty interrupt and receive data full interrupts can be used to activate the data transfer controller (DTC).
- Module stop mode can be set

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

Smart Card Interface

- Automatic transmission of error signal (parity error) in receive mode
- Error signal detection and automatic data retransmission in transmit mode
- Direct convention and inverse convention both supported

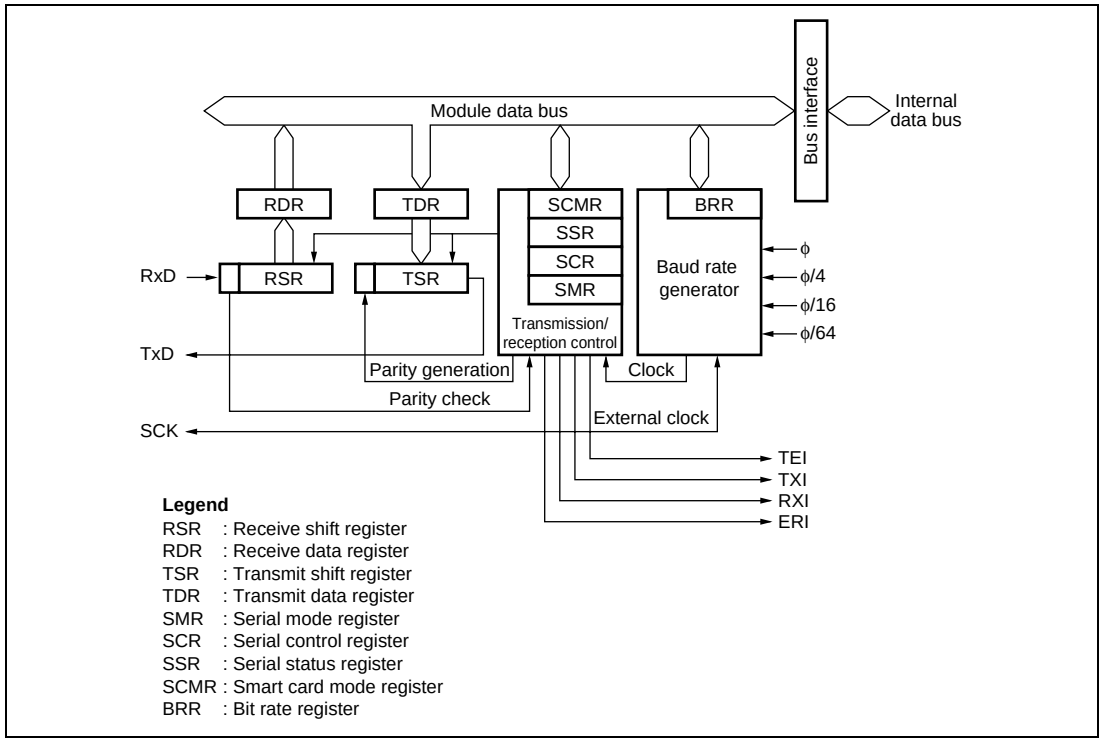



**Figure 14.1 Block Diagram of SCI**

## 14.2 Input/Output Pins

Table 14.1 shows the serial pins for each SCI channel.

**Table 14.1 Pin Configuration**

| Channel | Pin Name* | I/O | Function |
|---|---|---|---|
| 0 | SCK0 | I/O | SCI0 clock input/output |
| | RxD0 | Input | SCI0 receive data input |
| | TxD0 | Output | SCI0 transmit data output |
| 2 | SCK2 | I/O | SCI2 clock input/output |
| | RxD2 | Input | SCI2 receive data input |
| | TxD2 | Output | SCI2 transmit data output |

Note:* Pin names SCK, RxD, and TxD are used in the text for all channels, omitting the channel designation.

## 14.3 Register Descriptions

The SCI has the following registers for each channel. The serial mode register (SMR), serial status register (SSR), and serial control register (SCR) are described separately for normal serial communication interface mode and Smart Card interface mode because their bit functions differ in part.

- Receive Shift Register (RSR)
- Receive Data Register (RDR)
- Transmit Data Register (TDR)
- Transmit Shift Register (TSR)
- Serial Mode Register (SMR)
- Serial Control Register (SCR)
- Serial Status Register (SSR)
- Smart Card Mode Register (SCMR)
- Bit Rate Register (BRR)

### 14.3.1 Receive Shift Register (RSR)

RSR is a shift register that is used to receive serial data input to 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.

### 14.3.2 Receive Data Register (RDR)

RDR is an 8-bit register that stores received data. When the SCI 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.

### 14.3.3 Transmit Data Register (TDR)

TDR is an 8-bit register that stores data for transmission. When the SCI 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 serial transmission, the SCI transfers the written data to TSR to continue transmission. Although TDR can be read or written to by the CPU at all times, to achieve reliable serial transmission, write transmit data to TDR only once after confirming that the TDRE bit in SSR is set to 1.

### 14.3.4 Transmit Shift Register (TSR)

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

### 14.3.5 Serial Mode Register (SMR)

SMR is used to set the SCI's serial transfer format and select the baud rate generator clock source.

Some bit functions of SMR differ between normal serial communication interface mode and Smart Card interface mode.

- Normal Serial Communication Interface Mode (When SMIF in SCMR is 0)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | C/$\overline{A}$ | 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. LSB-first is fixed and the MSB of TDR is not transmitted in transmission.<br>In clocked synchronous mode, a fixed data length of 8 bits is used. |
| 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. For a multiprocessor format, parity bit addition and checking are not performed regardless of the PE bit setting. |
| 4 | O/$\overline{E}$ | 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>In reception, only the first stop bit is checked. If the second stop bit is 0, it is treated as the start bit of the next transmit character. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | MP | 0 | R/W | Multiprocessor Mode (enabled only in asynchronous mode)<br>When this bit is set to 1, the multiprocessor communication function is enabled. The PE bit and O/$\overline{E}$ bit settings are invalid in multiprocessor mode. |
| 1 | CKS1 | 0 | R/W | Clock Select 0 and 1:<br>These bits select the clock source for the baud rate generator.<br>00: $\phi$ clock (n = 0)<br>01: $\phi$/4 clock (n = 1)<br>10: $\phi$/16 clock (n = 2)<br>11: $\phi$/64 clock (n = 3)<br>For the relationship between the bit rate register setting and the baud rate, see section 14.3.9, Bit Rate Register (BRR). n is the decimal representation of the value of n in BRR (see section 14.3.9, Bit Rate Register (BRR)). |
| 0 | CKS0 | 0 | R/W | |

- Smart Card Interface Mode (When SMIF in SCMR is 1)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | GM | 0 | R/W | GSM Mode<br>When this bit is set to 1, the SCI operates in GSM mode. In GSM mode, the timing of the TEND setting is advanced by 11.0 etu (Elementary Time Unit: the time for transfer of one bit), and clock output control mode addition is performed. For details, refer to section 14.7.8, Clock Output Control. |
| 6 | BLK | 0 | R/W | When this bit is set to 1, the SCI operates in block transfer mode. For details on block transfer mode, refer to section 14.7.3, Block Transfer Mode. |
| 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 in transmission, and the parity bit is checked in reception. In Smart Card interface mode, this bit must be set to 1. |
| 4 | O/$\overline{E}$ | 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.<br>For details on setting this bit in Smart Card interface mode, refer to section 14.7.2, Data Format (Except for Block Transfer Mode). |
| 3 | BCP1 | 0 | R/W | Basic Clock Pulse 1 and 2 |
| 2 | BCP0 | 0 | R/W | These bits specify the number of basic clock periods in a 1-bit transfer interval on the Smart Card interface.<br>00: 32 clock (S = 32)<br>01: 64 clock (S = 64)<br>10: 372 clock (S = 372)<br>11: 256 clock (S = 256)<br>For details, refer to section 14.7.4, Receive Data Sampling Timing and Reception Margin in Smart Card Interface Mode. S stands for the value of S in BRR (see section 14.3.9, Bit Rate Register (BRR)). |

| 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 baud rate generator.<br>00: $\phi$ clock (n = 0)<br>01: $\phi$/4 clock (n = 1)<br>10: $\phi$/16 clock (n = 2)<br>11: $\phi$/64 clock (n = 3)<br>For the relationship between the bit rate register setting and the baud rate, see section 14.3.9, Bit Rate Register (BRR). n is the decimal representation of the value of n in BRR (see section 14.3.9, Bit Rate Register (BRR)). |

### 14.3.6 Serial Control Register (SCR)

SCR is a register that enables or disables SCI transfer operations and interrupt requests, and is also used to selection of the transfer clock source. For details on interrupt requests, refer to section 14.8, Interrupt Sources. Some bit functions of SCR differ between normal serial communication interface mode and Smart Card interface mode.

- Normal Serial Communication Interface Mode (When SMIF in SCMR is 0)

| 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 s set to 1, transmission is enabled. |
| 4 | RE | 0 | R/W | Receive Enable<br>When this bit is set to 1, reception is enabled. |
| 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 ORER 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 14.5, Multiprocessor Communication Function. |
| 2 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>This bit is set to 1, TEI interrupt request is enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | CKE1 | 0 | R/W | Clock Enable 0 and 1 |
| 0 | CKE0 | 0 | R/W | Selects the clock source and SCK pin function.<br>Asynchronous mode<br>00: Internal baud rate generator<br>SCK pin functions as I/O port<br>01: Internal baud rate generator<br>Outputs a clock of the same frequency as the bit rate from the SCK pin.<br>1X: External clock<br>Inputs a clock with a frequency 16 times the bit rate from the SCK pin.<br>Clocked synchronous mode<br>0X: Internal clock (SCK pin functions as clock output)<br>1X: External clock (SCK pin functions as clock input) |

**Legend**

X: Don’t care

- Smart Card Interface Mode (When SMIF in SCMR is 1)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TIE | 0 | R/W | Transmit Interrupt Enable<br>When this bit is set to 1, 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. |
| 3 | MPIE | 0 | R/W | Multiprocessor Interrupt Enable (enabled only when the MP bit in SMR is 1 in asynchronous mode)<br>Write 0 to this bit in Smart Card interface mode. |
| 2 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>Write 0 to this bit in Smart Card interface mode. |
| 1 | CKE1 | 0 | R/W | Clock Enable 0 and 1 |
| 0 | CKE0 | 0 | R/W | Enables or disables clock output from the SCK pin. The clock output can be dynamically switched in GSM mode. For details, refer to section 14.7.8, Clock Output Control.<br>When the GM bit in SMR is 0:<br>00: Output disabled (SCK pin can be used as an I/O port pin)<br>01: Clock output<br>1X: Reserved<br>When the GM bit in SMR is 1:<br>00: Output fixed low<br>01: Clock output<br>10: Output fixed high<br>11: Clock output |

**Legend**

X: Don’t care

### 14.3.7 Serial Status Register (SSR)

SSR is a register containing status flags of the SCI and multiprocessor bits for transfer. 1 cannot be written to flags TDRE, RDRF, ORER, PER, and FER; they can only be cleared. Some bit functions of SSR differ between normal serial communication interface mode and Smart Card interface mode.

- Normal Serial Communication Interface Mode (When SMIF in SCMR is 0)

| 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 SCR is 0<br>• When data is transferred from TDR to TSR and data can be written to TDR<br><br>[Clearing conditions]<br>• When 0 is written to TDRE after reading TDRE = 1<br>• When the DTC is activated by a TXI interrupt request and writes data 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><br>[Clearing conditions]<br>• When 0 is written to RDRF after reading RDRF = 1<br>• When the DTC is activated by an RXI interrupt and transferred data from RDR<br><br>The RDRF flag is not affected and retains their previous values when the RE bit in SCR is cleared to 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | ORER | 0 | R/W | Overrun Error<br>[Setting condition]<br>• When the next serial reception is completed while RDRF = 1<br>[Clearing condition]<br>• When 0 is written to ORER after reading ORER = 1 |
| 4 | FER | 0 | R/W | Framing Error<br>[Setting condition]<br>• When the stop bit is 0<br>[Clearing condition]<br>• When 0 is written to FER after reading FER = 1<br>In 2-stop-bit mode, only the first stop bit is checked. |
| 3 | PER | 0 | R/W | Parity Error<br>[Setting condition]<br>• When a parity error is detected 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 SCR 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 TDRE after reading TDRE = 1<br>• When the DTC is activated by a TXI interrupt and writes data to TDR |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | MPB | 0 | R | Multiprocessor Bit<br>MPB stores the multiprocessor bit in the receive data. When the RE bit in SCR 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 data. |

- Smart Card Interface Mode (When SMIF in SCMR is 1)

| 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 SCR is 0<br>• When data is transferred from TDR to TSR and data can be written to TDR<br>[Clearing conditions]<br>• When 0 is written to TDRE after reading TDRE = 1<br>• When the DTC is activated by a TXI interrupt request and writes data 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 the DTC is activated by an RXI interrupt and transferred data from RDR<br>The RDRF flag is not affected and retains their previous values when the RE bit in SCR is cleared to 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | ORER | 0 | R/W | Overrun Error<br>[Setting condition]<br>• When the next serial reception is completed while RDRF = 1<br>[Clearing condition]<br>• When 0 is written to ORER after reading ORER = 1 |
| 4 | ERS | 0 | R/W | Error Signal Status<br>[Setting condition]<br>• When the low level of the error signal is sampled<br>[Clearing conditions]<br>• When 0 is written to ERS after reading ERS = 1 |
| 3 | PER | 0 | R/W | Parity Error<br>[Setting condition]<br>• When a parity error is detected during reception<br>[Clearing condition]<br>• When 0 is written to PER after reading PER = 1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | TEND | 1 | R | Transmit End<br>This bit is set to 1 when no error signal has been sent back from the receiving end and the next transmit data is ready to be transferred to TDR.<br>[Setting conditions]<br>• When the TE bit in SCR is 0 and the ERS bit is also 0<br>• When the ERS bit is 0 and the TDRE bit is 1 after the specified interval following transmission of 1-byte data.<br>The timing of bit setting differs according to the register setting as follows:<br>When GM = 0 and BLK = 0, 2.5 etu after transmission starts<br>When GM = 0 and BLK = 1, 1.5 etu after transmission starts<br>When GM = 1 and BLK = 0, 1.0 etu after transmission starts<br>When GM = 1 and BLK = 1, 1.0 etu after transmission starts<br>[Clearing conditions]<br>• When 0 is written to TDRE after reading TDRE = 1<br>• When the DTC is activated by a TXI interrupt and writes data to TDR |
| 1 | MPB | 0 | R | Multiprocessor Bit<br>This bit is not used in Smart Card interface mode. |
| 0 | MPBT | 0 | R/W | Multiprocessor Bit Transfer<br>Write 0 to this bit in Smart Card interface mode. |

### 14.3.8 Smart Card Mode Register (SCMR)

SCMR is a register that selects Smart Card interface mode and its format.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 1 | — | Reserved<br>These bits are always read as 1. |
| 3 | SDIR | 0 | R/W | Smart Card Data Transfer Direction<br>Selects the serial/parallel conversion format.<br>0: LSB-first in transfer<br>1: MSB-first in transfer<br>The bit setting is valid only when the transfer data format is 8 bits. For 7-bit data, LSB-first is fixed. |
| 2 | SINV | 0 | R/W | Smart Card Data Invert<br>Specifies inversion of the data logic level. The SINV bit does not affect the logic level of the parity bit. To invert the parity bit, invert the O/$\overline{E}$ bit in SMR.<br>0: TDR contents are transmitted as they are. Receive data is stored as it is in RDR<br>1: TDR contents are inverted before being transmitted. Receive data is stored in inverted form in RDR |
| 1 | — | 1 | — | Reserved<br>This bit is always read as 1. |
| 0 | SMIF | 0 | R/W | Smart Card Interface Mode Select<br>This bit is set to 1 to make the SCI operate in Smart Card interface mode.<br>0: Normal asynchronous mode or clocked synchronous mode<br>1: Smart card interface mode |

### 14.3.9 Bit Rate Register (BRR)

BRR is an 8-bit register that adjusts the bit rate. As the SCI performs baud rate generator control independently for each channel, different bit rates can be set for each channel. Table 14.2 shows the relationships between the N setting in BRR and bit rate B for normal asynchronous mode, clocked synchronous mode, and Smart Card interface mode. The initial value of BRR is H′FF, and it can be read or written to by the CPU at all times.

**Table 14.2 The Relationships between The N Setting in BRR and Bit Rate B**

| Mode | BRR Setting N | Error |
|---|---|---|
| Asynchronous Mode | $N = \frac{\phi \times 10^6}{64 \times 2^{2n-1} \times B} - 1$ | $\text{Error (\%)} = \{ \frac{\phi \times 10^6}{B \times 64 \times 2^{2n-1} \times (N+1)} - 1 \} \times 100$ |
| Clocked Synchronous Mode | $N = \frac{\phi \times 10^6}{8 \times 2^{2n-1} \times B} - 1$ | |
| Smart Card Interface Mode | $N = \frac{\phi \times 10^6}{S \times 2^{2n+1} \times B} - 1$ | $\text{Error (\%)} = \{ \frac{\phi \times 10^6}{B \times S \times 2^{2n+1} \times (N+1)} - 1 \} \times 100$ |

Note: B: Bit rate (bit/s)
N: BRR setting for baud rate generator ($0 \le N \le 255$)
$\phi$: Operating frequency (MHz)
n and S: Determined by the SMR settings shown in the following tables.

| SMR Setting | | |
|---|---|---|
| CKS1 | CKS0 | n |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

| SMR Setting | | |
|---|---|---|
| BCP1 | BCP0 | S |
| 0 | 0 | 32 |
| 0 | 1 | 64 |
| 1 | 0 | 372 |
| 1 | 1 | 256 |

Table 14.3 shows sample N settings in BRR in normal asynchronous mode. Table 14.4 shows the maximum bit rate for each frequency in normal asynchronous mode. Table 14.6 shows sample N settings in BRR in clocked synchronous mode. Table 14.8 shows sample N settings in BRR in Smart Card interface mode. In Smart Card interface mode, S (the number of basic clock periods in a 1-bit transfer interval) can be selected. For details, refer to section 14.7.4, Receive Data Sampling Timing and Reception Margin in Smart Card Interface Mode. Tables 14.5 and 14.7 show the maximum bit rates with external clock input.

**Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (1)**

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

| Bit Rate (bit/s) | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | | | 6.144 | | | 7.3728 | | | 8 | | |
| | 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 | 7 | 0.00 |
| 38400 | 0 | 4 | –2.34 | 0 | 4 | 0.00 | 0 | 5 | 0.00 | — | — | — |

**Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (2)**

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 9.8304 | | | 10 | | | 12 | | | 12.288 | | |
| 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 |

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14 | | | 14.7456 | | | 16 | | | 17.2032 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 248 | –0.17 | 3 | 64 | 0.70 | 3 | 70 | 0.03 | 3 | 75 | 0.48 |
| 150 | 2 | 181 | 0.13 | 2 | 191 | 0.00 | 2 | 207 | 0.13 | 2 | 223 | 0.00 |
| 300 | 2 | 90 | 0.13 | 2 | 95 | 0.00 | 2 | 103 | 0.13 | 2 | 111 | 0.00 |
| 600 | 1 | 181 | 0.13 | 1 | 191 | 0.00 | 1 | 207 | 0.13 | 1 | 223 | 0.00 |
| 1200 | 1 | 90 | 0.13 | 1 | 95 | 0.00 | 1 | 103 | 0.13 | 1 | 111 | 0.00 |
| 2400 | 0 | 181 | 0.13 | 0 | 191 | 0.00 | 0 | 207 | 0.13 | 0 | 223 | 0.00 |
| 4800 | 0 | 90 | 0.13 | 0 | 95 | 0.00 | 0 | 103 | 0.13 | 0 | 111 | 0.00 |
| 9600 | 0 | 45 | –0.93 | 0 | 47 | 0.00 | 0 | 51 | 0.13 | 0 | 55 | 0.00 |
| 19200 | 0 | 22 | –0.93 | 0 | 23 | 0.00 | 0 | 25 | 0.13 | 0 | 27 | 0.00 |
| 31250 | 0 | 13 | 0.00 | 0 | 14 | –1.70 | 0 | 15 | 0.00 | 0 | 13 | 1.20 |
| 38400 | — | — | — | 0 | 11 | 0.00 | 0 | 12 | 0.13 | 0 | 13 | 0.00 |

**Table 14.3 BRR Settings for Various Bit Rates (Asynchronous Mode) (3)**

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 18 | | | 19.6608 | | | 20 | | | 24 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 79 | –0.12 | 3 | 86 | 0.31 | 3 | 88 | –0.25 | 3 | 106 | –0.44 |
| 150 | 2 | 233 | 0.16 | 2 | 255 | 0.00 | 3 | 64 | 0.16 | 3 | 77 | 0.16 |
| 300 | 2 | 116 | 0.16 | 2 | 127 | 0.00 | 2 | 129 | 0.16 | 2 | 155 | 0.16 |
| 600 | 1 | 233 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 | 2 | 77 | 0.16 |
| 1200 | 1 | 116 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 | 1 | 155 | 0.16 |
| 2400 | 0 | 233 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 | 1 | 77 | 0.16 |
| 4800 | 0 | 116 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 | 0 | 155 | 0.16 |
| 9600 | 0 | 58 | –0.69 | 0 | 63 | 0.00 | 0 | 64 | 0.16 | 0 | 77 | 0.16 |
| 19200 | 0 | 28 | 1.02 | 0 | 31 | 0.00 | 0 | 32 | –1.36 | 0 | 38 | 0.16 |
| 31250 | 0 | 17 | 0.00 | 0 | 19 | –1.70 | 0 | 19 | 0.00 | 0 | 23 | 0 |
| 38400 | 0 | 14 | –2.34 | 0 | 15 | 0.00 | 0 | 15 | 1.73 | 0 | 19 | –2.34 |

**Table 14.4 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 |
|---|---|---|---|---|---|---|---|
| 4 | 125000 | 0 | 0 | 12 | 375000 | 0 | 0 |
| 4.9152 | 153600 | 0 | 0 | 12.288 | 384000 | 0 | 0 |
| 5 | 156250 | 0 | 0 | 14 | 437500 | 0 | 0 |
| 6 | 187500 | 0 | 0 | 14.7456 | 460800 | 0 | 0 |
| 6.144 | 192000 | 0 | 0 | 16 | 500000 | 0 | 0 |
| 7.3728 | 230400 | 0 | 0 | 17.2032 | 537600 | 0 | 0 |
| 8 | 250000 | 0 | 0 | 18 | 562500 | 0 | 0 |
| 9.8304 | 307200 | 0 | 0 | 19.6608 | 614400 | 0 | 0 |
| 10 | 312500 | 0 | 0 | 20 | 625000 | 0 | 0 |
| | | | | 24 | 750000 | 0 | 0 |

**Table 14.5 Maximum Bit Rate with External Clock Input (Asynchronous Mode)**

| ϕ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bit/s) | ϕ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bit/s) |
|---|---|---|---|---|---|
| 4 | 1.0000 | 62500 | 12 | 3.0000 | 187500 |
| 4.9152 | 1.2288 | 76800 | 12.288 | 3.0720 | 192000 |
| 5 | 1.2500 | 78125 | 14 | 3.5000 | 218750 |
| 6 | 1.5000 | 93750 | 14.7456 | 3.6864 | 230400 |
| 6.144 | 1.5360 | 96000 | 16 | 4.0000 | 250000 |
| 7.3728 | 1.8432 | 115200 | 17.2032 | 4.3008 | 268800 |
| 8 | 2.0000 | 125000 | 18 | 4.5000 | 281250 |
| 9.8304 | 2.4576 | 153600 | 19.6608 | 4.9152 | 307200 |
| 10 | 2.5000 | 156250 | 20 | 5.0000 | 312500 |
| | | | 24 | 6.0000 | 375000 |

**Table 14.6 BRR Settings for Various Bit Rates (Clocked Synchronous Mode)**

| Bit Rate (bit/s) | Operating Frequency φ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 4 | | 8 | | 10 | | 16 | | 20 | | 24 | |
| | n | N | n | N | n | N | n | N | n | N | n | N |
| 110 | — | — | | | | | | | | | | |
| 250 | 2 | 249 | 3 | 124 | — | — | 3 | 249 | | | | |
| 500 | 2 | 124 | 2 | 249 | — | — | 3 | 124 | — | — | — | — |
| 1k | 1 | 249 | 2 | 124 | — | — | 2 | 249 | — | — | — | — |
| 2.5k | 1 | 99 | 1 | 199 | 1 | 249 | 2 | 99 | 2 | 124 | 2 | 149 |
| 5k | 0 | 199 | 1 | 99 | 1 | 124 | 1 | 199 | 1 | 249 | 2 | 74 |
| 10k | 0 | 99 | 0 | 199 | 0 | 249 | 1 | 99 | 1 | 124 | 1 | 149 |
| 25k | 0 | 39 | 0 | 79 | 0 | 99 | 0 | 159 | 0 | 199 | 1 | 59 |
| 50k | 0 | 19 | 0 | 39 | 0 | 49 | 0 | 79 | 0 | 99 | 1 | 29 |
| 100k | 0 | 9 | 0 | 19 | 0 | 24 | 0 | 39 | 0 | 49 | 0 | 59 |
| 250k | 0 | 3 | 0 | 7 | 0 | 9 | 0 | 15 | 0 | 19 | 0 | 23 |
| 500k | 0 | 1 | 0 | 3 | 0 | 4 | 0 | 7 | 0 | 9 | 0 | 11 |
| 1M | 0 | 0* | 0 | 1 | | | 0 | 3 | 0 | 4 | 0 | 5 |
| 2.5M | | | | | 0 | 0* | | | 0 | 1 | — | — |
| 5M | | | | | | | | | 0 | 0* | — | — |

**Legend**

Blank : Setting prohibited.

— : Can be set, but there will be a degree of error.

* : Continuous transfer is not possible.

**Table 14.7 Maximum Bit Rate with External Clock Input (Clocked Synchronous Mode)**

| φ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bit/s) | φ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bit/s) |
|---|---|---|---|---|---|
| 4 | 0.6667 | 666666.7 | 14 | 2.3333 | 2333333.3 |
| 6 | 1.0000 | 1.000000.0 | 16 | 2.6667 | 2666666.7 |
| 8 | 1.3333 | 1333333.3 | 18 | 3.0000 | 3000000.0 |
| 10 | 1.6667 | 1666666.7 | 20 | 3.3333 | 3333333.3 |
| 12 | 2.0000 | 2000000.0 | 24 | 4 | 4000000.0 |

**Table 14.8 Examples of Bit Rate for Various BRR Settings (Smart Card Interface Mode) (When n = 0 and S = 372)**

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7.1424 | | | 10.00 | | | 10.7136 | | | 13.00 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 9600 | 0 | 0 | 0.00 | 0 | 1 | 30 | 0 | 1 | 25 | 0 | 1 | 8.99 |

| | Operating Frequency ϕ (MHz) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14.2848 | | | 16.00 | | | 18.00 | | | 20.00 | | | 24.00 | | |
| Bit Rate (bit/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 9600 | 0 | 1 | 0.00 | 0 | 1 | 12.01 | 0 | 2 | 15.99 | 0 | 2 | 6.60 | 0 | 2 | 12.01 |

**Table 14.9 Maximum Bit Rate at Various Frequencies (Smart Card Interface Mode) (when S = 372)**

| ϕ (MHz) | Maximum Bit Rate (bit/s) | n | N |
|---|---|---|---|
| 7.1424 | 9600 | 0 | 0 |
| 10.00 | 13441 | 0 | 0 |
| 10.7136 | 14400 | 0 | 0 |
| 13.00 | 17473 | 0 | 0 |

| ϕ (MHz) | Maximum Bit Rate (bit/s) | n | N |
|---|---|---|---|
| 14.2848 | 19200 | 0 | 0 |
| 16.00 | 21505 | 0 | 0 |
| 18.00 | 24194 | 0 | 0 |
| 20.00 | 26882 | 0 | 0 |
| 24.00 | 32258 | 0 | 0 |

## 14.4 Operation in Asynchronous Mode

Figure 14.2 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). In asynchronous serial communication, the transmission line is usually held in the mark state (high level). The SCI monitors the transmission line. When the transmission line goes to the space state (low level), the SCI recognizes a start bit and starts serial communication. In asynchronous serial communication, the communication line is usually held in the mark state (high level). The SCI monitors the communication line, and when it goes to the space state (low level), recognizes a start bit and starts serial communication. Inside the SCI, 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.

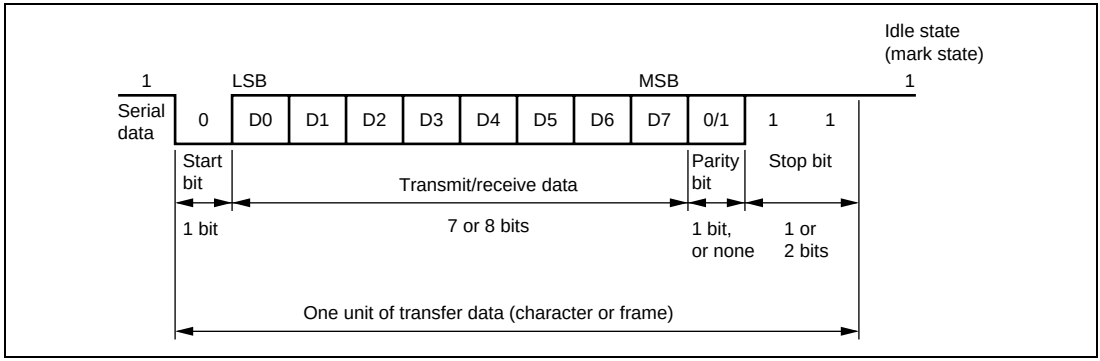



**Figure 14.2 Data Format in Asynchronous Communication (Example with 8-Bit Data, Parity, Two Stop Bits)**

### 14.4.1 Data Transfer Format

Table 14.10 shows the data transfer formats that can be used in asynchronous mode. Any of 12 transfer formats can be selected according to the SMR setting. For details on the multiprocessor bit, refer to section 14.5, Multiprocessor Communication Function.

**Table 14.10 Serial Transfer Formats (Asynchronous Mode)**

| SMR Settings | | | | Serial Transfer Format and Frame Length |
|---|---|---|---|---|
| CHR | PE | MP | STOP | 1 2 3 4 5 6 7 8 9 10 11 12 |
| 0 | 0 | 0 | 0 | S 8-bit data STOP |
| 0 | 0 | 0 | 1 | S 8-bit data STOP STOP |
| 0 | 1 | 0 | 0 | S 8-bit data P STOP |
| 0 | 1 | 0 | 1 | S 8-bit data P STOP STOP |
| 1 | 0 | 0 | 0 | S 7-bit data STOP |
| 1 | 0 | 0 | 1 | S 7-bit data STOP STOP |
| 1 | 1 | 0 | 0 | S 7-bit data P STOP |
| 1 | 1 | 0 | 1 | S 7-bit data P STOP STOP |
| 0 | — | 1 | 0 | S 8-bit data MPB STOP |
| 0 | — | 1 | 1 | S 8-bit data MPB STOP STOP |
| 1 | — | 1 | 0 | S 7-bit data MPB STOP |
| 1 | — | 1 | 1 | S 7-bit data MPB STOP STOP |

**Legend**

S : Start bit
STOP : Stop bit
P : Parity bit
MPB : Multiprocessor bit

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

In asynchronous mode, the SCI operates on a basic clock with a frequency of 16 times the transfer rate. In reception, the SCI 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 14.3. Thus, the reception margin in asynchronous mode is given by formula (1) below.

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

... Formula (1)

Where N : Ratio of bit rate to clock (N = 16)
D : Clock duty cycle (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 cycle) = 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.

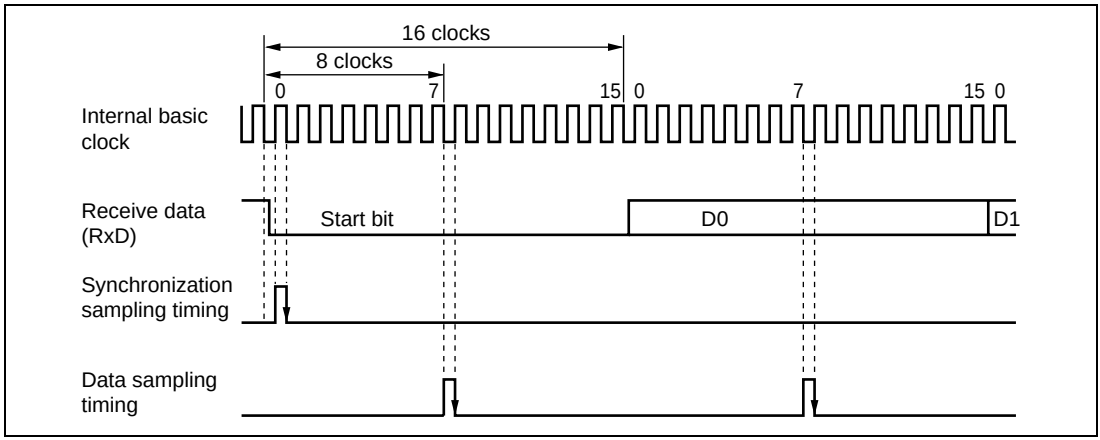



**Figure 14.3 Receive Data Sampling Timing in Asynchronous Mode**

### 14.4.3 Clock

Either an internal clock generated by the on-chip baud rate generator or an external clock input at the SCK pin can be selected as the SCI's serial clock, according to the setting of the C/$\overline{A}$ bit in SMR and the CKE0 and CKE1 bits in SCR. When an external clock is input at the SCK pin, the clock frequency should be 16 times the bit rate used.

When the SCI is operated on an internal clock, the clock can be output from the SCK 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 14.4.

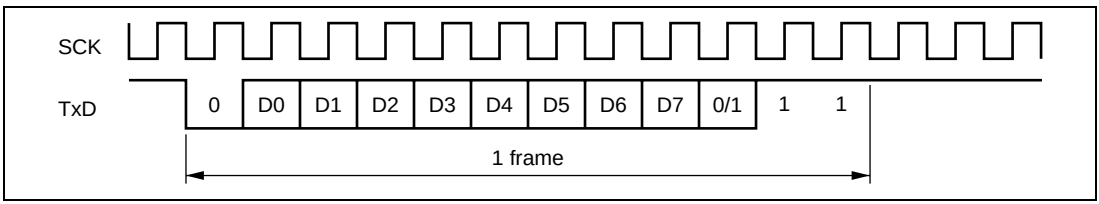
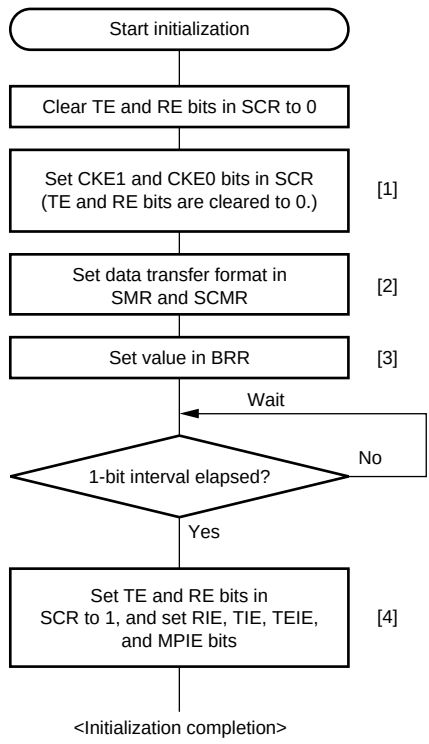



**Figure 14.4 Relationship between Output Clock and Transfer Data Phase (Asynchronous Mode)**

### 14.4.4 SCI Initialization (Asynchronous Mode)

Before transmitting and receiving data, first clear the TE and RE bits in SCR to 0, then initialize the SCI as described below. When the operating mode, or transfer format, is changed for example, the TE and RE bits must be cleared to 0 before making the change using the following procedure. 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 ORER 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 SCR. Be sure to clear bits RIE, TIE, TEIE, and MPIE, and bits TE and RE, to 0.

When the clock is selected in asynchronous mode, it is output immediately after SCR settings are made.

[2] Set the data transfer format in SMR and SCMR.

[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 SCR to 1. Also set the RIE, TIE, TEIE, and MPIE bits.

Setting the TE and RE bits enables the TxD and RxD pins to be used.

**Figure 14.5 Sample SCI Initialization Flowchart**

### 14.4.5 Data Transmission (Asynchronous Mode)

Figure 14.6 shows an example of operation for transmission in asynchronous mode. In transmission, the SCI operates as described below.

1. The SCI monitors the TDRE flag in SSR. If the flag is cleared to 0, the SCI 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 SCI sets the TDRE flag to 1 and starts transmission. If the TIE bit is set to 1 at this time, a transmit data empty interrupt request (TXI) 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. Data is sent from the TxD pin in the following order: start bit, transmit data, parity bit or multiprocessor bit (may be omitted depending on the format), and stop bit.
4. The SCI checks the TDRE flag at the timing for sending the stop bit.
5. 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.
6. 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 SCR is set to 1 at this time, a TEI interrupt request is generated.

Figure 14.7 shows a sample flowchart for transmission in asynchronous mode.

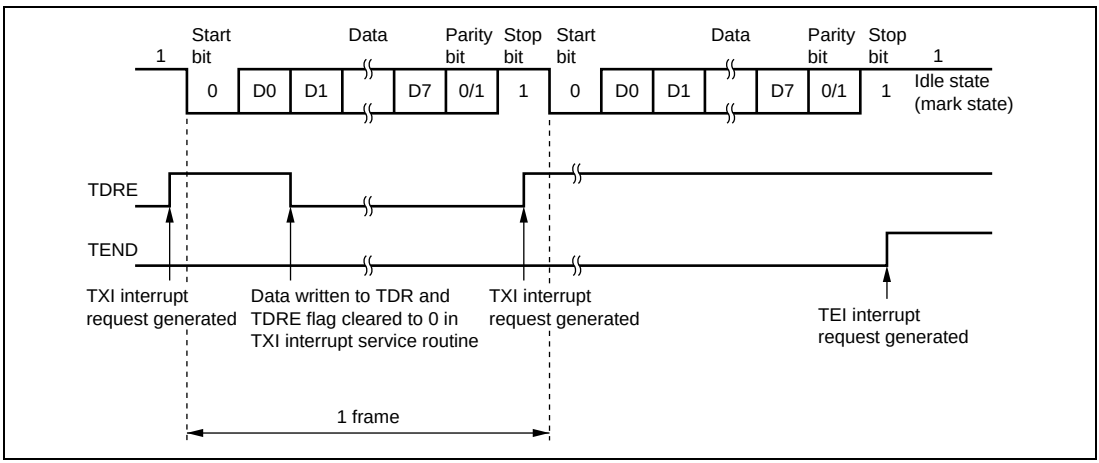
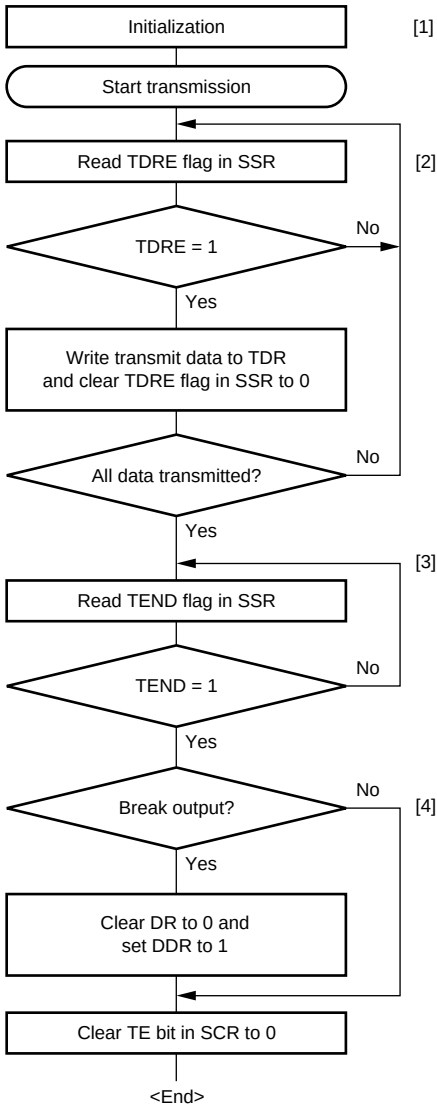



**Figure 14.6 Example of Operation in Transmission in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit)**




[1] SCI initialization:
The TxD pin is automatically designated as the transmit data output pin.
After the TE bit is set to 1, a frame of 1s is output, and transmission is enabled.

[2] SCI status check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR and clear the TDRE flag to 0.

[3] Serial transmission continuation procedure:
To continue serial transmission, read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR, and then clear the TDRE flag to 0. Checking and clearing of the TDRE flag is automatic when the DTC is activated by a transmit data empty interrupt (TXI) request, and data is written to TDR.

[4] Break output at the end of serial transmission:
To output a break in serial transmission, set DDR for the port corresponding to the TxD pin to 1, clear DR to 0, then clear the TE bit in SCR to 0.

**Figure 14.7 Sample Serial Transmission Flowchart**

### 14.4.6 Serial Data Reception (Asynchronous Mode)

Figure 14.8 shows an example of operation for reception in asynchronous mode. In serial reception, the SCI operates as described below.

1. The SCI monitors the communication line. If a start bit is detected, the SCI performs internal synchronization, receives receive 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 ORER bit in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR. The RDRF flag remains to be set to 1.
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 SCR 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 SCR 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 SCR 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.

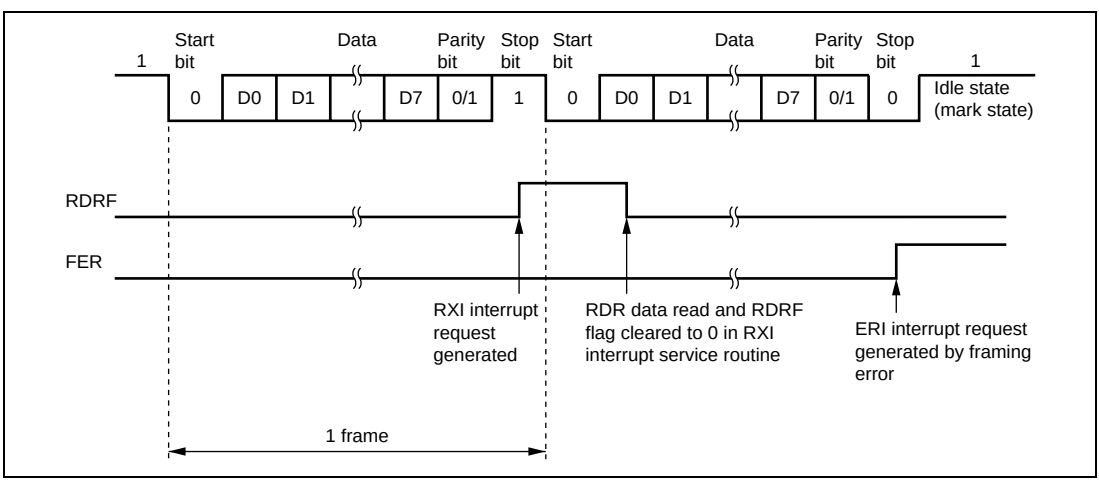


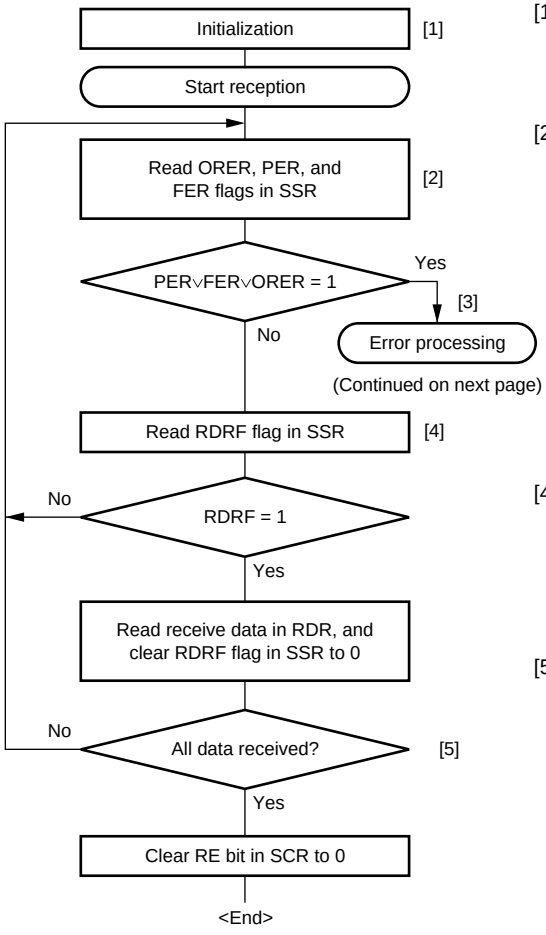

**Figure 14.8 Example of SCI Operation in Reception (Example with 8-Bit Data, Parity, One Stop Bit)**

Table 14.11 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 ORER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 14.9 shows a sample flow chart for serial data reception.

**Table 14.11 SSR Status Flags and Receive Data Handling**

| SSR Status Flag | | | | | |
|---|---|---|---|---|---|
| **RDRF*** | **ORER** | **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] SCI initialization:
The RxD pin is automatically designated as the receive data input pin.

[2] [3] Receive error processing and break detection:
If a receive error occurs, read the ORER, PER, and FER flags in SSR to identify the error. After performing the appropriate error processing, ensure that the ORER, 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.

[4] SCI status check and receive data read:
Read SSR and check that RDRF = 1, then read the receive data in RDR and clear the RDRF flag to 0. Transition of the RDRF flag from 0 to 1 can also be identified by an RXI interrupt.

[5] Serial reception continuation procedure:
To continue serial reception, before the stop bit for the current frame is received, read the RDRF flag, read RDR, and clear the RDRF flag to 0. The RDRF flag is cleared automatically when DTC is activated by an RXI interrupt and the RDR value is read.

**Figure 14.9 Sample Serial Reception Data Flowchart (1)**

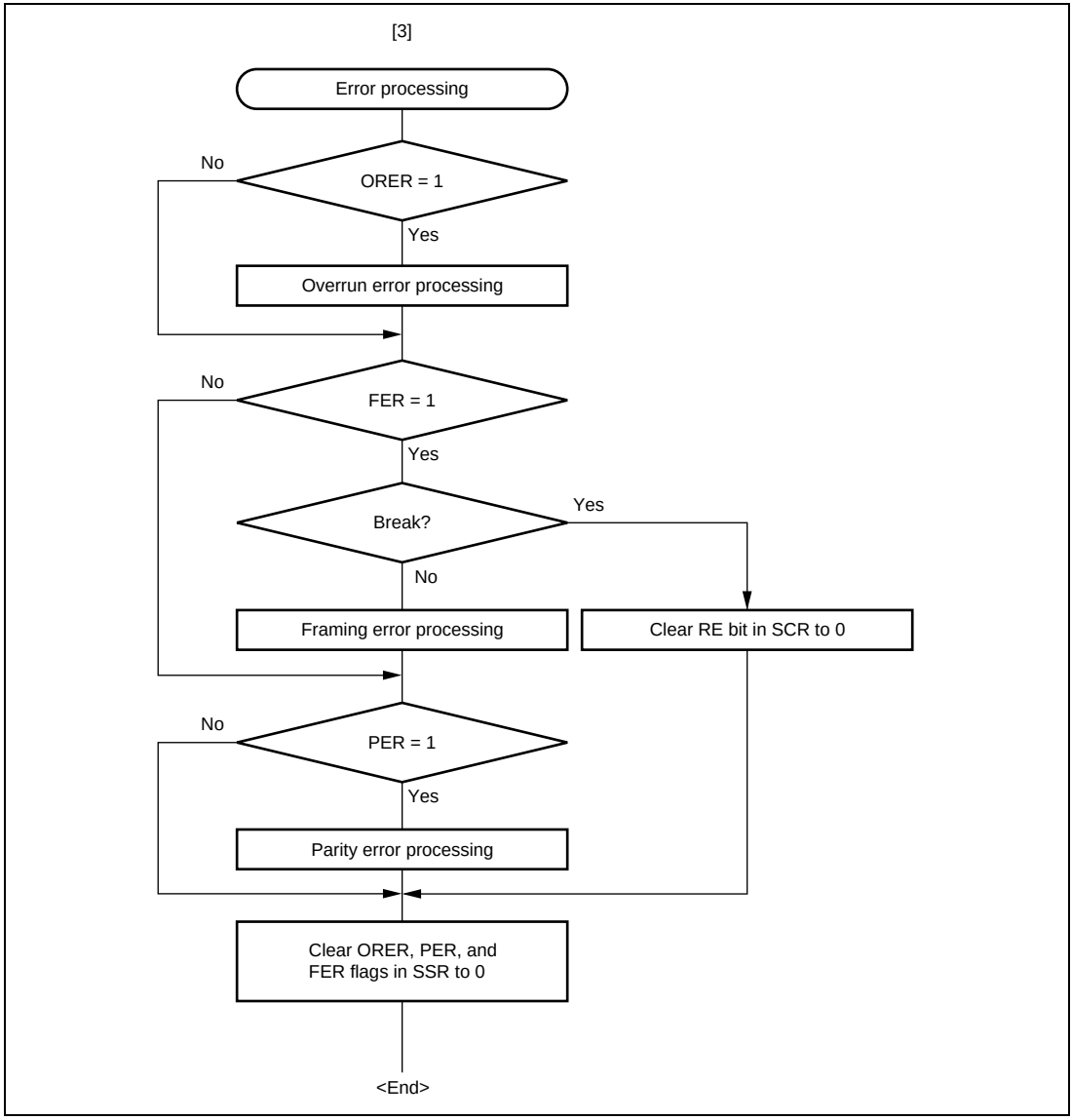



**Figure 14.9 Sample Serial Reception Data Flowchart (2)**

## 14.5 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 14.10 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 SCI uses the MPIE bit in SCR 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 ORER 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 MPB bit in SSR is set to 1 and the MPIE bit is automatically cleared, thus normal reception is resumed. If the RIE bit in SCR 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.

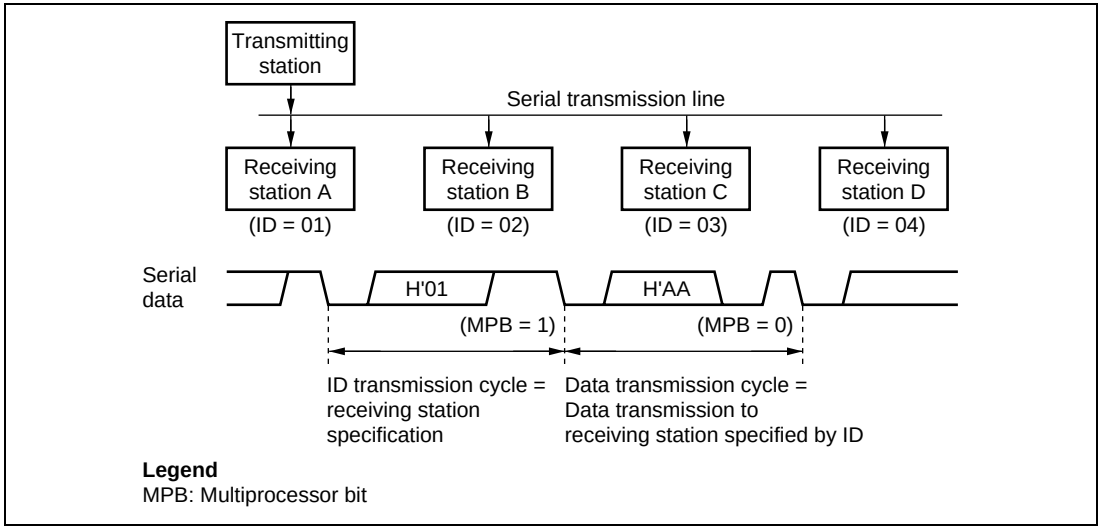
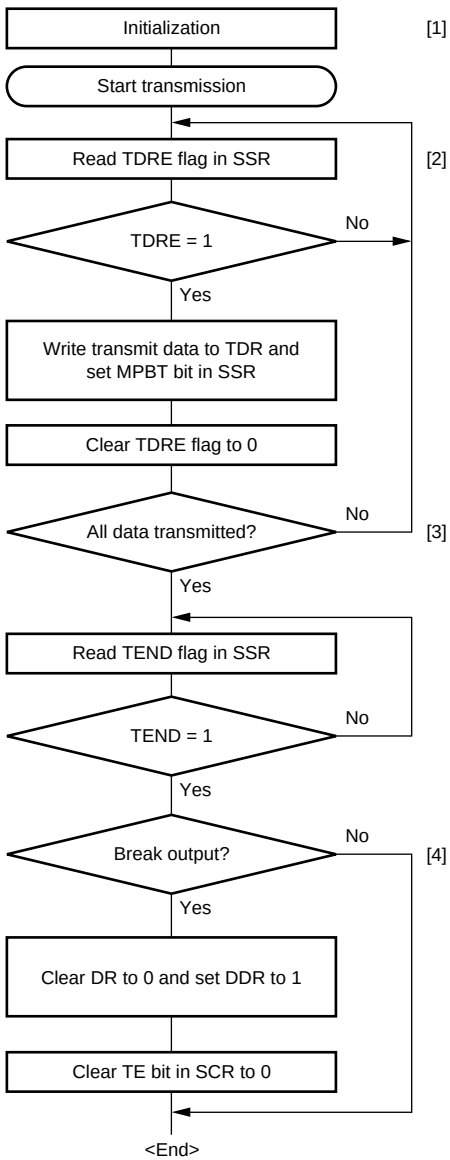



**Figure 14.10 Example of Communication Using Multiprocessor Format (Transmission of Data H′AA to Receiving Station A)**

### 14.5.1 Multiprocessor Serial Data Transmission

Figure 14.11 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 SCI operations are the same as those in asynchronous mode.




[1] SCI initialization:
The TxD pin is automatically designated as the transmit data output pin.
After the TE bit is set to 1, a frame of 1s is output, and transmission is enabled.

[2] SCI status check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR. Set the MPBT bit in SSR to 0 or 1. Finally, clear the TDRE flag to 0.

[3] Serial transmission continuation procedure:
To continue serial transmission, be sure to read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR, and then clear the TDRE flag to 0. Checking and clearing of the TDRE flag is automatic when the DTC is activated by a transmit data empty interrupt (TXI) request, and data is written to TDR.

[4] Break output at the end of serial transmission:
To output a break in serial transmission, set the port DDR to 1, clear DR to 0, then clear the TE bit in SCR to 0.

**Figure 14.11 Sample Multiprocessor Serial Transmission Flowchart**

### 14.5.2 Multiprocessor Serial Data Reception

Figure 14.13 shows a sample flowchart for multiprocessor serial data reception. If the MPIE bit in SCR is set to 1, data is skipped until data with a 1 multiprocessor bit is sent. 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 SCI operations are the same as in asynchronous mode. Figure 14.12 shows an example of SCI operation for multiprocessor format reception.

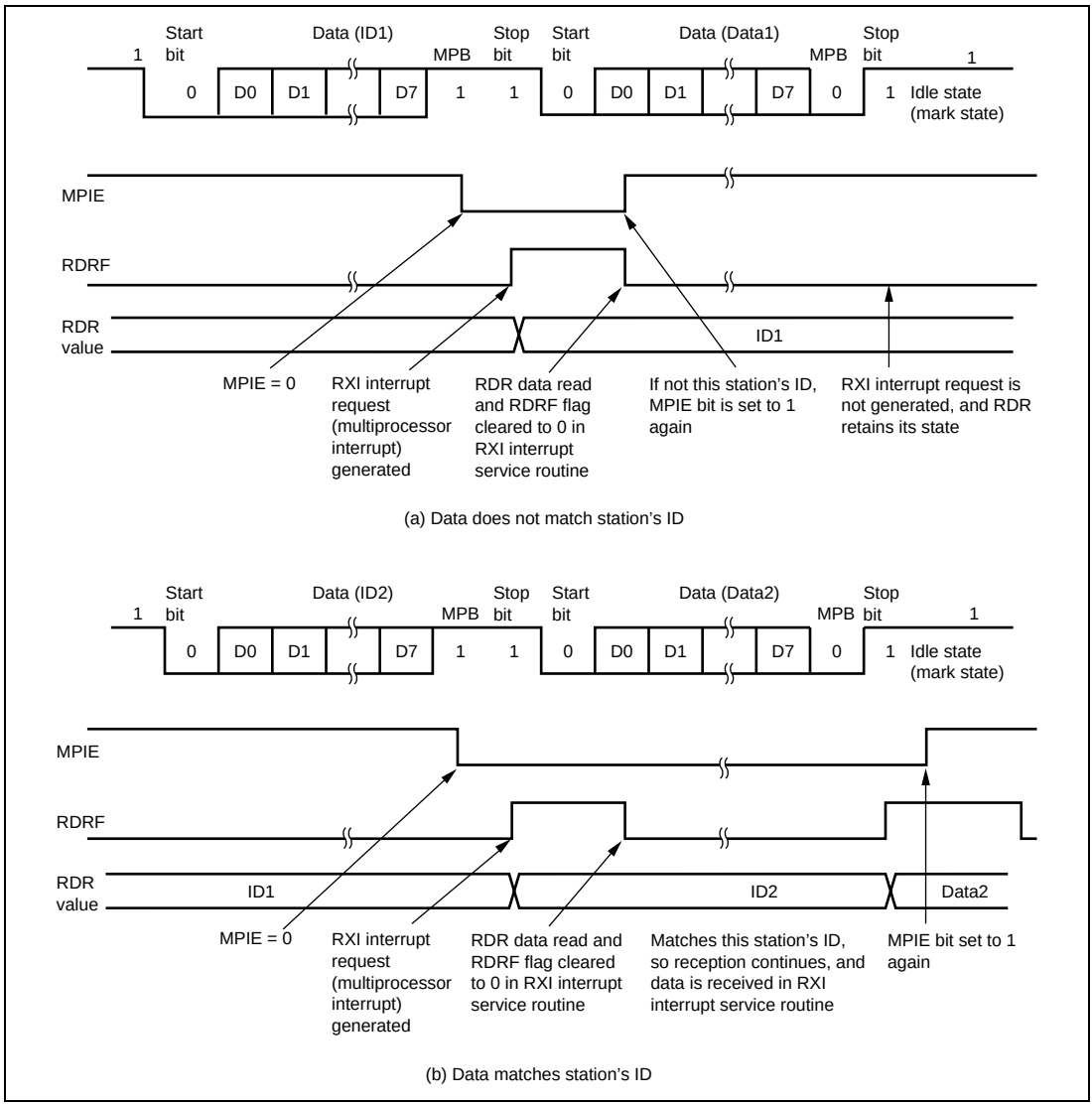
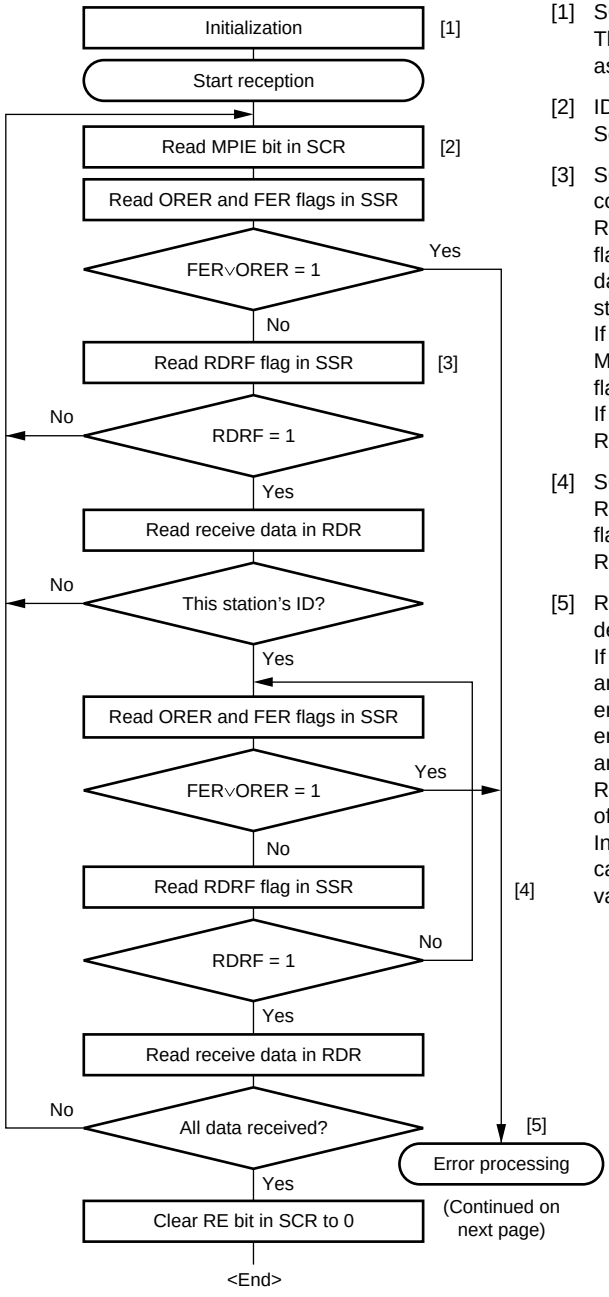



**Figure 14.12 Example of SCI Operation in Reception (Example with 8-Bit Data, Multiprocessor Bit, One Stop Bit)**




[1] SCI initialization:
The RxD pin is automatically designated as the receive data input pin.

[2] ID reception cycle:
Set the MPIE bit in SCR to 1.

[3] SCI status check, ID reception and comparison:
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, and clear the RDRF flag to 0.
If the data is this station's ID, clear the RDRF flag to 0.

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

[5] Receive error processing and break detection:
If a receive error occurs, read the ORER and FER flags in SSR to identify the error. After performing the appropriate error processing, ensure that the ORER 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 14.13 Sample Multiprocessor Serial Reception Flowchart (1)**

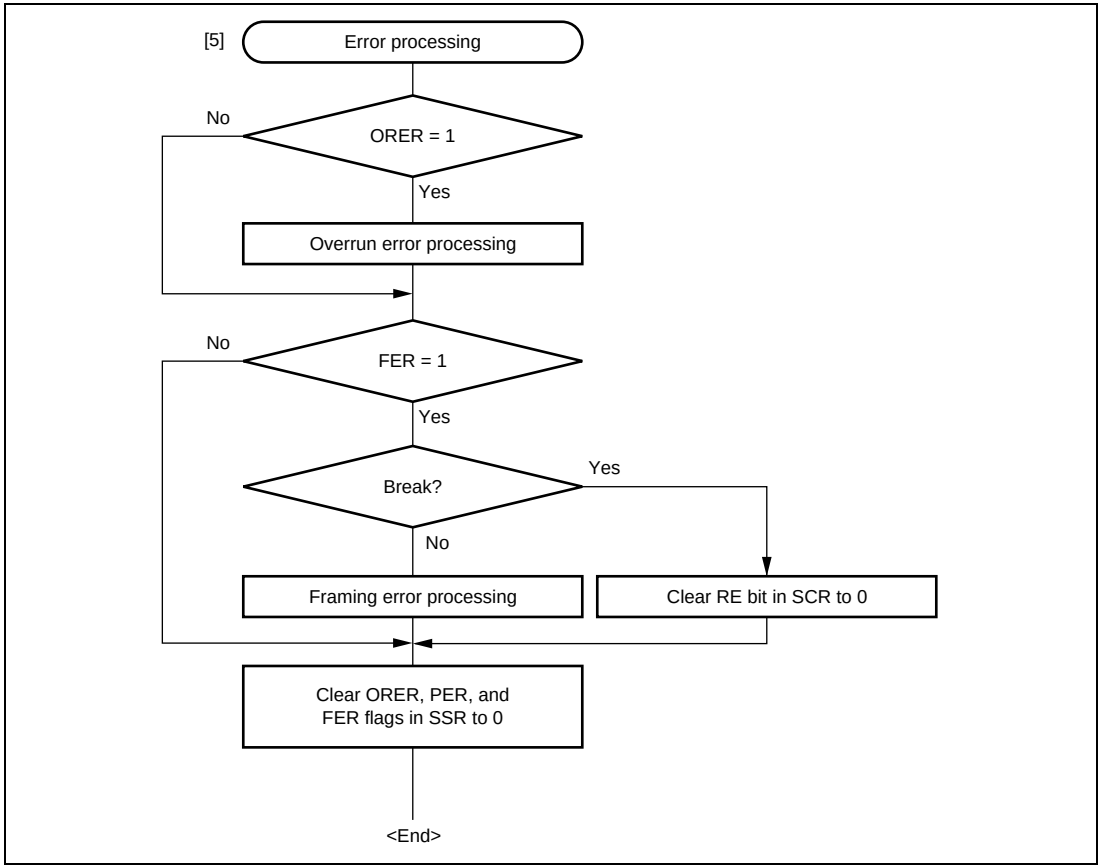



**Figure 14.13 Sample Multiprocessor Serial Reception Flowchart (2)**

## 14.6 Operation in Clocked Synchronous Mode

Figure 14.14 shows the general format for clocked synchronous communication. In clocked synchronous mode, data is transmitted or received synchronous with clock pulses. 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 SCI 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 SCI, 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.

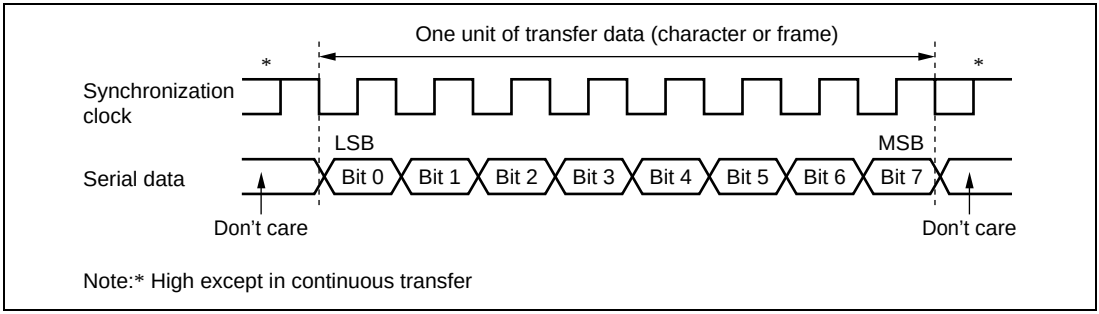


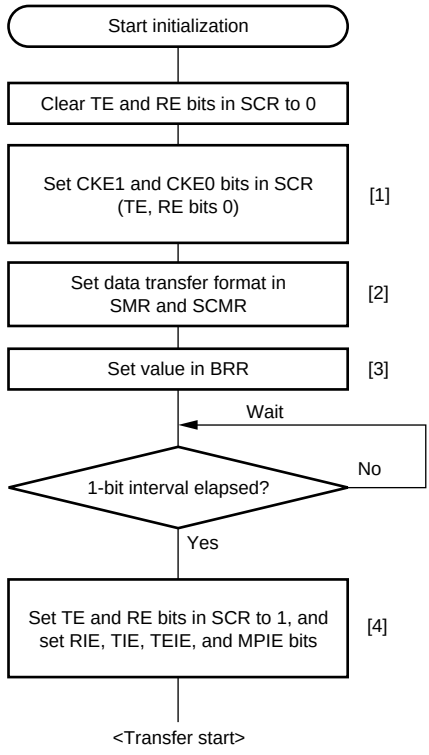

**Figure 14.14 Data Format in Synchronous Communication (For LSB-First)**

### 14.6.1 Clock

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

### 14.6.2 SCI Initialization (Clocked Synchronous Mode)

Before transmitting and receiving data, the TE and RE bits in SCR should be cleared to 0, then the SCI should be initialized as described in a sample flowchart in figure 14.15. When the operating mode, or transfer format, is changed for example, the TE and RE bits must be cleared to 0 before making the change using the following procedure. 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 change the contents of the RDRF, PER, FER, and ORER flags, or the contents of RDR.




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

[2] Set the data transfer format in SMR and SCMR.

[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 SCR to 1. Also set the RIE, TIE TEIE, and MPIE bits. Setting the TE and RE bits enables the TxD and RxD pins to be used.

Note:* In simultaneous transmit and receive operations, the TE and RE bits should both be cleared to 0 or set to 1 simultaneously.

**Figure 14.15 Sample SCI Initialization Flowchart**

### 14.6.3 Serial Data Transmission (Clocked Synchronous Mode)

Figure 14.16 shows an example of SCI operation for transmission in clocked synchronous mode. In serial transmission, the SCI operates as described below.

1. The SCI 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. After transferring data from TDR to TSR, the SCI sets the TDRE flag to 1 and starts transmission. If the TIE bit in SCR is set to 1 at this time, a transmit data empty interrupt (TXI) is generated. Continuous transmission is possible because the TXI interrupt routine writes the next transmit data to TDR before transmission of the current transmit data has been completed.
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.
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 SCR is set to 1 at this time, a TEI interrupt request is generated. The SCK pin is fixed high.

Figure 14.17 shows a sample flow chart for serial data transmission. Even if the TDRE flag is cleared to 0, transmission will not start while a receive error flag (ORER, FER, or PER) is set to 1. Make sure that the receive error flags are cleared to 0 before starting transmission. Note that clearing the RE bit to 0 does not clear the receive error flags.

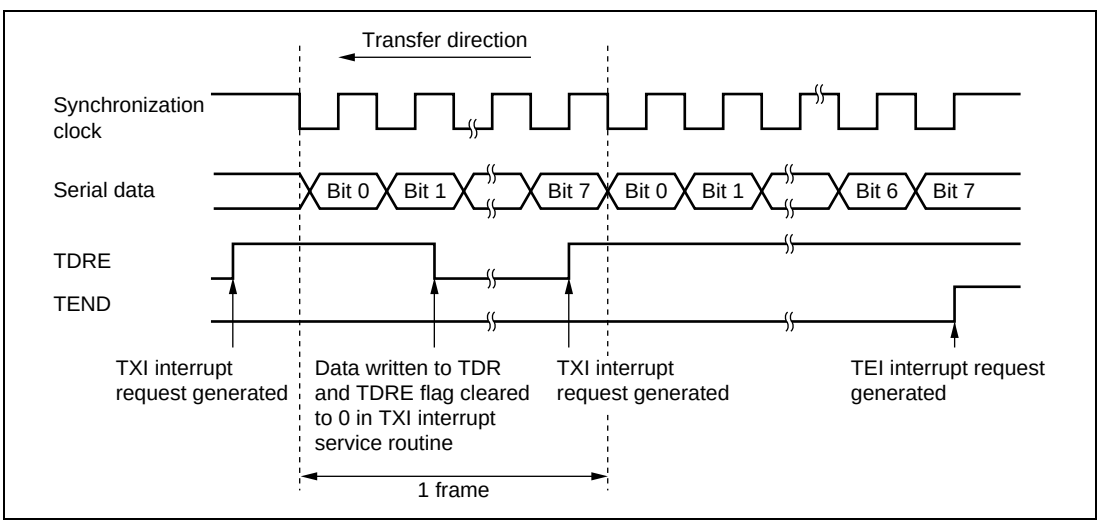
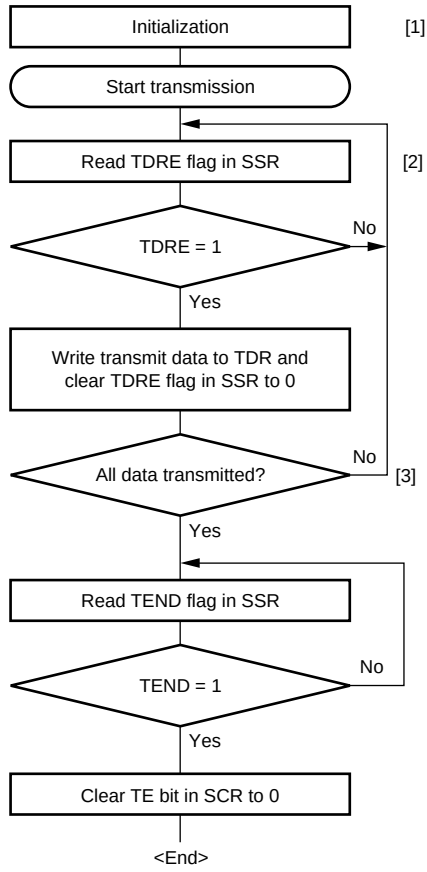



**Figure 14.16 Sample SCI Transmission Operation in Clocked Synchronous Mode**




[1] SCI initialization:
The TxD pin is automatically designated as the transmit data output pin.

[2] SCI status check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR and clear the TDRE flag to 0.

[3] Serial transmission continuation procedure:
To continue serial transmission, be sure to read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR, and then clear the TDRE flag to 0.
Checking and clearing of the TDRE flag is automatic when the DTC is activated by a transmit data empty interrupt (TXI) request and data is written to TDR.

**Figure 14.17 Sample Serial Transmission Flowchart**

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

Figure 14.18 shows an example of SCI operation for reception in clocked synchronous mode. In serial reception, the SCI operates as described below.

1. The SCI performs internal initialization synchronous with a synchronous clock input or output, starts receiving data, and stores the received data in RSR.
2. 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 ORER bit in SSR is set to 1. If the RIE bit in SCR 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.
3. 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 SCR 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 finished.

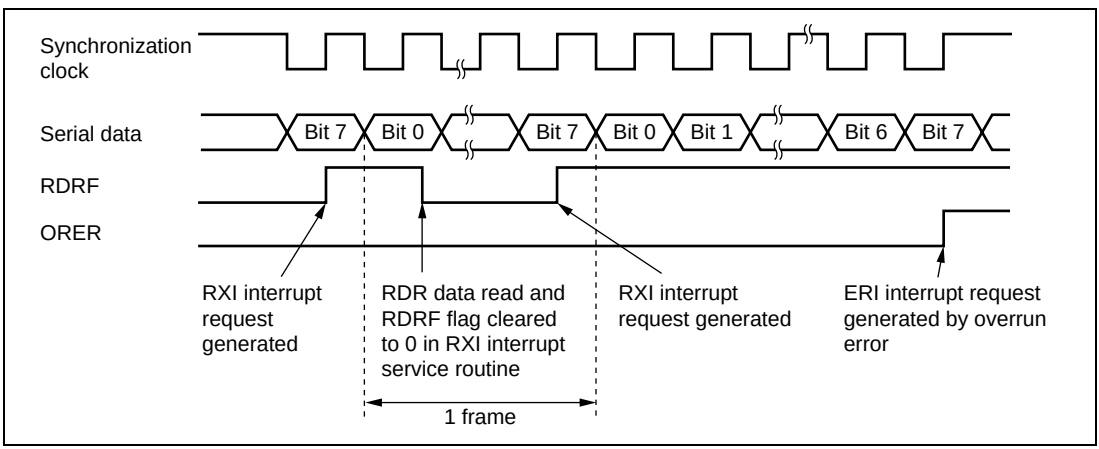
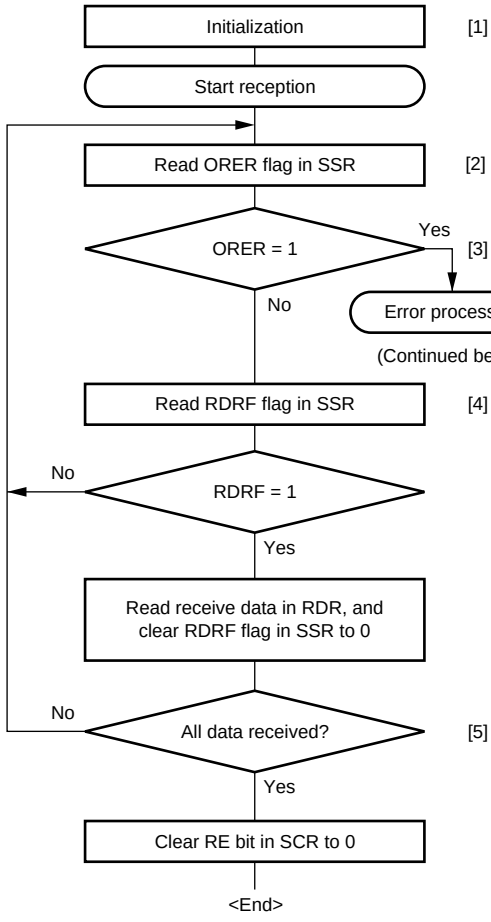
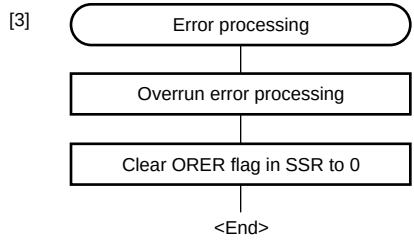



**Figure 14.18 Example of SCI Operation in Reception**

Reception cannot be resumed while a receive error flag is set to 1. Accordingly, clear the ORER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 14.19 shows a sample flow chart for serial data reception.




[1] SCI initialization:
The RxD pin is automatically designated as the receive data input pin.

[2] [3] Receive error processing:
If a receive error occurs, read the ORER flag in SSR, and after performing the appropriate error processing, clear the ORER flag to 0. Transfer cannot be resumed if the ORER flag is set to 1.

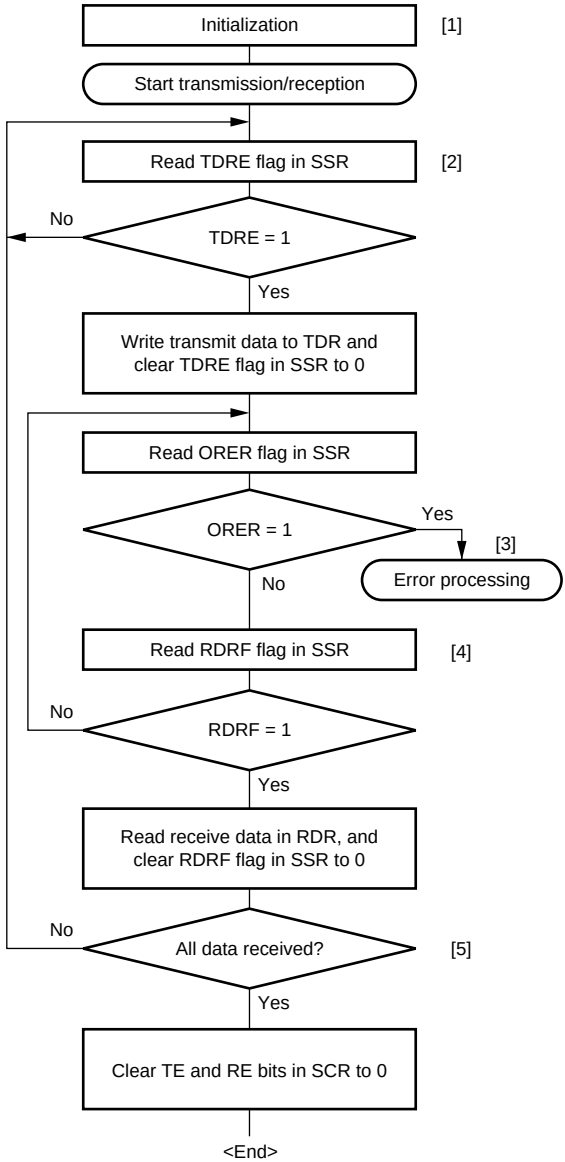
[4] SCI status check and receive data read:
Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR and clear the RDRF flag to 0.
Transition of the RDRF flag from 0 to 1 can also be identified by an RXI interrupt.

[5] Serial reception continuation procedure:
To continue serial reception, before the MSB (bit 7) of the current frame is received, reading the RDRF flag, reading RDR, and clearing the RDRF flag to 0 should be finished. The RDRF flag is cleared automatically when the DTC is activated by a receive data full interrupt (RXI) request and the RDR value is read.

**Figure 14.19 Sample Serial Reception Flowchart**

### 14.6.5 Simultaneous Serial Data Transmission and Reception (Clocked Synchronous Mode)

Figure 14.20 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 after initializing the SCI. To switch from transmit mode to simultaneous transmit and receive mode, after checking that the SCI 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 SCI has finished reception, clear RE to 0. Then after checking that the RDRF and receive error flags (ORER, FER, and PER) are cleared to 0, simultaneously set TE and RE to 1 with a single instruction.




Note:* When switching from transmit or receive operation to simultaneous transmit and receive operations, first clear the TE bit and RE bit to 0, then set both these bits to 1 simultaneously.

[1] SCI initialization:
The TxD pin is designated as the transmit data output pin, and the RxD pin is designated as the receive data input pin, enabling simultaneous transmit and receive operations.

[2] SCI status check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR and clear the TDRE flag to 0.
Transition of the TDRE flag from 0 to 1 can also be identified by a TXI interrupt.

[3] Receive error processing:
If a receive error occurs, read the ORER flag in SSR, and after performing the appropriate error processing, clear the ORER flag to 0. Transmission/reception cannot be resumed if the ORER flag is set to 1.

[4] SCI status check and receive data read:
Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR and clear the RDRF flag to 0. Transition of the RDRF flag from 0 to 1 can also be identified by an RXI interrupt.

[5] Serial transmission/reception continuation procedure:
To continue serial transmission/reception, before the MSB (bit 7) of the current frame is received, finish reading the RDRF flag, reading RDR, and clearing the RDRF flag to 0. 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 and clear the TDRE flag to 0. Checking and clearing of the TDRE flag is automatic when the DTC is activated by a transmit data empty interrupt (TXI) request and data is written to TDR. Also, the RDRF flag is cleared automatically when the DTC is activated by a receive data full interrupt (RXI) request and the RDR value is read.

**Figure 14.20 Sample Flowchart of Simultaneous Serial Transmit and Receive Operations**

## 14.7 Operation in Smart Card Interface

The SCI supports an IC card (Smart Card) interface that conforms to ISO/IEC 7816-3 (Identification Card) as a serial communication interface extension function. Switching between the normal serial communication interface and the Smart Card interface mode is carried out by means of a register setting.

### 14.7.1 Pin Connection Example

Figure 14.21 shows an example of connection with the Smart Card. In communication with an IC card, as both transmission and reception are carried out on a single data transmission line, the TxD pin and RxD pin should be connected to the LSI pin. The data transmission line should be pulled up to the $V_{CC}$ power supply with a resistor. If an IC card is not connected, and the TE and RE bits are both set to 1, closed transmission/reception is possible, enabling self-diagnosis to be carried out. When the clock generated on the Smart Card interface is used by an IC card, the SCK pin output is input to the CLK pin of the IC card. This LSI port output is used as the reset signal.

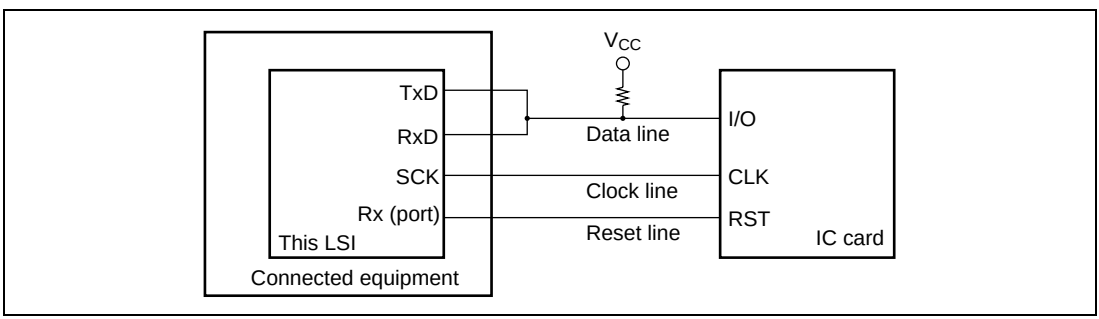



**Figure 14.21 Schematic Diagram of Smart Card Interface Pin Connections**

### 14.7.2 Data Format (Except for Block Transfer Mode)

Figure 14.22 shows the transfer data format in Smart Card interface mode.

- One frame consists of 8-bit data plus a parity bit in asynchronous mode.
- In transmission, a guard time of at least 2 etu (Elementary Time Unit: the time for transfer of one bit) is left between the end of the parity bit and the start of the next frame.
- If a parity error is detected during reception, a low error signal level is output for one etu period, 10.5 etu after the start bit.
- If an error signal is sampled during transmission, the same data is retransmitted automatically after a delay of 2 etu or longer.

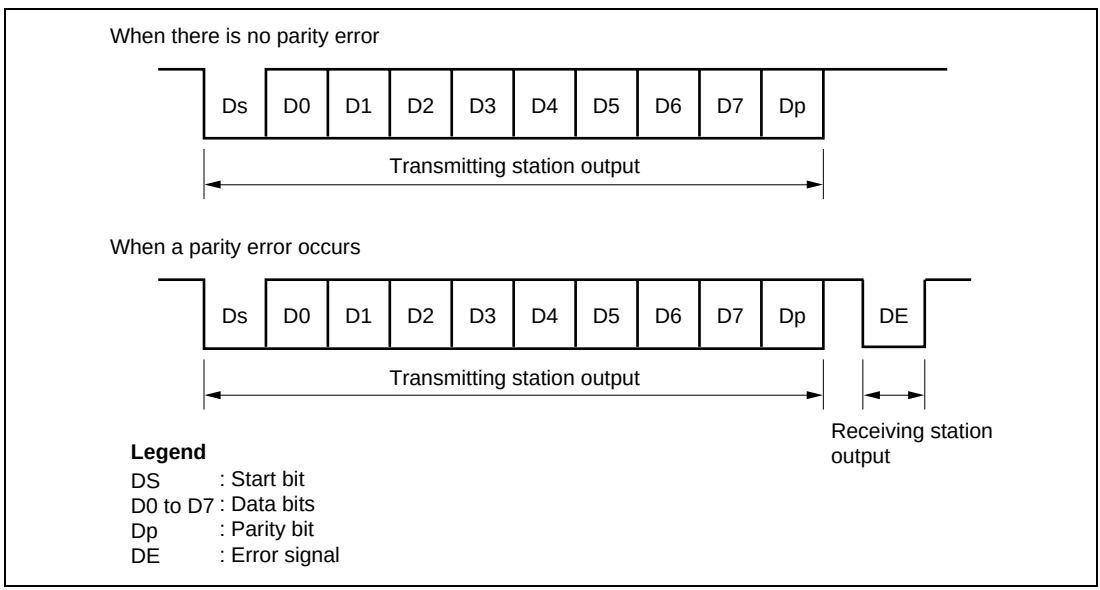



**Figure 14.22 Normal Smart Card Interface Data Format**

Data transfer with other types of IC cards (direct convention and inverse convention) are performed as described in the following.

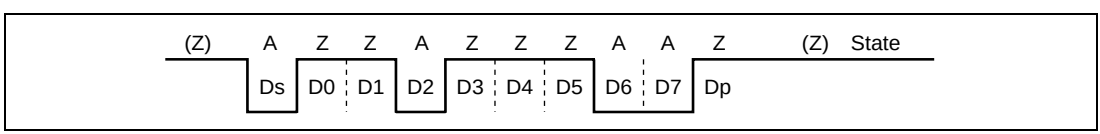



**Figure 14.23 Direct Convention (SDIR = SINV = O/$\overline{E}$ = 0)**

With the direction convention type IC and the above sample start character, the logic 1 level corresponds to state Z and the logic 0 level to state A, and transfer is performed in LSB-first order. The start character data above is H′3B. For the direct convention type, clear the SDIR and SINV bits in SCMR to 0. According to Smart Card regulations, clear the O/$\overline{E}$ bit in SMR to 0 to select even parity mode.

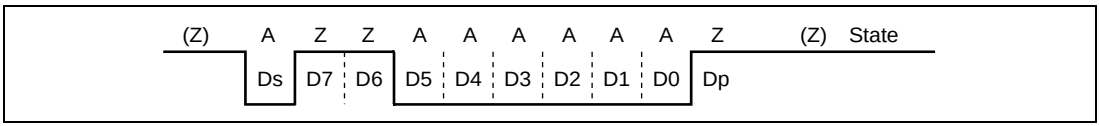



**Figure 14.24 Inverse Convention (SDIR = SINV = O/$\overline{E}$ = 1)**

With the inverse convention type, the logic 1 level corresponds to state A and the logic 0 level to state Z, and transfer is performed in MSB-first order. The start character data for the above is H′3F. For the inverse convention type, set the SDIR and SINV bits in SCMR to 1. According to Smart Card regulations, even parity mode is the logic 0 level of the parity bit, and corresponds to state Z. In this LSI, the SINV bit inverts only data bits D0 to D7. Therefore, set the O/$\overline{E}$ bit in SMR to 1 to invert the parity bit for both transmission and reception.

### 14.7.3 Block Transfer Mode

Operation in block transfer mode is the same as that in SCI asynchronous mode, except for the following points.

- In reception, though the parity check is performed, no error signal is output even if an error is detected. However, the PER bit in SSR is set to 1 and must be cleared before receiving the parity bit of the next frame.
- In transmission, a guard time of at least 1 etu is left between the end of the parity bit and the start of the next frame.
- In transmission, because retransmission is not performed, the TEND flag is set to 1, 11.5 etu after transmission start.
- As with the normal Smart Card interface, the ERS flag indicates the error signal status, but since error signal transfer is not performed, this flag is always cleared to 0.

### 14.7.4 Receive Data Sampling Timing and Reception Margin in Smart Card Interface Mode

In Smart Card interface mode, the SCI operates on a basic clock with a frequency of 32, 64, 372, or 256 times the transfer rate (fixed at 16 times in normal asynchronous mode) as determined by bits BCP1 and BCP0. In reception, the SCI samples the falling edge of the start bit using the basic clock, and performs internal synchronization. As shown in figure 14.25, by sampling receive data at the rising-edge of the 16th, 32nd, 186th, or 128th pulse of the basic clock, data can be latched at the middle of the bit. The reception margin is given by the following formula.

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

Where M: Reception margin (%)
N: Ratio of bit rate to clock (N = 32, 64, 372, and 256)
D: Clock duty cycle (D = 0 to 1.0)
L: Frame length (L = 10)
F: Absolute value of clock frequency deviation

Assuming values of F = 0, D = 0.5 and N = 372 in the above formula, the reception margin formula is as follows.

$$M = (0.5 - 1/2 \times 372) \times 100\%$$
$$= 49.866\%$$

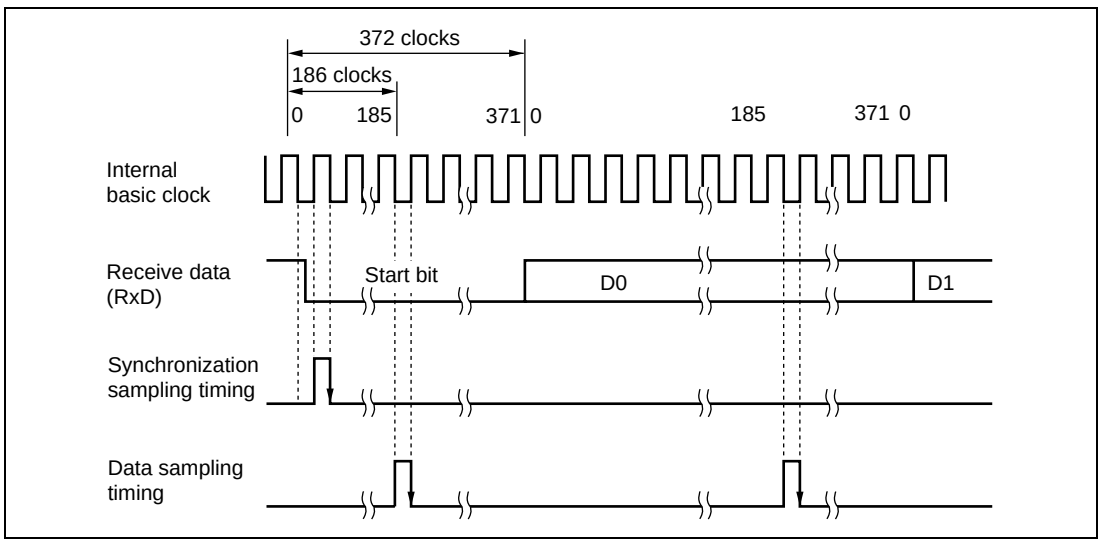



**Figure 14.25 Receive Data Sampling Timing in Smart Card Mode (Using Clock of 372 Times the Transfer Rate)**

### 14.7.5 Initialization

Before transmitting and receiving data, initialize the SCI as described below. Initialization is also necessary when switching from transmit mode to receive mode, or vice versa.

1. Clear the TE and RE bits in SCR to 0.
2. Clear the error flags ERS, PER, and ORER in SSR to 0.
3. Set the GM, BLK, O/$\overline{E}$, BCP0, BCP1, CKS0, CKS1 bits in SMR. Set the PE bit to 1.
4. Set the SMIF, SDIR, and SINV bits in SCMR.
   When the SMIF bit is set to 1, the TxD and RxD pins are both switched from ports to SCI pins, and are placed in the high-impedance state.
5. Set the value corresponding to the bit rate in BRR.
6. Set the CKE0 and CKE1 bits in SCR. Clear the TIE, RIE, TE, RE, MPIE, and TEIE bits to 0.
   If the CKE0 bit is set to 1, the clock is output from the SCK pin.
7. Wait at least one bit interval, then set the TIE, RIE, TE, and RE bits in SCR. Do not set the TE bit and RE bit at the same time, except for self-diagnosis.

To switch from receive mode to transmit mode, after checking that the SCI has finished reception, initialize the SCI, and set RE to 0 and TE to 1. Whether SCI has finished reception or not can be checked with the RDRF, PER, or ORER flags. To switch from transmit mode to receive mode, after checking that the SCI has finished transmission, initialize the SCI, and set TE to 0 and RE to 1. Whether SCI has finished transmission or not can be checked with the TEND flag.

### 14.7.6 Data Transmission (Except for Block Transfer Mode)

As data transmission in Smart Card interface mode involves error signal sampling and retransmission processing, the operations are different from those in normal serial communication interface mode (except for block transfer mode). Figure 14.26 illustrates the retransfer operation when the SCI is in transmit mode.

1. If an error signal is sent back from the receiving end after transmission of one frame is complete, the ERS bit in SSR is set to 1. If the RIE bit in SCR is enabled at this time, an ERI interrupt request is generated. The ERS bit in SSR should be kept cleared to 0 until the next parity bit is sampled.
2. The TEND bit in SSR is not set for a frame in which an error signal indicating an abnormality is received. Data is retransferred from TDR to TSR, and retransmitted automatically.
3. If an error signal is not sent back from the receiving end, the ERS bit in SSR is not set. Transmission of one frame, including a retransfer, is judged to have been completed, and the TEND bit in SSR is set to 1. If the TIE bit in SCR is enabled at this time, a TXI interrupt request is generated. Writing transmit data to TDR transfers the next transmit data.

Figure 14.28 shows a flowchart for transmission. The sequence of transmit operations can be performed automatically by specifying the DTC to be activated with a TXI interrupt source. In a transmit operation, the TDRE flag is set to 1 at the same time as the TEND flag in SSR is set, and a TXI interrupt will be generated if the TIE bit in SCR has been set to 1. If the TXI request is designated beforehand as a DTC activation source, the DTC will be activated by the TXI request, and transfer of the transmit data will be carried out. The TDRE and TEND flags are automatically cleared to 0 when data is transferred by the DTC. In the event of an error, the SCI retransmits the same data automatically. During this period, the TEND flag remains cleared to 0 and the DTC is not activated. Therefore, the SCI and DTC will automatically transmit the specified number of bytes in the event of an error, including retransmission. However, the ERS flag is not cleared automatically when an error occurs, and so the RIE bit should be set to 1 beforehand so that an ERI request will be generated in the event of an error, and the ERS flag will be cleared.

When performing transfer using the DTC, it is essential to set and enable the DTC before carrying out SCI setting. For details of the DTC setting procedures, refer to section 8, Data Transfer Controller (DTC).

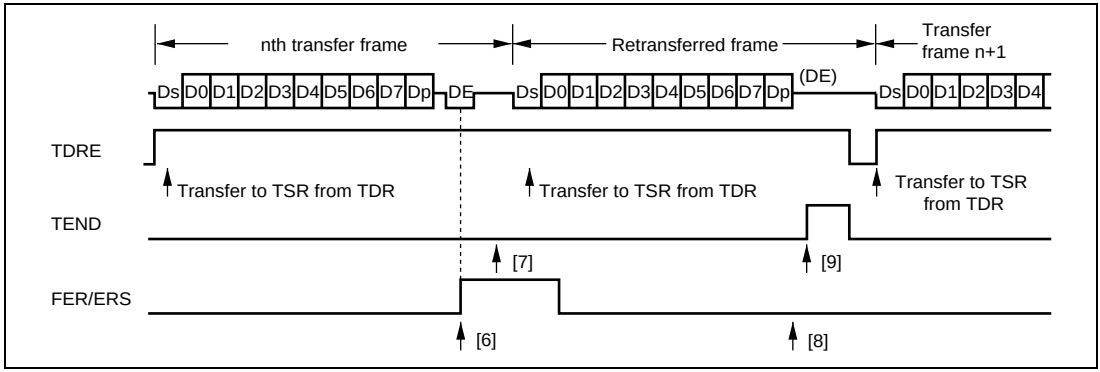



**Figure 14.26 Retransfer Operation in SCI Transmit Mode**

The timing for setting the TEND flag depends on the value of the GM bit in SMR. The TEND flag set timing is shown in figure 14.27.

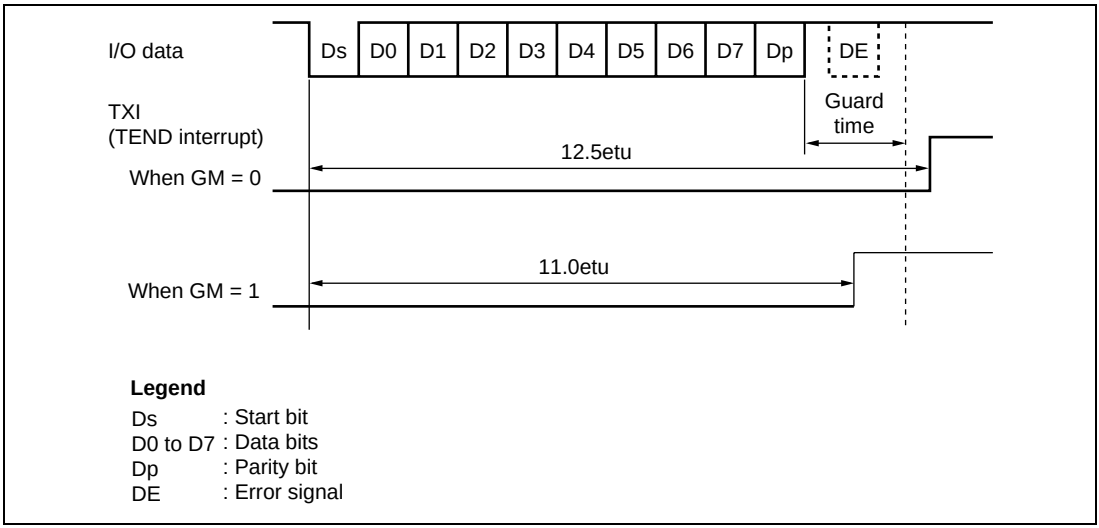



**Figure 14.27 TEND Flag Generation Timing in Transmission Operation**

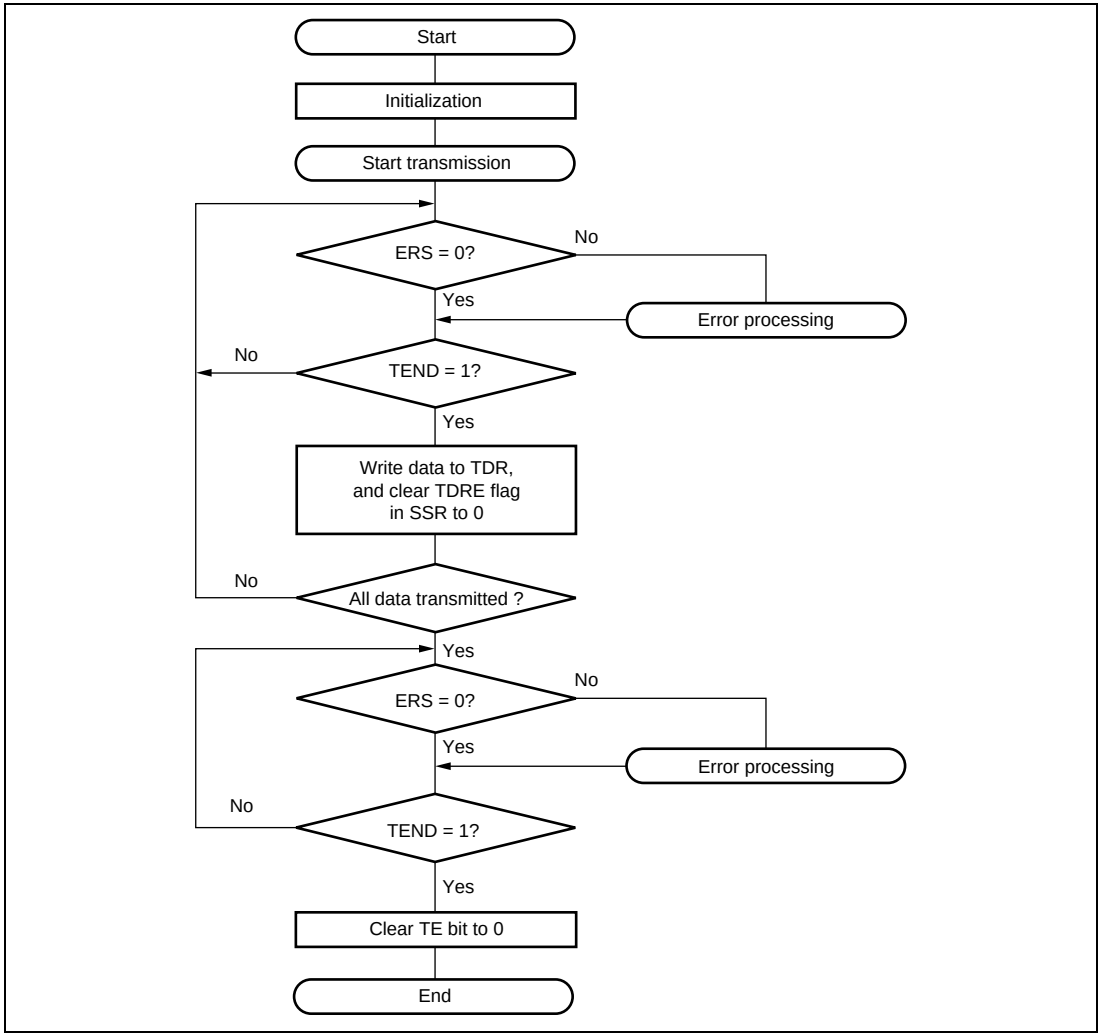



**Figure 14.28 Example of Transmission Processing Flow**

### 14.7.7 Serial Data Reception (Except for Block Transfer Mode)

Data reception in Smart Card interface mode uses the same operation procedure as for normal serial communication interface mode. Figure 14.29 illustrates the retransfer operation when the SCI is in receive mode.

1. If an error is found when the received parity bit is checked, the PER bit in SSR is automatically set to 1. If the RIE bit in SCR is set at this time, an ERI interrupt request is generated. The PER bit in SSR should be kept cleared to 0 until the next parity bit is sampled.
2. The RDRF bit in SSR is not set for a frame in which an error has occurred.
3. If no error is found when the received parity bit is checked, the PER bit in SSR is not set to 1, the receive operation is judged to have been completed normally, and the RDRF flag in SSR is automatically set to 1. If the RIE bit in SCR is enabled at this time, an RXI interrupt request is generated.

Figure 14.30 shows a flowchart for reception. A sequence of receive operations can be performed automatically by specifying the DTC to be activated using an RXI interrupt source. In a receive operation, an RXI interrupt request is generated when the RDRF flag in SSR is set to 1. If the RXI request is designated beforehand as a DTC activation source, the DTC will be activated by the RXI request, and the receive data will be transferred. The RDRF flag is cleared to 0 automatically when data is transferred by the DTC. If an error occurs in receive mode and the ORER or PER flag is set to 1, a transfer error interrupt (ERI) request will be generated. Hence, so the error flag must be cleared to 0. In the event of an error, the DTC is not activated and receive data is skipped. Therefore, receive data is transferred for only the specified number of bytes in the event of an error. Even when a parity error occurs in receive mode and the PER flag is set to 1, the data that has been received is transferred to RDR and can be read from there.

Note: For details on receive operations in block transfer mode, refer to section 14.4, Operation in Asynchronous Mode.

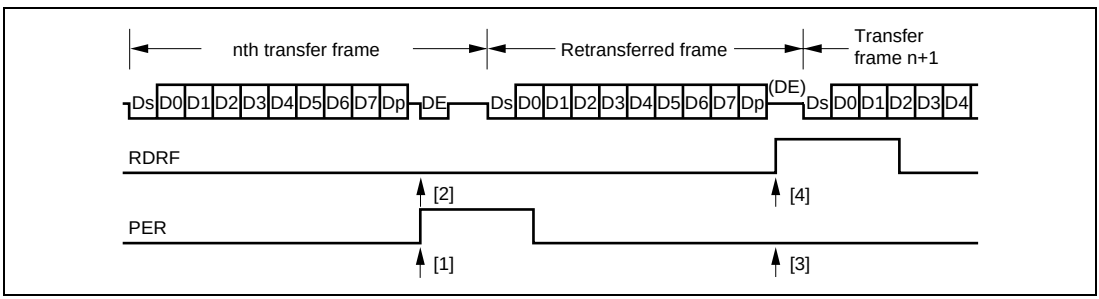



**Figure 14.29 Retransfer Operation in SCI Receive Mode**

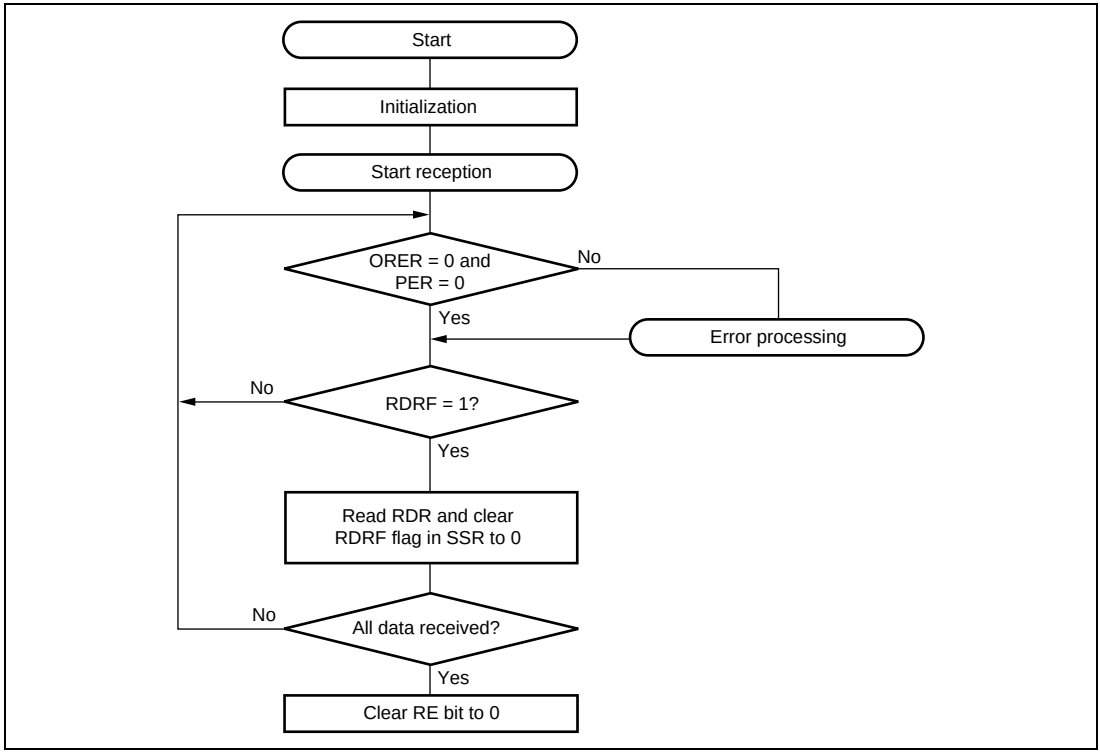



**Figure 14.30 Example of Reception Processing Flow**

### 14.7.8 Clock Output Control

When the GM bit in SMR is set to 1, the clock output level can be fixed with bits CKE0 and CKE1 in SCR. At this time, the minimum clock pulse width can be made the specified width. Figure 14.31 shows the timing for fixing the clock output level. In this example, GM is set to 1, CKE1 is cleared to 0, and the CKE0 bit is controlled.

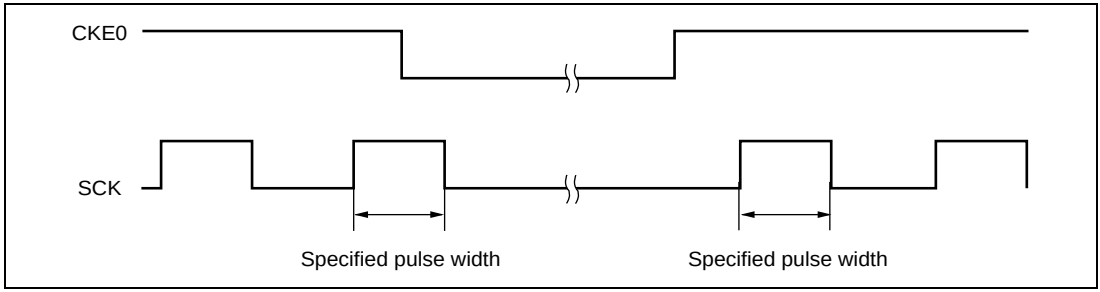



**Figure 14.31 Timing for Fixing Clock Output Level**

When turning on the power or switching between Smart Card interface mode and software standby mode, the following procedures should be followed in order to maintain the clock duty cycle.

**Powering On:** To secure clock duty cycle from power-on, the following switching procedure should be followed.

1. The initial state is port input and high impedance. Use a pull-up resistor or pull-down resistor to fix the potential.
2. Fix the SCK pin to the specified output level with the CKE1 bit in SCR.
3. Set SMR and SCMR, and switch to smart card mode operation.
4. Set the CKE0 bit in SCR to 1 to start clock output.

**When changing from smart card interface mode to software standby mode:**

1. Set the data register (DR) and data direction register (DDR) corresponding to the SCK pin to the value for the fixed output state in software standby mode.
2. Write 0 to the TE bit and RE bit in the serial control register (SCR) to halt transmit/receive operation. At the same time, set the CKE1 bit to the value for the fixed output state in software standby mode.
3. Write 0 to the CKE0 bit in SCR to halt the clock.
4. Wait for one serial clock period.

   During this interval, clock output is fixed at the specified level, with the duty cycle preserved.
5. Make the transition to the software standby state.

**When returning to smart card interface mode from software standby mode:**

1. Exit the software standby state.
2. Write 1 to the CKE0 bit in SCR and output the clock. Signal generation is started with the normal duty cycle.

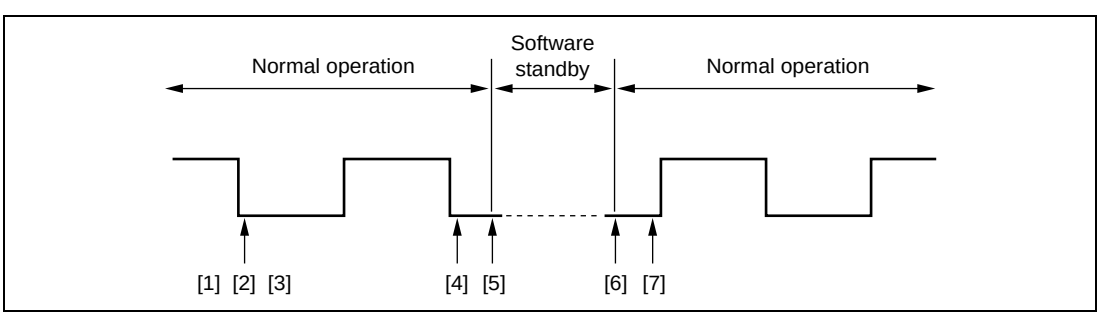



**Figure 14.32 Clock Halt and Restart Procedure**

## 14.8 Interrupt Sources

### 14.8.1 Interrupts in Normal Serial Communication Interface Mode

Table 14.12 shows the interrupt sources in normal serial communication interface mode. A different interrupt vector is assigned to each interrupt source, and individual interrupt sources can be enabled or disabled using the enable bits in SCR.

When the TDRE flag in SSR is set to 1, a TXI interrupt request is generated. When the TEND flag in SSR is set to 1, a TEI interrupt request is generated. A TXI interrupt can activate the DTC to perform data transfer. The TDRE flag is cleared to 0 automatically when data is transferred by the DTC.

When the RDRF flag in SSR is set to 1, an RXI interrupt request is generated. When the ORER, PER, or FER flag in SSR is set to 1, an ERI interrupt request is generated. An RXI interrupt request can activate the DTC to transfer data. The RDRF flag is cleared to 0 automatically when data is transferred by the DTC.

A TEI interrupt is requested when the TEND flag is set to 1 and the TEIE bit is set to 1. If a TEI interrupt and a TXI interrupt are requested simultaneously, the TXI interrupt has priority for acceptance. However, if the TDRE and TEND flags are cleared simultaneously by the TXI interrupt routine, the SCI cannot branch to the TEI interrupt routine later.

**Table 14.12 SCI Interrupt Sources**

| Channel | Name | Interrupt Source | Interrupt Flag | DTC Activation |
|---|---|---|---|---|
| 0 | ERI_0 | Receive Error | ORER, FER, PER | Not possible |
| | RXI_0 | Receive Data Full | RDRF | Possible |
| | TXI_0 | Transmit Data Empty | TDRE | Possible |
| | TEI_0 | Transmission End | TEND | Not possible |
| 2 | ERI_2 | Receive Error | ORER, FER, PER | Not possible |
| | RXI_2 | Receive Data Full | RDRF | Possible |
| | TXI_2 | Transmit Data Empty | TDRE | Possible |
| | TEI_2 | Transmission End | TEND | Not possible |

### 14.8.2 Interrupts in Smart Card Interface Mode

Table 14.13 shows the interrupt sources in Smart Card interface mode. The transmit end interrupt (TEI) request cannot be used in this mode.

**Table 14.13 SCI Interrupt Sources**

| Channel | Name | Interrupt Source | Interrupt Flag | DTC Activation |
|---|---|---|---|---|
| 0 | ERI_0 | Receive Error, error signal detection | ORER, PER, ERS | Not possible |
| | RXI_0 | Receive Data Full | RDRF | Possible |
| | TXI_0 | Transmit Data Empty | TEND | Possible |
| 2 | ERI_2 | Receive Error, error signal detection | ORER, PER, ERS | Not possible |
| | RXI_2 | Receive Data Full | RDRF | Possible |
| | TXI_2 | Transmit Data Empty | TEND | Possible |

In Smart Card interface mode, as in normal serial communication interface mode, transfer can be carried out using the DTC. In transmit operations, the TDRE flag is also set to 1 at the same time as the TEND flag in SSR is set, and a TXI interrupt is generated. If the TXI request is designated beforehand as a DTC activation source, the DTC will be activated by the TXI request, and transmit data will be transferred. The TDRE and TEND flags are automatically cleared to 0 when data is transferred by the DTC. In the event of an error, the SCI retransmits the same data automatically. During this period, the TEND flag remains cleared to 0 and the DTC is not activated. Therefore, the SCI and DTC will automatically transmit the specified number of bytes in the event of an error, including retransmission. However, the ERS flag is not cleared automatically when an error occurs. Hence, the RIE bit should be set to 1 beforehand so that an ERI request will be generated in the event of an error, and the ERS flag will be cleared.

When transferring using the DTC, it is essential to set and enable the DTC before carrying out SCI setting. For details of the DTC setting procedures, refer to section 8, Data Transfer Controller (DTC).

In receive operations, an RXI interrupt request is generated when the RDRF flag in SSR is set to 1. If the RXI request is designated beforehand as a DTC activation source, the DTC will be activated by the RXI request, and the receive data will be transferred. The RDRF flag is cleared to 0 automatically when data is transferred by the DTC. If an error occurs, an error flag is set but the RDRF flag is not. Consequently, the DTC is not activated, instead, an ERI interrupt request is sent to the CPU. Therefore, the error flag should be cleared.

## 14.9 Usage Notes

### 14.9.1 Module Stop Mode Setting

SCI operation can be disabled or enabled using the module stop control register. The initial setting is for SCI operation to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 14.9.2 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 0s, setting the FER flag, and possibly the PER flag. Note that as the SCI continues the receive operation after receiving a break, even if the FER flag is cleared to 0, it will be set to 1 again.

### 14.9.3 Mark State and Break Detection

When TE is 0, the TxD pin is used as an I/O port whose direction (input or output) and level are determined by DR and DDR. 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 DDR and DR 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 DDR to 1 and DR 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.

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

Transmission cannot be started when a receive error flag (ORER, 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.

# Section 15 Hitachi Controller Area Network (HCAN)

The HCAN is a module for controlling a controller area network (CAN) for realtime communication in vehicular and industrial equipment systems, etc. For details on CAN specification, refer to Bosch CAN Specification Version 2.0 1991, Robert Bosch GmbH.

The block diagram of the HCAN is shown in figure 15.1.

## 15.1 Features

- CAN version: Bosch 2.0B active compatible
  - Communication systems: NRZ (Non-Return to Zero) system (with bit-stuffing function)
  - Broadcast communication system
  - Transmission path: Bidirectional 2-wire serial communication
  - Communication speed: Max. 1 Mbps
  - Data length: 0 to 8 bytes
- Number of channels: 1
- Data buffers: 16 (one receive-only buffer and 15 buffers settable for transmission/reception)
- Data transmission: Two methods
  - Mailbox (buffer) number order (low-to-high)
  - Message priority (identifier) reverse-order (high-to-low)
- Data reception: Two methods
  - Message identifier match (transmit/receive-setting buffers)
  - Reception with message identifier masked (receive-only)
- CPU interrupts: 12
  - Error interrupt
  - Reset processing interrupt
  - Message reception interrupt
  - Message transmission interrupt
- HCAN operating modes
- Support for various modes
  - Hardware reset
  - Software reset
  - Normal status (error-active, error-passive)
  - Bus off status
  - HCAN configuration mode
  - HCAN sleep mode
  - HCAN halt mode

- Other features
  - DTC can be activated by message reception mailbox (HCAN mailbox 0 only)
- Module stop mode can be set

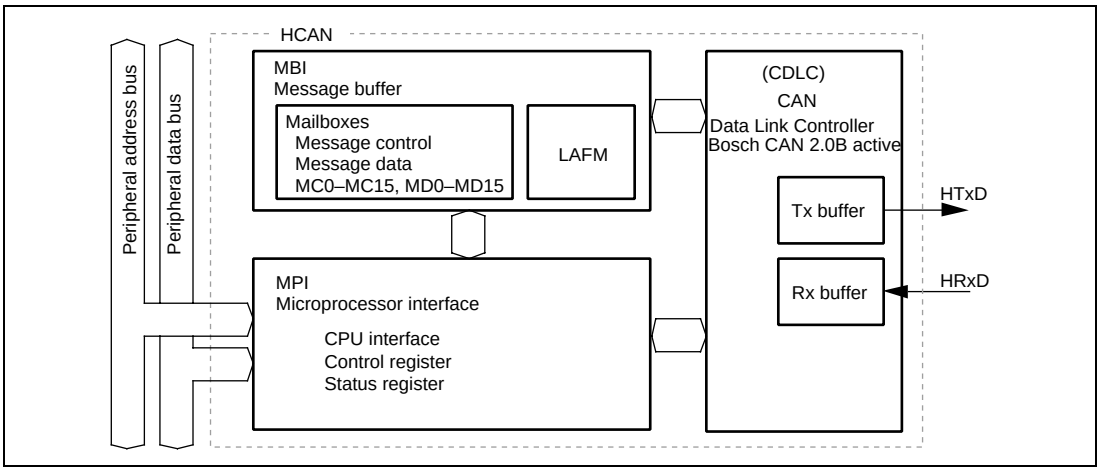



**Figure 15.1 HCAN Block Diagram**

- Message Buffer Interface (MBI)

  The MBI, consisting of mailboxes and a local acceptance filter mask (LAFM), stores CAN transmit/receive messages (identifiers, data, etc.) Transmit messages are written by the CPU. For receive messages, the data received by the CDLC is stored automatically.
- Microprocessor Interface (MPI)

  The MPI, consisting of a bus interface, control register, status register, etc., controls HCAN internal data, status, and so forth.
- CAN Data Link Controller (CDLC)

  The CDLC, conforming to the Bosch CAN Ver. 2.0B active standard, performs transmission and reception of messages (data frames, remote frames, error frames, overload frames, inter-frame spacing), as well as CRC checking, bus arbitration, and other functions.

## 15.2 Input/Output Pins

Table 15.1 shows the HCAN's pins.

When using HCAN pins, settings must be made in the HCAN configuration mode (during initialization: MCR0 = 1 and GSR3 = 1).

**Table 15.1 HCAN Pins**

| Name | Abbreviation | Input/Output | Function |
|---|---|---|---|
| HCAN transmit data pin | HTxD | Output | CAN bus transmission pin |
| HCAN receive data pin | HRxD | Input | CAN bus reception pin |

A bus driver is necessary for the interface between the pins and the CAN bus. A Philips PCA82C250 compatible model is recommended.

## 15.3 Register Descriptions

The HCAN has the following registers.

- Master control register (MCR)
- General status register (GSR)
- Bit configuration register (BCR)
- Mailbox configuration register (MBCR)
- Transmit wait register (TXPR)
- Transmit wait cancel register (TXCR)
- Transmit acknowledge register (TXACK)
- Abort acknowledge register (ABACK)
- Receive complete register (RXPR)
- Remote request register (RFPR)
- Interrupt register (IRR)
- Mailbox interrupt mask register (MBIMR)
- Interrupt mask register (IMR)
- Receive error counter (REC)
- Transmit error counter (TEC)
- Unread message status register (UMSR)
- Local acceptance filter mask H (LAFMH)
- Local acceptance filter mask L (LAFML)
- Message control (8-bit × 8 registers × 16 sets) (MC0 to MC15)
- Message data (8-bit × 8 registers × 16 sets) (MD0 to MD15)

- HCAN Monitor Register (HCANMON)

### 15.3.1 Master Control Register (MCR)

MCR is an 8-bit register that controls the HCAN.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | MCR7 | 0 | R/W | HCAN Sleep Mode Release<br>When this bit is set to 1, the HCAN automatically exits HCAN sleep mode on detection of CAN bus operation. |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | MCR5 | 0 | R/W | HCAN Sleep Mode<br>When this bit is set to 1, the HCAN transits to HCAN sleep mode. When this bit is cleared to 0, HCAN sleep mode is released. |
| 4, 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | MCR2 | 0 | R/W | Message Transmission Method<br>0: Transmission order determined by message identifier priority<br>1: Transmission order determined by mailbox (buffer) number priority (TXPR1 > TXPR15) |
| 1 | MCR1 | 0 | R/W | Halt Request<br>When this bit is set to 1, the HCAN transits to HCAN HALT mode. When this bit is cleared to 0, HCAN HALT mode is released. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | MCR0 | 1 | R/W | Reset Request<br>When this bit is set to 1, the HCAN transits to reset mode. For details, refer to section 15.4.1, Hardware and Software Resets.<br>[Setting conditions]<br>• Power-on reset<br>• Hardware standby<br>• Software standby<br>• 1-write (software reset)<br>[Clearing condition]<br>• When 0 is written to this bit while the GSR3 bit in GSR is 1 |

### 15.3.2 General Status Register (GSR)

GSR is an 8-bit register that indicates the status of the CAN bus.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | GSR3 | 1 | R | Reset Status Bit<br>Indicates whether the HCAN module is in the normal operation state or the reset state. This bit cannot be modified.<br>[Setting condition]<br>• When entering configuration mode after the HCAN internal reset has finished<br>• Sleep mode<br>[Clearing condition]<br>• When entering the normal operation state after the MCR0 bit in MCR is cleared to 0 (Note that there is a delay between clearing of the MCR0 bit and the GSR3 bit.) |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | GSR2 | 1 | R | Message Transmission Status Flag<br>Flag that indicates whether the module is currently in the message transmission period. This bit cannot be modified.<br>[Setting condition]<br>• Start of message transmission (SOF)<br>[Clearing condition]<br>• Interval of three bits after EOF (End of Frame) |
| 1 | GSR1 | 0 | R | Transmit/Receive Warning Flag<br>This bit cannot be modified.<br>[Clearing condition]<br>• When TEC < 96 and REC < 96 or TEC ≥ 256<br>[Setting condition]<br>• When TEC ≥ 96 or REC ≥ 96 |
| 0 | GSR0 | 0 | R | Bus Off Flag<br>This bit cannot be modified.<br>[Setting condition]<br>• When TEC ≥ 256 (bus off state)<br>[Clearing condition]<br>• Recovery from bus off state |

### 15.3.3 Bit Configuration Register (BCR)

BCR is a 16-bit register that is used to set HCAN bit timing parameters and the baud rate prescaler. For details on parameters, refer to section 15.4.2, Initialization after Hardware Reset.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | BCR7 | 0 | R/W | Re-Synchronization Jump Width (SJW) |
| 14 | BCR6 | 0 | R/W | Set the maximum bit synchronization width.<br>00: 1 time quantum<br>01: 2 time quanta<br>10: 3 time quanta<br>11: 4 time quanta |
| 13 | BCR5 | 0 | R/W | Baud Rate Prescaler (BRP) |
| 12 | BCR4 | 0 | R/W | Set the length of time quanta. |
| 11 | BCR3 | 0 | R/W | 000000: 2 × system clock |
| 10 | BCR2 | 0 | R/W | 000001: 4 × system clock |
| 9 | BCR1 | 0 | R/W | 000010: 6 × system clock |
| 8 | BCR0 | 0 | R/W | :<br>111111: 128 × system clock |
| 7 | BCR15 | 0 | R/W | Bit Sample Point (BSP)<br>Sets the point at which data is sampled.<br>0: Bit sampling at one point (end of time segment 1 (TSEG1))<br>1: Bit sampling at three points (end of TSEG1 and preceding and following time quanta) |
| 6 | BCR14 | 0 | R/W | Time Segment 2 (TSEG2) |
| 5 | BCR13 | 0 | R/W | Set the TSEG2 width within a range of 2 to 8 time quanta. |
| 4 | BCR12 | 0 | R/W | 000: Setting prohibited<br>001: 2 time quanta<br>010: 3 time quanta<br>011: 4 time quanta<br>100: 5 time quanta<br>101: 6 time quanta<br>110: 7 time quanta<br>111: 8 time quanta |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | BCR11 | 0 | R/W | Time Segment 1 (TSEG1) |
| 2 | BCR10 | 0 | R/W | Set the TSEG1 (PRSEG + PHSEG1) width to between 4 and 16 time quanta. |
| 1 | BCR9 | 0 | R/W | |
| 0 | BCR8 | 0 | R/W | 0000: Setting prohibited |
| | | | | 0001: Setting prohibited |
| | | | | 0010: Setting prohibited |
| | | | | 0011: 4 time quanta |
| | | | | 0100: 5 time quanta |
| | | | | 0101: 6 time quanta |
| | | | | 0110: 7 time quanta |
| | | | | 0111: 8 time quanta |
| | | | | 1000: 9 time quanta |
| | | | | 1001: 10 time quanta |
| | | | | 1010: 11 time quanta |
| | | | | 1011: 12 time quanta |
| | | | | 1100: 13 time quanta |
| | | | | 1101: 14 time quanta |
| | | | | 1110: 15 time quanta |
| | | | | 1111: 16 time quanta |

### 15.3.4 Mailbox Configuration Register (MBCR)

MBCR is a 16-bit register that is used to set the transfer direction for each mailbox.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | MBCR7 | 0 | R/W | These bits set the transfer direction for the corresponding mailboxes 1 to 15. MBCRn determines the transfer direction for mailbox n (n =1 to 15).<br>0: Corresponding mailbox is set for transmission<br>1: Corresponding mailbox is set for reception |
| 14 | MBCR6 | 0 | R/W | |
| 13 | MBCR5 | 0 | R/W | |
| 12 | MBCR4 | 0 | R/W | |
| 11 | MBCR3 | 0 | R/W | |
| 10 | MBCR2 | 0 | R/W | |
| 9 | MBCR1 | 0 | R/W | |
| 8 | — | 1 | R | Bit 8 is reserved. This bit is always read as 1. The write value should always be 1. |
| 7 | MBCR15 | 0 | R/W | |
| 6 | MBCR14 | 0 | R/W | |
| 5 | MBCR13 | 0 | R/W | |
| 4 | MBCR12 | 0 | R/W | |
| 3 | MBCR11 | 0 | R/W | |
| 2 | MBCR10 | 0 | R/W | |
| 1 | MBCR9 | 0 | R/W | |
| 0 | MBCR8 | 0 | R/W | |

### 15.3.5 Transmit Wait Register (TXPR)

TXPR is a 16-bit register that is used to set a transmit wait after a transmit message is stored in a mailbox (buffer) (CAN bus arbitration wait).

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | TXPR7 | 0 | R/W | These bits set a transmit wait (CAN bus arbitration wait) for the corresponding mailboxes 1 to 15. When TXPRn (n = 1 to 15) is set to 1, the message in mailbox n becomes the transmit wait state.<br>[Clearing condition]<br>• Completion of message transmission<br>• Completion of transmission cancellation |
| 14 | TXPR6 | 0 | R/W | |
| 13 | TXPR5 | 0 | R/W | |
| 12 | TXPR4 | 0 | R/W | |
| 11 | TXPR3 | 0 | R/W | |
| 10 | TXPR2 | 0 | R/W | |
| 9 | TXPR1 | 0 | R/W | |
| 8 | — | 0 | R | Bit 8 is reserved. This bit is always read as 1. The write value should always be 1. |
| 7 | TXPR15 | 0 | R/W | |
| 6 | TXPR14 | 0 | R/W | |
| 5 | TXPR13 | 0 | R/W | |
| 4 | TXPR12 | 0 | R/W | |
| 3 | TXPR11 | 0 | R/W | |
| 2 | TXPR10 | 0 | R/W | |
| 1 | TXPR9 | 0 | R/W | |
| 0 | TXPR8 | 0 | R/W | |

### 15.3.6 Transmit Wait Cancel Register (TXCR)

TXCR is a 16-bit register that controls the cancellation of transmit wait messages in mailboxes (buffers).

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | TXCR7 | 0 | R/W | These bits cancel the transmit wait message in the corresponding mailboxes 1 to 15. When TXCRn (n = 1 to 15) is set to 1, the transmit wait message in mailbox n is canceled.<br>[Clearing condition]<br>• Completion of TXPR clearing when transmit message is canceled normally |
| 14 | TXCR6 | 0 | R/W | |
| 13 | TXCR5 | 0 | R/W | |
| 12 | TXCR4 | 0 | R/W | |
| 11 | TXCR3 | 0 | R/W | |
| 10 | TXCR2 | 0 | R/W | |
| 9 | TXCR1 | 0 | R/W | |
| 8 | — | 0 | R | Bit 8 is reserved. This bit is always read as 0. The write value should always be 0. |
| 7 | TXCR15 | 0 | R/W | |
| 6 | TXCR14 | 0 | R/W | |
| 5 | TXCR13 | 0 | R/W | |
| 4 | TXCR12 | 0 | R/W | |
| 3 | TXCR11 | 0 | R/W | |
| 2 | TXCR10 | 0 | R/W | |
| 1 | TXCR9 | 0 | R/W | |
| 0 | TXCR8 | 0 | R/W | |

### 15.3.7 Transmit Acknowledge Register (TXACK)

TXACK is a 16-bit register containing status flags that indicate the normal transmission of mailbox (buffer) transmit messages.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | TXACK7 | 0 | R/(W)* | These bits are status flags that indicate error-free transmission of the transmit message in the corresponding mailboxes 1 to 15. When the message in mailbox n (n = 1 to 15) has been transmitted error-free, TXACKn is set to 1.<br>[Setting condition]<br>• Completion of message transmission for corresponding mailbox<br>[Clearing condition]<br>• Writing 1 |
| 14 | TXACK6 | 0 | R/(W)* | |
| 13 | TXACK5 | 0 | R/(W)* | |
| 12 | TXACK4 | 0 | R/(W)* | |
| 11 | TXACK3 | 0 | R/(W)* | |
| 10 | TXACK2 | 0 | R/(W)* | |
| 9 | TXACK1 | 0 | R/(W)* | |
| 8 | — | 0 | R | Bit 8 is reserved. This bit is always read as 0. The write value should always be 0. |
| 7 | TXACK15 | 0 | R/(W)* | |
| 6 | TXACK14 | 0 | R/(W)* | |
| 5 | TXACK13 | 0 | R/(W)* | |
| 4 | TXACK12 | 0 | R/(W)* | |
| 3 | TXACK11 | 0 | R/(W)* | |
| 2 | TXACK10 | 0 | R/(W)* | |
| 1 | TXACK9 | 0 | R/(W)* | |
| 0 | TXACK8 | 0 | R/(W)* | |

Note: * Only 1 can be written to clear the flag.

### 15.3.8 Abort Acknowledge Register (ABACK)

ABACK is a 16-bit register containing status flags that indicate the normal cancellation (aborting) of mailbox (buffer) transmit messages.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | ABACK7 | 0 | R/(W)* | These bits are status flags that indicate error-free cancellation (abortion) of the transmit message in the corresponding mailboxes 1 to 15. When the message in mailbox n (n = 1 to 15) has been canceled error-free, ABACKn is set to 1.<br>[Setting condition]<br>• Completion of transmit message cancellation for corresponding mailbox<br>[Clearing condition]<br>• Writing 1<br>Bit 8 is reserved. This bit is always read as 0. The write value should always be 0. |
| 14 | ABACK6 | 0 | R/(W)* | |
| 13 | ABACK5 | 0 | R/(W)* | |
| 12 | ABACK4 | 0 | R/(W)* | |
| 11 | ABACK3 | 0 | R/(W)* | |
| 10 | ABACK2 | 0 | R/(W)* | |
| 9 | ABACK1 | 0 | R/(W)* | |
| 8 | — | 0 | R | |
| 7 | ABACK15 | 0 | R/(W)* | |
| 6 | ABACK14 | 0 | R/(W)* | |
| 5 | ABACK13 | 0 | R/(W)* | |
| 4 | ABACK12 | 0 | R/(W)* | |
| 3 | ABACK11 | 0 | R/(W)* | |
| 2 | ABACK10 | 0 | R/(W)* | |
| 1 | ABACK9 | 0 | R/(W)* | |
| 0 | ABACK8 | 0 | R/(W)* | |

Note: * Only 1 can be written to clear the flag.

### 15.3.9 Receive Complete Register (RXPR)

RXPR is a 16-bit register containing status flags that indicate the normal reception of messages in mailboxes (buffers). For reception of a remote frame, when a bit in this register is set to 1, the corresponding remote request register (RFPR) bit is also set to 1 simultaneously.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | RXPR7 | 0 | R/(W)* | When the message in mailbox n (n = 1 to 15) has been received error-free, RXPRn is set to 1.<br>[Setting condition]<br>• Completion of message (data frame or remote frame) reception in corresponding mailbox<br>[Clearing condition]<br>• Writing 1 |
| 14 | RXPR6 | 0 | R/(W)* | |
| 13 | RXPR5 | 0 | R/(W)* | |
| 12 | RXPR4 | 0 | R/(W)* | |
| 11 | RXPR3 | 0 | R/(W)* | |
| 10 | RXPR2 | 0 | R/(W)* | |
| 9 | RXPR1 | 0 | R/(W)* | |
| 8 | RXPR0 | 0 | R/(W)* | |
| 7 | RXPR15 | 0 | R/(W)* | |
| 6 | RXPR14 | 0 | R/(W)* | |
| 5 | RXPR13 | 0 | R/(W)* | |
| 4 | RXPR12 | 0 | R/(W)* | |
| 3 | RXPR11 | 0 | R/(W)* | |
| 2 | RXPR10 | 0 | R/(W)* | |
| 1 | RXPR9 | 0 | R/(W)* | |
| 0 | RXPR8 | 0 | R/(W)* | |

Note: * Only 1 can be written to clear the flag.

### 15.3.10 Remote Request Register (RFPR)

RFPR is a 16-bit register containing status flags that indicate normal reception of remote frames in mailboxes (buffers). When a bit in this register is set to 1, the corresponding receive complete register (RXPR) bit is also set to 1 simultaneously.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | RFPR7 | 0 | R/(W)* | When mailbox n (n = 0 to 15) has received the remote frame error-free, RFPRn (n = 1 to 15) is set to 1.<br>[Setting condition]<br>• Completion of remote frame reception in corresponding mailbox<br>[Clearing condition]<br>• Writing 1 |
| 14 | RFPR6 | 0 | R/(W)* | |
| 13 | RFPR5 | 0 | R/(W)* | |
| 12 | RFPR4 | 0 | R/(W)* | |
| 11 | RFPR3 | 0 | R/(W)* | |
| 10 | RFPR2 | 0 | R/(W)* | |
| 9 | RFPR1 | 0 | R/(W)* | |
| 8 | RFPR0 | 0 | R/(W)* | |
| 7 | RFPR15 | 0 | R/(W)* | |
| 6 | RFPR14 | 0 | R/(W)* | |
| 5 | RFPR13 | 0 | R/(W)* | |
| 4 | RFPR12 | 0 | R/(W)* | |
| 3 | RFPR11 | 0 | R/(W)* | |
| 2 | RFPR10 | 0 | R/(W)* | |
| 1 | RFPR9 | 0 | R/(W)* | |
| 0 | RFPR8 | 0 | R/(W)* | |

Note: * Only 1 can be written to clear the flag.

### 15.3.11 Interrupt Register (IRR)

IRR is a 16-bit interrupt status flag register.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | IRR7 | 0 | R/(W)* | Overload Frame<br>[Setting condition]<br>• When an overload frame is transmitted in error active/passive state<br>[Clearing condition]<br>• Writing 1 |
| 14 | IRR6 | 0 | R/(W)* | Bus Off Interrupt Flag<br>Status flag indicating the bus off state caused by the transmit error counter.<br>[Setting condition]<br>• When TEC ≥ 256<br>[Clearing condition]<br>• Writing 1 |
| 13 | IRR5 | 0 | R/(W)* | Error Passive Interrupt Flag<br>Status flag indicating the error passive state caused by the transmit/receive error counter.<br>[Setting condition]<br>When TEC ≥ 128 or REC ≥ 128<br>[Clearing condition]<br>• Writing 1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | IRR4 | 0 | R/(W)* | Receive Overload Warning Interrupt Flag<br>Status flag indicating the error warning state caused by the receive error counter.<br>[Setting condition]<br>When REC ≥ 96<br>[Clearing condition]<br>• Writing 1 |
| 11 | IRR3 | 0 | R/(W)* | Transmit Overload Warning Interrupt Flag<br>Status flag indicating the error warning state caused by the transmit error counter.<br>[Setting condition]<br>• When TEC ≥ 96<br>[Clearing condition]<br>• Writing 1 |
| 10 | IRR2 | 0 | R | Remote Frame Request Interrupt Flag<br>Status flag indicating that a remote frame has been received in a mailbox (buffer).<br>[Setting condition]<br>• When remote frame reception is completed, when corresponding MBIMR = 0<br>[Clearing condition]<br>• Clearing of all bits in RFPR (remote request register) |
| 9 | IRR1 | 0 | R | Receive Message Interrupt Flag<br>Status flag indicating that a mailbox (buffer) receive message has been received normally.<br>[Setting condition]<br>• When data frame or remote frame reception is completed, when corresponding MBIMR = 0<br>[Clearing condition]<br>• Clearing of all bits in RXPR (receive complete register) |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | IRR0 | 1 | R/(W)* | Reset Interrupt Flag<br>Status flag indicating that the HCAN module has been reset. This bit cannot be masked by the interrupt mask register (IMR). If this bit is not cleared to 0 after entering power-on reset or returning from software standby mode, interrupt processing will start immediately when the interrupt controller enables interrupts.<br>[Setting condition]<br>• When the reset operation has finished after entering power-on reset or software standby mode<br>[Clearing condition]<br>• Writing 1 |
| 7 to 5 | — | All 0 | — | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | IRR12 | 0 | R/(W)* | Bus Operation Interrupt Flag<br>Status flag indicating detection of a dominant bit due to bus operation when the HCAN module is in HCAN sleep mode.<br>[Setting condition]<br>• Bus operation (dominant bit) detection in HCAN sleep mode<br>[Clearing condition]<br>• Writing 1 |
| 3, 2 | — | All 0 | — | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | IRR9 | 0 | R | Unread Interrupt Flag<br>Status flag indicating that a receive message has been overwritten before being read.<br>[Setting condition]<br>• When UMSR (unread message status register) is set<br>[Clearing condition]<br>• Clearing of all bits in UMSR (unread message status register) |
| 0 | IRR8 | 0 | R/(W)* | Mailbox Empty Interrupt Flag<br>Status flag indicating that the next transmit message can be stored in the mailbox.<br>[Setting condition]<br>• When TXPR (transmit wait register) is cleared by completion of transmission or completion of transmission abort<br>[Clearing condition]<br>• Writing 1 |

Note: * Only 1 can be written to clear the flag.

### 15.3.12 Mailbox Interrupt Mask Register (MBIMR)

MBIMR is a 16-bit register that controls the enabling or disabling of individual mailbox (buffer) interrupt requests.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | MBIMR7 | 1 | R/W | Mailbox Interrupt Mask (MBIMRx) |
| 14 | MBIMR6 | 1 | R/W | When MBIMRn (n = 1 to 15) is cleared to 0, the interrupt request in mailbox n is enabled. When set to 1, the interrupt request is masked. |
| 13 | MBIMR5 | 1 | R/W | |
| 12 | MBIMR4 | 1 | R/W | The interrupt source in a transmit mailbox is TXPR clearing caused by transmission end or transmission cancellation. The interrupt source in a receive mailbox is RXPR setting on reception end. |
| 11 | MBIMR3 | 1 | R/W | |
| 10 | MBIMR2 | 1 | R/W | |
| 9 | MBIMR1 | 1 | R/W | |
| 8 | MBIMR0 | 1 | R/W | |
| 7 | MBIMR15 | 1 | R/W | |
| 6 | MBIMR14 | 1 | R/W | |
| 5 | MBIMR13 | 1 | R/W | |
| 4 | MBIMR12 | 1 | R/W | |
| 3 | MBIMR11 | 1 | R/W | |
| 2 | MBIMR10 | 1 | R/W | |
| 1 | MBIMR9 | 1 | R/W | |
| 0 | MBIMR8 | 1 | R/W | |

### 15.3.13 Interrupt Mask Register (IMR)

IMR is a 16-bit register containing flags that enable or disable requests by individual interrupt sources. The reset interrupt flag cannot be masked.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | IMR7 | 1 | R/W | Overload Frame Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR7 (OVR0) is enabled. When set to 1, it is masked. |
| 14 | IMR6 | 1 | R/W | Bus Off Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR6 (ERS0) is enabled. When set to 1, it is masked. |
| 13 | IMR5 | 1 | R/W | Error Passive Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR5 (ERS0) is enabled. When set to 1, it is masked. |
| 12 | IMR4 | 1 | R/W | Receive Overload Warning Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR4 (OVR0) is enabled. When set to 1, it is masked. |
| 11 | IMR3 | 1 | R/W | Transmit Overload Warning Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR3 (OVR0) is enabled. When set to 1, it is masked. |
| 10 | IMR2 | 1 | R/W | Remote Frame Request Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR2 (OVR0) is enabled. When set to 1, it is masked. |
| 9 | IMR1 | 1 | R/W | Receive Message Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR1 (RM1) is enabled. When set to 1, it is masked. |
| 8 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 7 to 5 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | IMR12 | 1 | R/W | Bus Operation Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR12 (OVR0) is enabled. When set to 1, it is masked. |
| 3, 2 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 0. |
| 1 | IMR9 | 1 | R/W | Unread Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR9 (OVR0) is enabled. When set to 1, it is masked. |
| 0 | IMR8 | 1 | R/W | Mailbox Empty Interrupt Mask<br>When this bit is cleared to 0, an interrupt request by IRR8 (SLE0) is enabled. When set to 1, it is masked. |

### 15.3.14 Receive Error Counter (REC)

The receive error counter (REC) is an 8-bit read-only register that functions as a counter indicating the number of receive message errors on the CAN bus. The count value is stipulated in the CAN protocol.

### 15.3.15 Transmit Error Counter (TEC)

The transmit error counter (TEC) is an 8-bit read-only register that functions as a counter indicating the number of transmit message errors on the CAN bus. The count value is stipulated in the CAN protocol.

### 15.3.16 Unread Message Status Register (UMSR)

UMSR is a 16-bit register containing status flags that indicate, for individual mailboxes (buffers), that a received message has been overwritten by a new receive message before being read. When overwritten by a new message, data in the unread receive message is lost.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | UMSR7 | 0 | R/(W)* | [Setting condition] |
| 14 | UMSR6 | 0 | R/(W)* | • When a new message is received before RXPR is cleared |
| 13 | UMSR5 | 0 | R/(W)* | |
| 12 | UMSR4 | 0 | R/(W)* | [Clearing condition] |
| 11 | UMSR3 | 0 | R/(W)* | • Writing 1 |
| 10 | UMSR2 | 0 | R/(W)* | • When the received message has been overwritten by a new message before being read. |
| 9 | UMSR1 | 0 | R/(W)* | |
| 8 | UMSR0 | 0 | R/(W)* | |
| 7 | UMSR15 | 0 | R/(W)* | |
| 6 | UMSR14 | 0 | R/(W)* | |
| 5 | UMSR13 | 0 | R/(W)* | |
| 4 | UMSR12 | 0 | R/(W)* | |
| 3 | UMSR11 | 0 | R/(W)* | |
| 2 | UMSR10 | 0 | R/(W)* | |
| 1 | UMSR9 | 0 | R/(W)* | |
| 0 | UMSR8 | 0 | R/(W)* | |

Note: * Only 1 can be written to clear the flag.

### 15.3.17 Local Acceptance Filter Masks (LAFML, LAFMH)

LAFML and LAFMH are 16-bit registers that individually set the identifier bits of the message to be stored in mailbox 0 as Don't Care. For details, refer to section 15.4.4, Message Reception. The relationship between the identifier bits and mask bits are shown in the following.

LAFML

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | LAFML7 | 0 | R/W | When this bit is set to 1, ID-7 of the receive message identifier is not compared. |
| 14 | LAFML6 | 0 | R/W | When this bit is set to 1, ID-6 of the receive message identifier is not compared. |
| 13 | LAFML5 | 0 | R/W | When this bit is set to 1, ID-5 of the receive message identifier is not compared. |
| 12 | LAFML4 | 0 | R/W | When this bit is set to 1, ID-4 of the receive message identifier is not compared. |
| 11 | LAFML3 | 0 | R/W | When this bit is set to 1, ID-3 of the receive message identifier is not compared. |
| 10 | LAFML2 | 0 | R/W | When this bit is set to 1, ID-2 of the receive message identifier is not compared. |
| 9 | LAFML1 | 0 | R/W | When this bit is set to 1, ID-1 of the receive message identifier is not compared. |
| 8 | LAFML0 | 0 | R/W | When this bit is set to 1, ID-0 of the receive message identifier is not compared. |
| 7 | LAFML15 | 0 | R/W | When this bit is set to 1, ID-15 of the receive message identifier is not compared. |
| 6 | LAFML14 | 0 | R/W | When this bit is set to 1, ID-14 of the receive message identifier is not compared. |
| 5 | LAFML13 | 0 | R/W | When this bit is set to 1, ID-13 of the receive message identifier is not compared. |
| 4 | LAFML12 | 0 | R/W | When this bit is set to 1, ID-12 of the receive message identifier is not compared. |
| 3 | LAFML11 | 0 | R/W | When this bit is set to 1, ID-11 of the receive message identifier is not compared. |
| 2 | LAFML10 | 0 | R/W | When this bit is set to 1, ID-10 of the receive message identifier is not compared. |
| 1 | LAFML9 | 0 | R/W | When this bit is set to 1, ID-9 of the receive message identifier is not compared. |
| 0 | LAFML8 | 0 | R/W | When this bit is set to 1, ID-8 of the receive message identifier is not compared. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | LAFMH7 | 0 | R/W | When this bit is set to 1, ID-20 of the receive message identifier is not compared. |
| 14 | LAFMH6 | 0 | R/W | When this bit is set to 1, ID-19 of the receive message identifier is not compared. |
| 13 | LAFMH5 | 0 | R/W | When this bit is set to 1, ID-18 of the receive message identifier is not compared. |
| 12 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | LAFMH1 | 0 | R/W | When this bit is set to 1, ID-17 of the receive message identifier is not compared. |
| 8 | LAFMH0 | 0 | R/W | When this bit is set to 1, ID-16 of the receive message identifier is not compared. |
| 7 | LAFMH15 | 0 | R/W | When this bit is set to 1, ID-28 of the receive message identifier is not compared. |
| 6 | LAFMH14 | 0 | R/W | When this bit is set to 1, ID-27 of the receive message identifier is not compared. |
| 5 | LAFMH13 | 0 | R/W | When this bit is set to 1, ID-26 of the receive message identifier is not compared. |
| 4 | LAFMH12 | 0 | R/W | When this bit is set to 1, ID-25 of the receive message identifier is not compared. |
| 3 | LAFMH11 | 0 | R/W | When this bit is set to 1, ID-24 of the receive message identifier is not compared. |
| 2 | LAFMH10 | 0 | R/W | When this bit is set to 1, ID-23 of the receive message identifier is not compared. |
| 1 | LAFMH9 | 0 | R/W | When this bit is set to 1, ID-22 of the receive message identifier is not compared. |
| 0 | LAFMH8 | 0 | R/W | When this bit is set to 1, ID-21 of the receive message identifier is not compared. |

### 15.3.18 Message Control (MC0 to MC15)

The message control register sets consist of eight 8-bit registers for one mailbox. The HCAN has 16 sets of these registers. Because message control registers are in RAM, their initial values after power-on are undefined. Be sure to initialize them by writing 0 or 1. Figure 15.2 shows the register names for each mailbox.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mail box 0 | MC0[1] | MC0[2] | MC0[3] | MC0[4] | MC0[5] | MC0[6] | MC0[7] | MC0[8] |
| Mail box 1 | MC1[1] | MC1[2] | MC1[3] | MC1[4] | MC1[5] | MC1[6] | MC1[7] | MC1[8] |
| Mail box 2 | MC2[1] | MC2[2] | MC2[3] | MC2[4] | MC2[5] | MC2[6] | MC2[7] | MC2[8] |
| Mail box 3 | MC3[1] | MC3[2] | MC3[3] | MC3[4] | MC3[5] | MC3[6] | MC3[7] | MC3[8] |
| | | | | | | | | |
| Mail box 15 | MC15[1] | MC15[2] | MC15[3] | MC15[4] | MC15[5] | MC15[6] | MC15[7] | MC15[8] |

**Figure 15.2 Message Control Register Configuration**

The setting of message control registers are shown in the following. Figures 15.3 and 15.4 show the correspondence between the identifiers and register bit names.

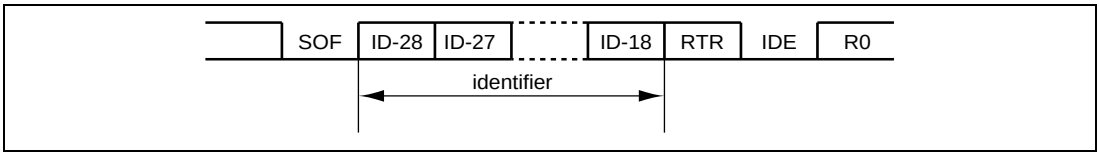



**Figure 15.3 Standard Format**

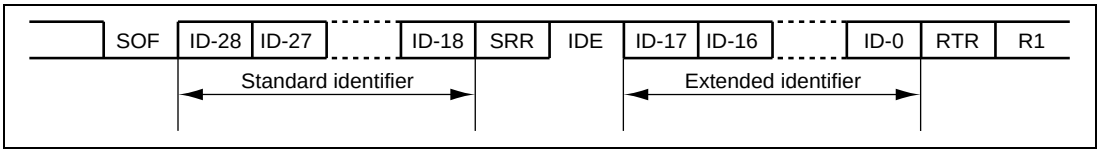



**Figure 15.4 Extended Format**

| Register Name | Bit | Bit Name | R/W | Description |
|---|---|---|---|---|
| MCx[1] | 7 to 4 | — | R/W | The initial value of these bits is undefined. They must be initialized by writing 0 or 1. |
| | 3 to 0 | DLC3 to DLC0 | R/W | Data Length Code<br>Set the data length of a data frame or the data length requested in a remote frame within the range of 0 to 8 bits.<br>0000: 0 byte<br>0001: 1 byte<br>0010: 2 bytes<br>0011: 3 bytes<br>0100: 4 bytes<br>0101: 5 bytes<br>0110: 6 bytes<br>0111: 7 bytes<br>1000: 8 bytes<br>:<br>:<br>1111: 8 bytes |
| MCx[2] | 7 to 0 | — | R/W | The initial value of these bits is undefined; they must be initialized by writing 0 or 1. |
| MCx[3] | 7 to 0 | — | R/W | |
| MCx[4] | 7 to 0 | — | R/W | |
| MCx[5] | 7 to 5 | ID-20 to ID-18 | R/W | Sets ID-20 to ID-18 in the identifier. |
| | 4 | RTR | R/W | Remote Transmission Request<br>Used to distinguish between data frames and remote frames.<br>0: Data frame<br>1: Remote frame |
| | 3 | IDE | R/W | Identifier Extension<br>Used to distinguish between the standard format and extended format of data frames and remote frames.<br>0: Standard format<br>1: Extended format |
| | 2 | — | R/W | The initial value of this bit is undefined. It must be initialized by writing 0 or 1. |
| | 1 to 0 | ID-17 to ID-16 | R/W | Sets ID-17 and ID-16 in the identifier. |
| MCx[6] | 7 to 0 | ID-28 to ID-21 | R/W | Sets ID-28 to ID-21 in the identifier. |
| MCx[7] | 7 to 0 | ID-7 to ID-0 | R/W | Sets ID-7 to ID-0 in the identifier. |
| MCx[8] | 7 to 0 | ID-15 to ID-8 | R/W | Sets ID-15 to ID-8 in the identifier. |

Note: x: Mailbox number

### 15.3.19 Message Data (MD0 to MD15)

The message data register sets consist of eight 8-bit registers for one mailbox. The HCAN has 16 sets of these registers. Because message data registers are in RAM, their initial values after power-on are undefined. Be sure to initialize them by writing 0 or 1. Figure 15.5 shows the register names for each mailbox.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mail box 0 | MD0[1] | MD0[2] | MD0[3] | MD0[4] | MD0[5] | MD0[6] | MD0[7] | MD0[8] |
| Mail box 1 | MD1[1] | MD1[2] | MD1[3] | MD1[4] | MD1[5] | MD1[6] | MD1[7] | MD1[8] |
| Mail box 2 | MD2[1] | MD2[2] | MD2[3] | MD2[4] | MD2[5] | MD2[6] | MD2[7] | MD2[8] |
| Mail box 3 | MD3[1] | MD3[2] | MD3[3] | MD3[4] | MD3[5] | MD3[6] | MD3[7] | MD3[8] |
| | | | | | | | | |
| Mail box 15 | MD15[1] | MD15[2] | MD15[3] | MD15[4] | MD15[5] | MD15[6] | MD15[7] | MD15[8] |

**Figure 15.5 Message Data Configuration**

### 15.3.20 HCAN Monitor Register (HCANMON)

HCANMON is an 8-bit register that enables/disables an HCAN receive interrupt, controls transmission stop of the HTxD pin, and reflects the states of the HCAN pins.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | RxDIE | 0 | R/W | HRxD Interrupt Enable<br>Selects whether an $\overline{\text{IRQ2}}$ interrupt is caused by PF0 or HRxD pin.<br>0: An $\overline{\text{IRQ2}}$ interrupt is caused by pin PF0<br>1: An $\overline{\text{IRQ2}}$ interrupt is caused by the HRxD pin |
| 6 | TxSTP | 0 | R/W | HTxD Transmission Stop<br>Controls transmission stop of the HTxD pin.<br>0: Enables transmission from the HTxD pin<br>1: Fixes an output level of the HTxD pin at 1 and transmission is stopped |
| 5 to 2 | — | Undefined | — | Reserved<br>These bits are always read as undefined values and cannot be modified. |
| 1 | TxD | Undefined | R | Transmission pin<br>The state of the HTxD pin is read.<br>This bit cannot be modified. |
| 0 | RxD | Undefined | R | Reception pin<br>The state of the HRxD pin is read.<br>This bit cannot be modified. |

## 15.4 Operation

### 15.4.1 Hardware and Software Resets

The HCAN can be reset by a hardware reset or software reset.

- Hardware Reset

  At power-on reset, or in hardware or software standby mode, the HCAN is initialized by automatically setting the MCR reset request bit (MCR0) in MCR and the reset state bit (GSR3) in GSR. At the same time, all internal registers, except for message control and message data registers, are initialized by a hardware reset.
- Software Reset

  The HCAN can be reset by setting the MCR reset request bit (MCR0) in MCR via software. In a software reset, the error counters (TEC and REC) are initialized, however other registers are not. If the MCR0 bit is set while the CAN controller is performing a communication operation (transmission or reception), the initialization state is not entered until message transfer has been completed. The reset status bit (GSR3) in GSR is set on completion of initialization.

### 15.4.2 Initialization after Hardware Reset

After a hardware reset, the following initialization processing should be carried out:

1. Clearing of IRR0 bit in the interrupt register (IRR)
2. Bit rate setting
3. Mailbox transmit/receive settings
4. Mailbox (RAM) initialization
5. Message transmission method setting

These initial settings must be made while the HCAN is in bit configuration mode. Configuration mode is a state in which the GSR3 bit in GSR is set to 1 by a reset. Configuration mode is exited by clearing the MCR0 bit in MCR to 0; when the MCR0 bit is cleared to 0, the HCAN automatically clears the GSR3 bit in GSR. There is a delay between clearing the MCR0 bit and clearing the GSR3 bit because the HCAN needs time to be internally reset. After the HCAN exits configuration mode, the power-up sequence begins, and communication with the CAN bus is possible as soon as 11 consecutive recessive bits have been detected.

**IRR0 Clearing:** The reset interrupt flag (IRR0) is always set after a power-on reset or recovery from software standby mode. Since an HCAN interrupt is initiated immediately when interrupts are enabled, IRR0 should be cleared.

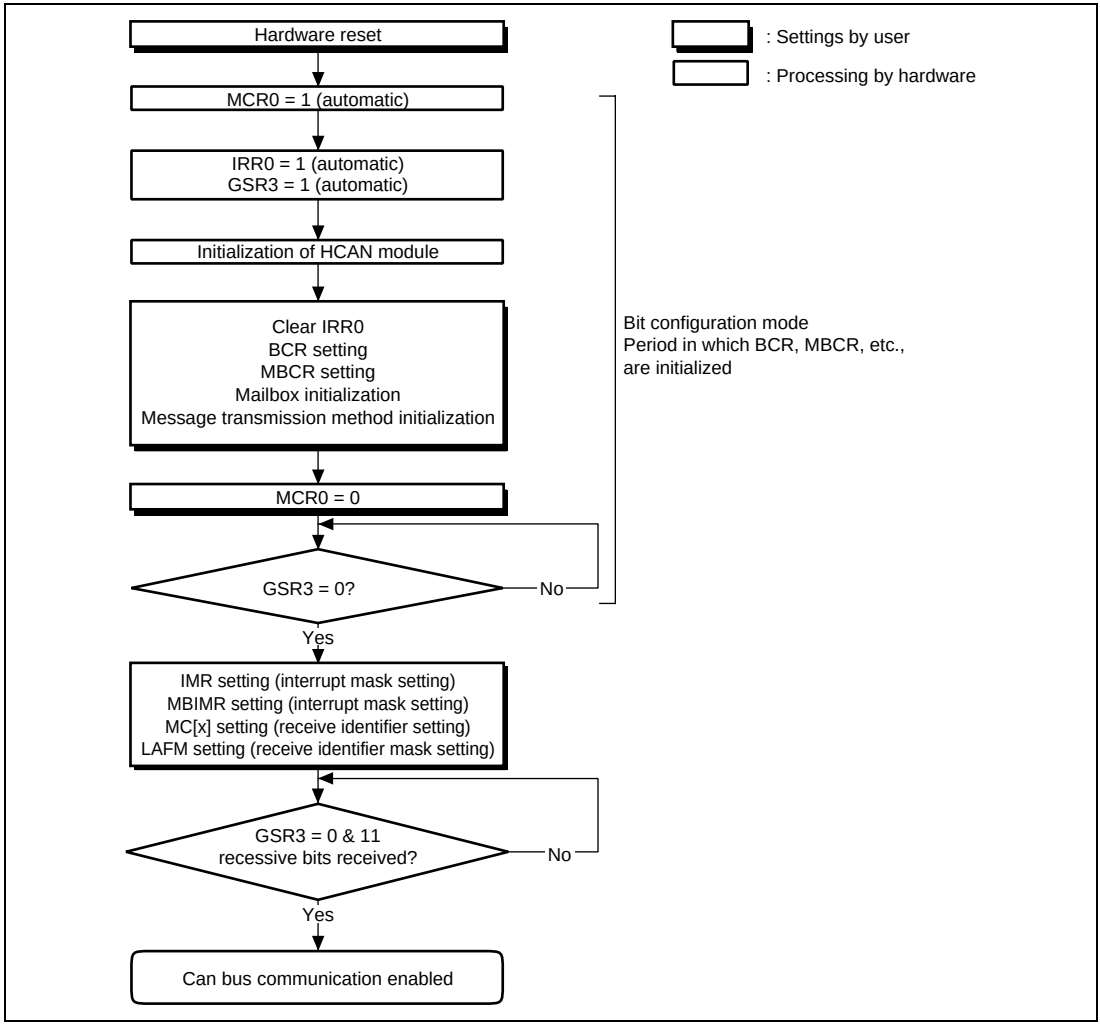



**Figure 15.6 Hardware Reset Flowchart**

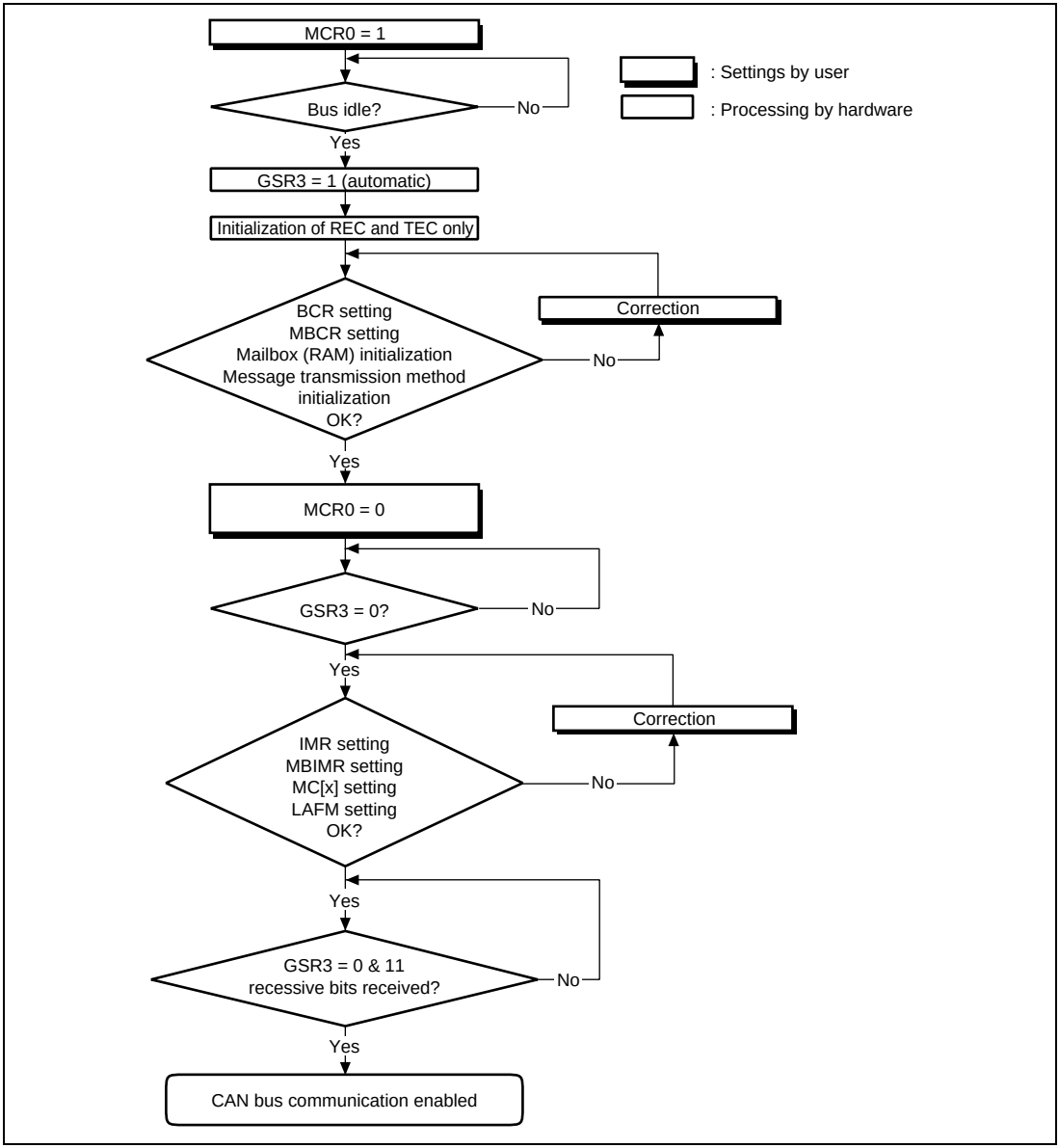



**Figure 15.7 Software Reset Flowchart**

**Bit Rate and Bit Timing Settings:** The bit rate and bit timing settings are made in the bit configuration register (BCR). Settings should be made such that all CAN controllers connected to the CAN bus have the same baud rate and bit width. The 1-bit time consists of the total of the settable time quanta (tq).

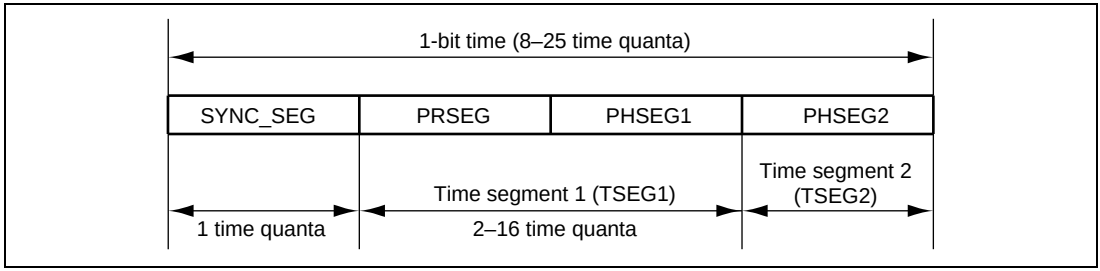



**Figure 15.8 Detailed Description of One Bit**

SYNC_SEG is a segment for establishing the synchronization of nodes on the CAN bus. Normal bit edge transitions occur in this segment. PRSEG is a segment for compensating for the physical delay between networks. PHSEG1 is a buffer segment for correcting phase drift (positive). This segment is extended when synchronization (resynchronization) is established. PHSEG2 is a buffer segment for correcting phase drift (negative). This segment is shortened when synchronization (resynchronization) is established. Limits on the settable value (TSEG1, TSEG2, BRP, sample point, and SJW) are shown in table 15.2.

**Table 15.2 Limits for the Settable Value**

| Name | Abbreviation | Min. Value | Max. Value |
|---|---|---|---|
| Time segment 1 | TSEG1 | B′0011*[2] | B′1111 |
| Time segment 2 | TSEG2 | B′001*[3] | B′111 |
| Baud rate prescaler | BRP | B′000000 | B′111111 |
| Bit sample point | BSP | B′0 | B′1 |
| Re-synchronization jump width | SJW*[1] | B′00 | B′11 |

Notes: 1. SJW is stipulated in the CAN specifications:
$3 \geq SJW \geq 0$
2. The minimum value of TSEG2 is stipulated in the CAN specifications:
$TSEG2 \geq SJW$
3. The minimum value of TSEG1 is stipulated in the CAN specifications:
$TSEG1 > TSEG2$

Time quanta (tq) is an integer multiple of the number of system clocks, and is determined by the baud rate prescaler (BRP) as follows. $f_{CLK}$ is the system clock frequency.

$tq = 2 \times (BPR\ setting + 1)/f_{CLK}$

The following formula is used to calculate the 1-bit time and bit rate.

1-bit time = $tq \times (3 + TSEG1 + TSEG2)$
Bit rate = 1/Bit time
$= f_{CLK}/\{2 \times (BPR\ setting + 1) \times (3 + TSEG1 + TSEG2)\}$

Note: $f_{CLK} = \phi$ (system clock)
A BCR value is used for BRP, TSEG1, and TSEG2.

Example: With a system clock of 24 MHz, a BRP setting of B′000000, a TSEG1 setting of B′0101, and a TSEG2 setting of B′100:

Bit rate = $24/\{2 \times (0 + 1) \times (3 + 5 + 4)\} = 1$ Mbps

**Table 15.3 Setting Range for TSEG1 and TSEG2 in BCR**

| | | TSEG2 (BCR14–BCR12) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| TSEG1 (BCR11–BCR8) | 0011 | No | Yes | No | No | No | No | No |
| | 0100 | Yes* | Yes | Yes | No | No | No | No |
| | 0101 | Yes* | Yes | Yes | Yes | No | No | No |
| | 0110 | Yes* | Yes | Yes | Yes | Yes | No | No |
| | 0111 | Yes* | Yes | Yes | Yes | Yes | Yes | No |
| | 1000 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1001 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1010 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1011 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1100 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1101 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1110 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |
| | 1111 | Yes* | Yes | Yes | Yes | Yes | Yes | Yes |

Note: The time quantum values for TSEG1 and TSEG2 are determined by TSEG value + 1.
* Settable when bits BRP13 to BRP8 are not B′000000.

**Mailbox Transmit/Receive Settings:** The HCAN has 16 mailboxes. Mailbox 0 is receive-only, while mailboxes 1 to 15 can be set for transmission or reception. The initial status of mailboxes 1 to 15 is for transmission. Mailbox transmit/receive settings are not initialized by a software reset.

Clearing a bit to 0 in the mailbox configuration register (MBCR) designates the corresponding mailbox for transmission use, whereas a setting of 1 in MBCR designates the corresponding mailbox for reception use. When setting mailboxes for reception, in order to improve message reception efficiency, high-priority messages should be set in low-to-high mailbox order.

**Mailbox (Message Control/Data) Initial Settings:** Message control/data are held in RAM, and so their initial values are undefined after power is supplied. Initial values must therefore be set in all the mailboxes (by writing 0s or 1s).

**Setting the Message Transmission Method:** The following two kinds of message transmission methods are available.

- Transmission order determined by message identifier priority
- Transmission order determined by mailbox number priority

Either of the message transmission methods can be selected with the message transmission method bit (MCR2) in the master control register (MCR): When messages are set to be transmitted according to the message identifier priority, if several messages are designated as waiting for transmission (TXPR = 1), the message with the highest priority in the message identifier is stored in the transmit buffer. CAN bus arbitration is then carried out for the message stored in the transmit buffer, and the message is transmitted when the transmission right is acquired. When the TXPR bit is set, the highest-priority message is found and stored in the transmit buffer.

When messages are set to be transmitted according to the mailbox number priority, if several messages are designated as waiting for transmission (TXPR = 1), messages are stored in the transmit buffer in low-to-high mailbox order. CAN bus arbitration is then carried out for the message stored in the transmit buffer, and the message is transmitted when the transmission right is acquired.

### 15.4.3 Message Transmission

Messages are transmitted using mailboxes 1 to 15. The transmission procedure after initial settings is described below, and a transmission flowchart is shown in figure 15.9.

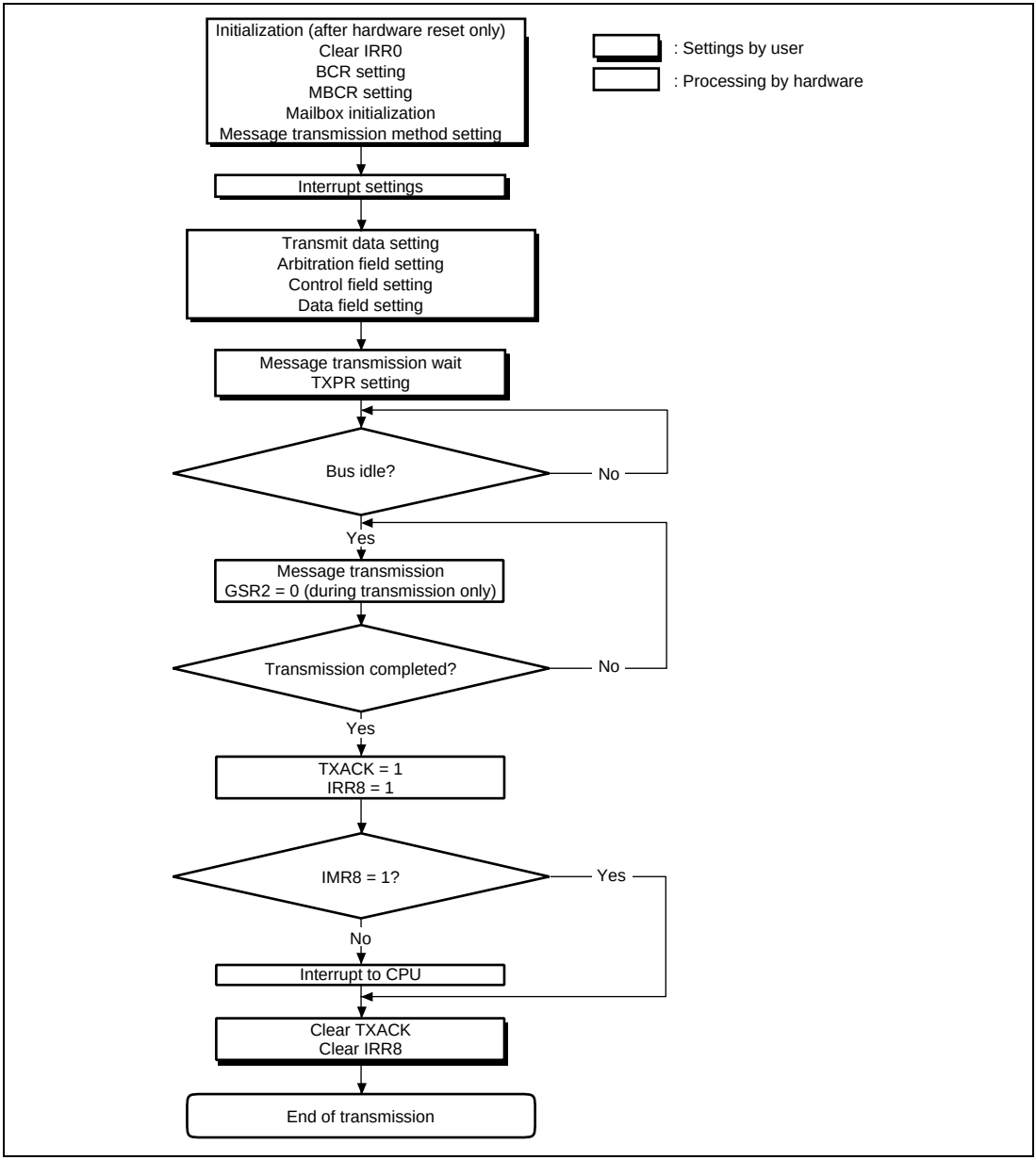



**Figure 15.9 Transmission Flowchart**

**CPU interrupt source settings:** The CPU interrupt source is set by the interrupt mask register (IMR) and mailbox interrupt mask register (MBIMR). Transmission acknowledge and transmission abort acknowledge interrupts can be generated for individual mailboxes in the mailbox interrupt mask register (MBIMR).

**Arbitration field setting:** The arbitration field is set by the message control registers MCx[5] to MCx[8] in a transmit mailbox. For a standard format, an 11-bit identifier (ID-28 to ID-18) and the RTR bit are set, and the IDE bit is cleared to 0. For an extended format, a 29-bit identifier (ID-28 to ID-0) and the RTR bit are set, and the IDE bit is set to 1.

**Control field setting:** In the control field, the byte length of the data to be transmitted is set within the range of zero to eight bytes. The register to be set is the message control register MCx[1] in a transmit mailbox.

**Data field setting:** In the data field, the data to be transmitted is set within the range zero to eight. The registers to be set are the message data registers MDx[1] to MDx[8]. The byte length of the data to be transmitted is determined by the data length code in the control field. Even if data exceeding the value set in the control field is set in the data field, up to the byte length set in the control field will actually be transmitted.

**Message transmission:** If the corresponding mailbox transmit wait bit (TXPR1 to TXPR15) in the transmit wait register (TXPR) is set to 1 after message control and message data registers have been set, the message enters transmit wait state. If the message is transmitted error-free, the corresponding acknowledge bit (TXACK1 to TXACK15) in the transmit acknowledge register (TXACK) is set to 1, and the corresponding transmit wait bit (TXPR1 to TXPR15) in the transmit wait register (TXPR) is automatically cleared to 0. Also, if the corresponding bit (MBIMR1 to MBIMR15) in the mailbox interrupt mask register (MBIMR) and the mailbox empty interrupt bit (IRR8) in the interrupt mask register (IMR) are both simultaneously set to enable interrupts, interrupts may be sent to the CPU.

If transmission of a transmit message is aborted in the following cases, the message is retransmitted automatically:

- CAN bus arbitration failure (failure to acquire the bus)
- Error during transmission (bit error, stuff error, CRC error, frame error, or ACK error)

**Message transmission cancellation:** Transmission cancellation can be specified for a message stored in a mailbox as a transmit wait message. A transmit wait message is canceled by setting the bit for the corresponding mailbox (TXCR1 to TXCR15) to 1 in the transmit cancel register (TXCR). Clearing the transmit wait register (TXPR) does not cancel transmission. When cancellation is executed, the transmit wait register (TXPR) is automatically reset, and the corresponding bit is set to 1 in the abort acknowledge register (ABACK), and then an interrupt to the CPU can be requested. Also, if the corresponding bit (MBIMR1-MBIMR15) in the mailbox

interrupt mask register (MBIMR) and the mailbox empty interrupt bit (IRR8) in the interrupt mask register (IMR) are both simultaneously set to enable interrupts, interrupts may be sent to the CPU.

However, a transmit wait message cannot be canceled at the following times:

- During internal arbitration or CAN bus arbitration
- During data frame or remote frame transmission

Figure 15.10 shows a flowchart for transmit message cancellation.

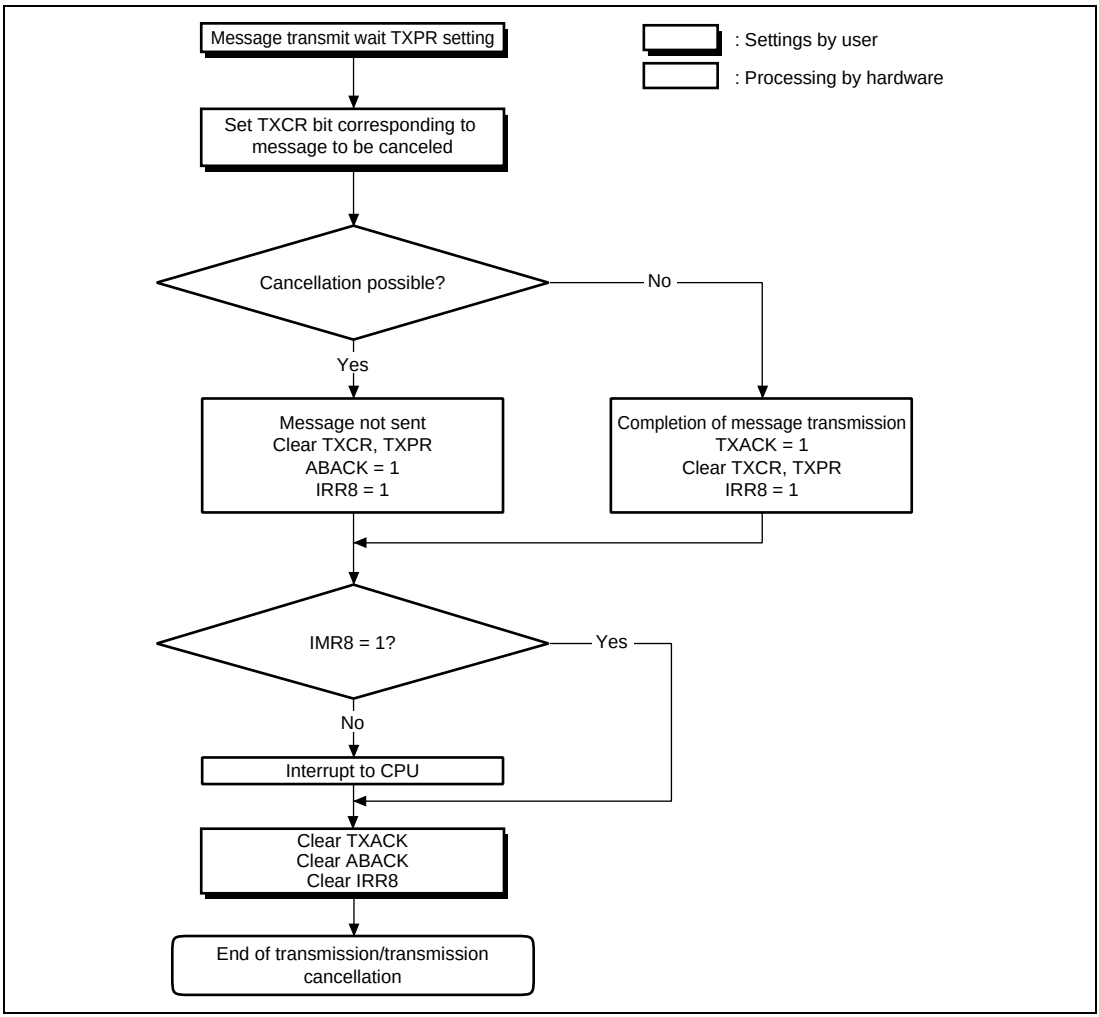



**Figure 15.10 Transmit Message Cancellation Flowchart**

### 15.4.4 Message Reception

The reception procedure after initial settings is described below. A reception flowchart is shown in Figure 15.11.

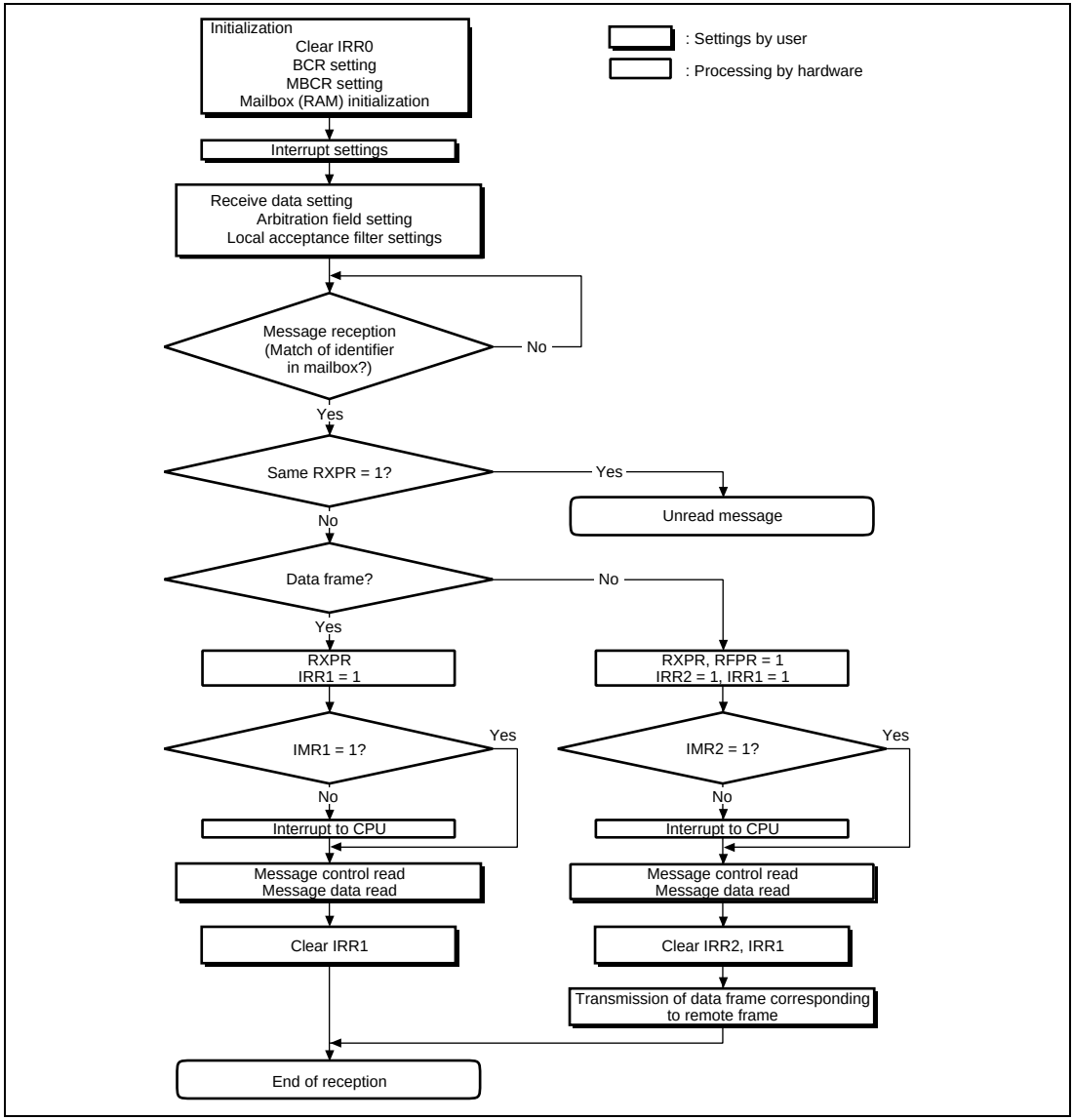



**Figure 15.11 Reception Flowchart**

**CPU interrupt source settings:** CPU interrupt source settings are made in the interrupt mask register (IMR) and mailbox interrupt register (MBIMR). The message to be received is also specified. Data frame and remote frame receive wait interrupt requests can be generated for individual mailboxes in the MBIMR.

**Arbitration field setting:** To receive a message, the message identifier must be set in advance in the message control registers (MCx[1] to MCx[8]) for the receiving mailbox. When a message is received, all the bits in the receive message identifier are compared with those in each message control register identifier, and if a complete match is found, the message is stored in the matching mailbox. Mailbox 0 has a local acceptance filter mask (LAFM) that allows Don't Care settings. The LAFM setting can be made only for mailbox 0. By setting the Don't Care for all the bits in the receive message identifier, messages of multiple identifiers can be received.

Examples:

- When the identifier of mailbox 1 is 010_1010_1010 (standard format), only one kind of message identifier can be received by mailbox 1:

  Identifier 1: 010_1010_1010
- When the identifier of mailbox 0 is 010_1010_1010 (standard format) and the LAFM setting is 000_0000_0011 (0: Care, 1: Don't Care), a total of four kinds of message identifiers can be received by mailbox 0:

  Identifier 1: 010_1010_1000

  Identifier 2: 010_1010_1001

  Identifier 3: 010_1010_1010

  Identifier 4: 010_1010_1011

**Message reception:** When a message is received, a CRC check is performed automatically. If the result of the CRC check is normal, ACK is transmitted in the ACK field irrespective of whether the message can be received or not.

- Data frame reception

  If the received message is confirmed to be error-free by the CRC check, the identifier in the mailbox (and also LAFM in the case of mailbox 0 only) and the identifier of the receive message are compared. If a complete match is found, the message is stored in the matching mailbox. The message identifier comparison is carried out on each mailbox in turn, starting with mailbox 0 and ending with mailbox 15. If a complete match is found, the comparison ends at that point, the message is stored in the matching mailbox, and the corresponding receive complete bit (RXPR0 to RXPR15) in the receive complete register (RXPR) is set. However, if the identifier matches that of mailbox 0 LAFM, the mailbox comparison sequence does not end at that point, but continues from mailbox 1. Therefore, the message for mailbox 0 can also be received by another mailbox. Note that the same message cannot be stored in two

or more mailbox of the mailboxes 1 to 15. On receiving a message, a CPU interrupt request may be generated according to the settings of the mailbox interrupt mask register (MBIMR) and interrupt mask register (IMR).

- Remote frame reception

  A mailbox can store two kinds of messages: data frames and remote frames. A remote frame differs from a data frame in the value of the remote transmission request bit (RTR) in the message control register and its 0-byte data field. The data length to be returned in a data frame must be stored in the data length code (DLC) in the message control.

  When a remote frame (RTR = recessive) is received, the corresponding bit in the remote request wait register (RFPR) is set. Interrupts can be sent to the CPU according to the settings of the corresponding bit (MBIMR0 to MBIMR15) in the mailbox interrupt mask register (MBIMR) and the remote frame request interrupt mask (IRR2) in the interrupt mask register (IMR).

**Unread message overwrite:** If the receive message identifier matches the mailbox identifier, the receive message is stored in the mailbox regardless of whether the mailbox contains an unread message or not. If a message overwrite occurs, the corresponding bit (UMSR0 to UMSR15) in the unread message register (UMSR) is set. In overwriting an unread message, the unread message register (UMSR) is set when a new message is received before the corresponding bit in the receive complete register (RXPR) has been cleared. If the unread interrupt flag (IRR9) in the interrupt mask register (IMR) is set to enable interrupts at this time, an interrupt can be sent to the CPU. Figure 15.12 shows a flowchart for unread message overwriting.

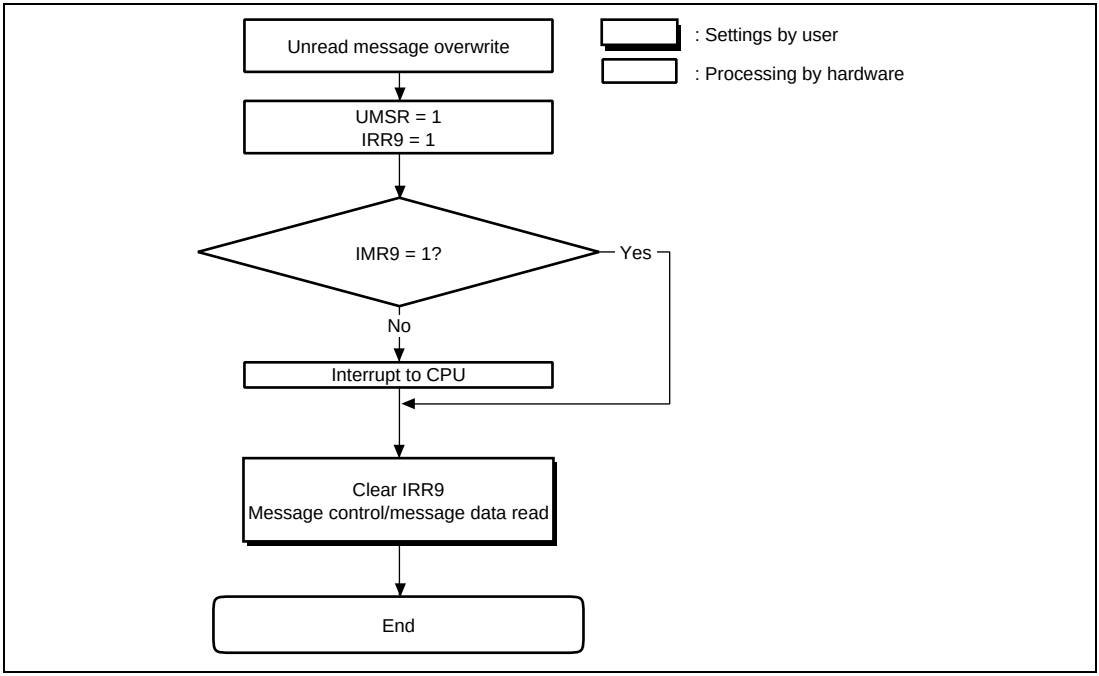



**Figure 15.12 Unread Message Overwrite Flowchart**

### 15.4.5 HCAN Sleep Mode

The HCAN is provided with an HCAN sleep mode that places the HCAN module in the sleep state in order to reduce current consumption. Figure 15.13 shows a flowchart of the HCAN sleep mode.

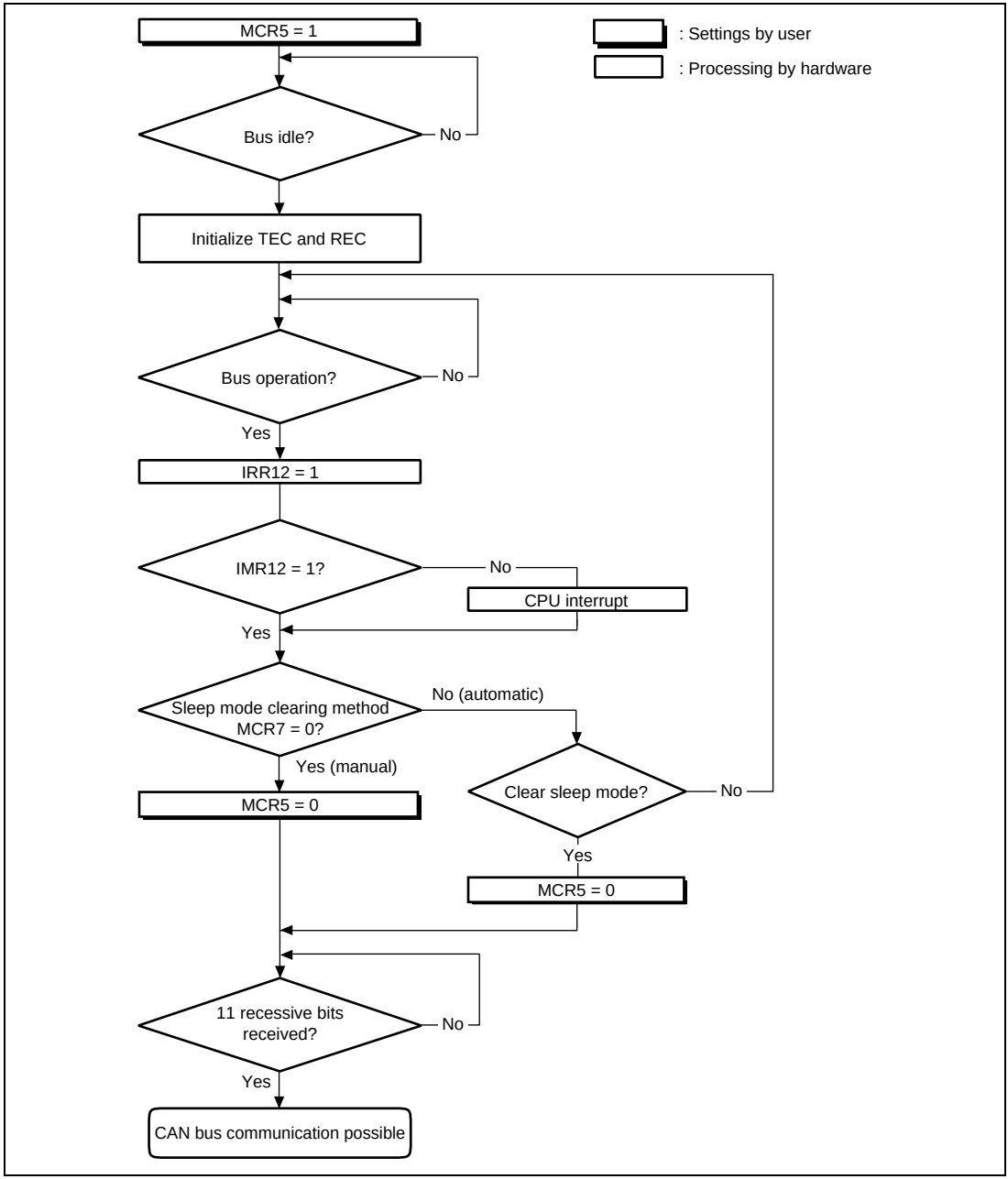



**Figure 15.13 HCAN Sleep Mode Flowchart**

HCAN sleep mode is entered by setting the HCAN sleep mode bit (MCR5) to 1 in the master control register (MCR). If the CAN bus is operating, the transition to HCAN sleep mode is delayed until the bus becomes idle.

Either of the following methods of clearing HCAN sleep mode can be selected:

- Clearing by software
- Clearing by CAN bus operation

In order to re-enter CAN bus communication enabled state, eleven recessive bits must be received after HCAN sleep mode was cleared.

**Clearing by software:** HCAN sleep mode is cleared by writing a 0 to MCR5 from the CPU.

**Clearing by CAN bus operation:** The cancellation method is selected by the MCR7 bit setting in MCR. Clearing by CAN bus operation occurs automatically when the CAN bus performs an operation and this change is detected. In this case, the first message is not stored in a mailbox; messages will be received normally from the second message onward. When a change is detected on the CAN bus in HCAN sleep mode, the bus operation interrupt flag (IRR12) is set in the interrupt register (IRR). If the bus interrupt mask (IMR12) in the interrupt mask register (IMR) is set to enable interrupts at this time, an interrupt can be sent to the CPU.

### 15.4.6 HCAN Halt Mode

The HCAN halt mode is provided to enable mailbox settings to be changed without performing an HCAN hardware or software reset. Figure 15.14 shows a flowchart of the HCAN halt mode.

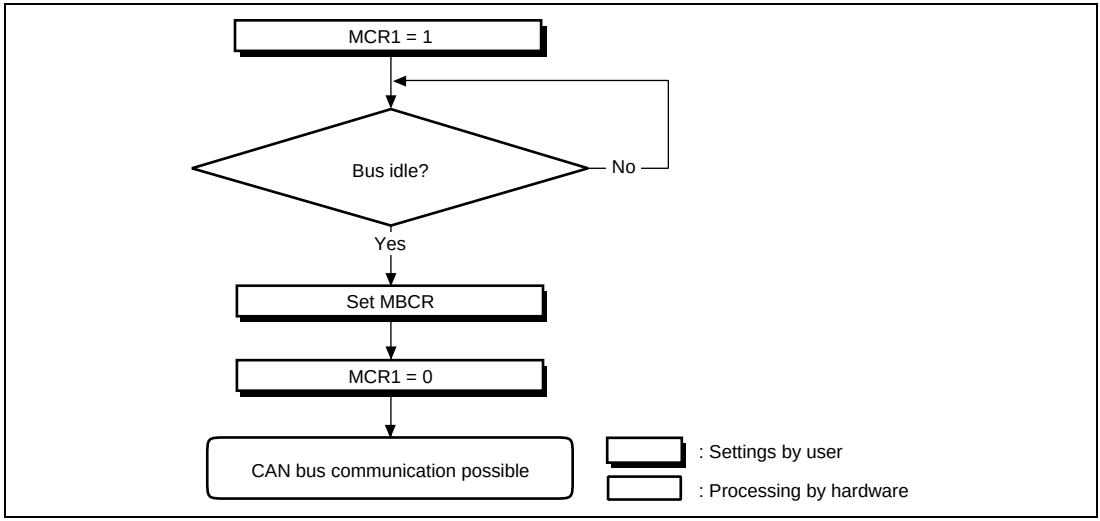



**Figure 15.14 HCAN Halt Mode Flowchart**

HCAN halt mode is entered by setting the halt request bit (MCR1) to 1 in the master control register (MCR). If the CAN bus is operating, the transition to HCAN halt mode is delayed until the bus becomes idle.

HCAN halt mode is cleared by clearing MCR1 to 0.

## 15.5 Interrupt Sources

Table 15.4 lists the HCAN interrupt sources. These sources can be masked except the reset processing interrupt by power-on reset (IRR0). Masking is implemented using the mailbox interrupt mask register (MBIMR), interrupt mask register (IMR), and IRQ enable register (IER). For details on the interrupt vector of each interrupt source, refer to section 5, Interrupt Controller.

**Table 15.4 HCAN Interrupt Sources**

| Name | Description | Interrupt Flag | DTC Activation |
|---|---|---|---|
| ERS0/OVR0 | Error passive interrupt (TEC ≥ 128 or REC ≥ 128) | IRR5 | Not possible |
| | Bus off interrupt (TEC ≥ 256) | IRR6 | |
| | Reset processing interrupt by power-on reset | IRR0 | |
| | Remote frame reception | IRR2 | |
| | Error warning interrupt (TEC ≥ 96) | IRR3 | |
| | Error warning interrupt (REC ≥ 96) | IRR4 | |
| | Overload frame transmission interrupt | IRR7 | |
| | Unread message overwrite | IRR9 | |
| | Detection of CAN bus operation in HCAN sleep mode | IRR12 | |
| RM0 | Mailbox 0 message reception | IRR1 | Possible |
| RM1 | Mailbox 1-15 message reception | IRR1 | Not possible |
| SLE0 | Message transmission/cancellation | IRR8 | Not possible |
| IRQ2 | Setting the RxDIE bit in HCANMON to 1 generates an IRQ2 interrupt caused by an HRxD input signal. | IRQ2F | Possible |

## 15.6 DTC Interface

The DTC can be activated by the reception of a message in HCAN mailbox 0. When the DTC activation is set and DTC transfer ends, the RXPR0 and RFPR0 flags are automatically cleared. An interrupt request is not sent to the CPU by a reception interrupt from the HCAN. Figure 15.15 shows a DTC transfer flowchart.

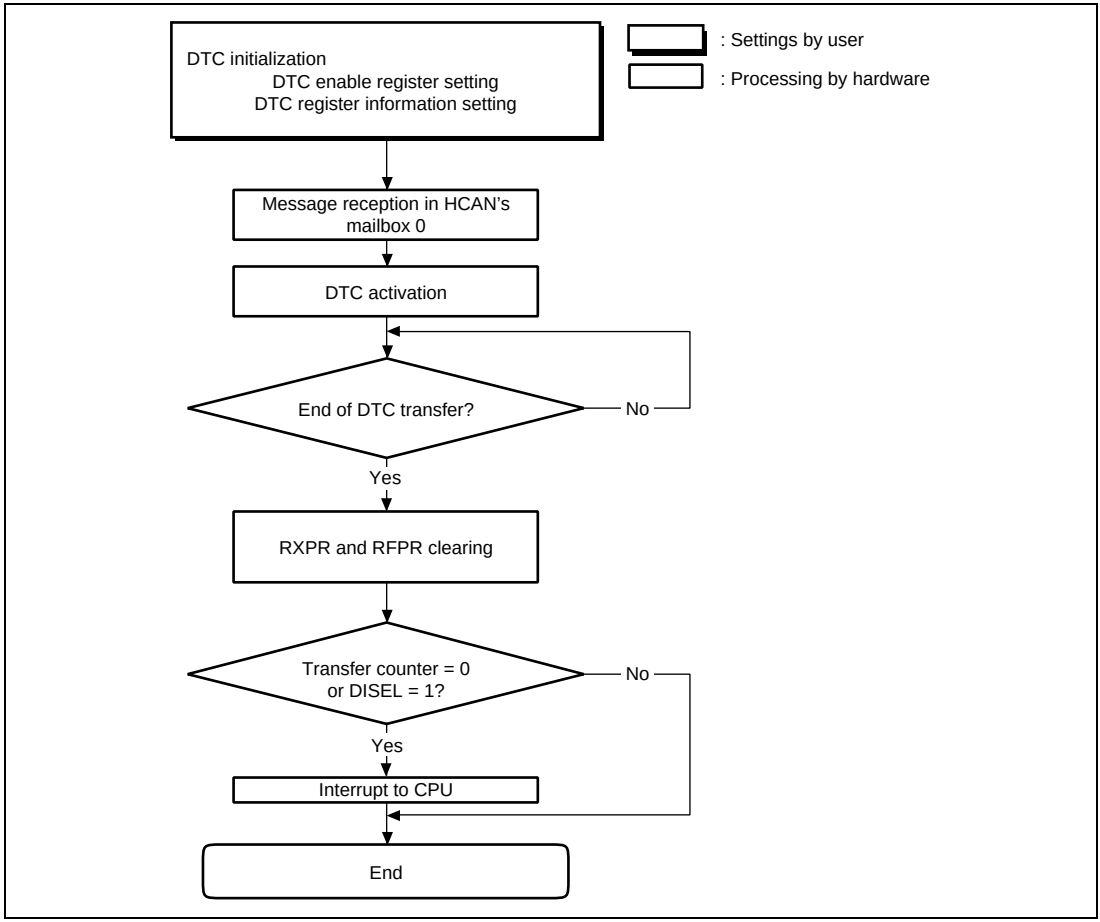



**Figure 15.15 DTC Transfer Flowchart**

## 15.7 CAN Bus Interface

A bus transceiver IC is necessary to connect the H8S/2628 Series to a CAN bus. A Philips PCA82C250 transceiver IC is recommended. If any other product is used, confirm that it is compatible with the PCA82C250. Figure 15.16 shows a sample connection diagram.

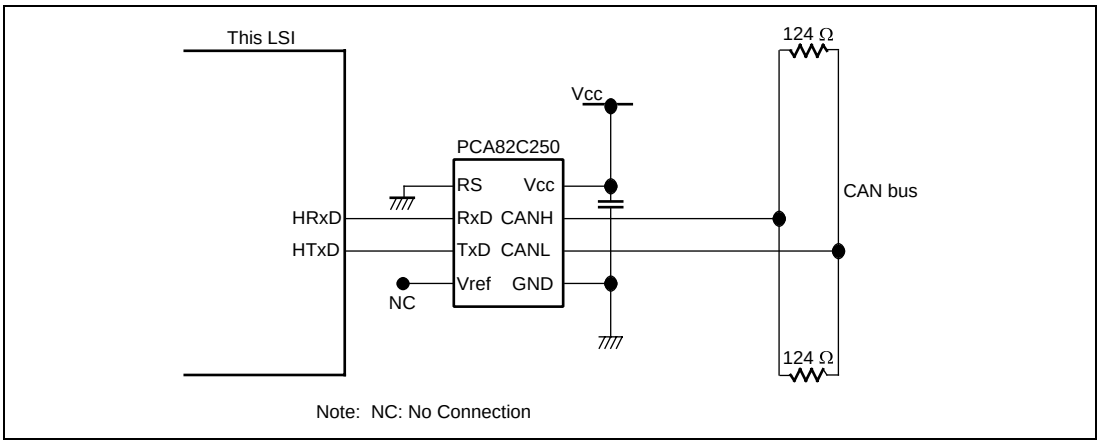



**Figure 15.16 High-Speed Interface Using PCA82C250**

## 15.8 Usage Notes

### 15.8.1 Module Stop Mode Setting

HCAN operation can be disabled or enabled using the module stop control register. The HCAN operation is set to be halted initially. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 15.8.2 Reset

The HCAN is reset by a power-on reset, in hardware standby mode, and in software standby mode. All the registers are initialized by a reset, however mailboxes (message control (MCx[x])/message data (MDx[x])) are not initialized. Mailboxes (message control (MCx[x])/message data (MDx[x])) are initialized after power-on and at this time, their initial values are undefined. Therefore, always initialize mailboxes after a power-on reset, a transition to hardware standby mode, or software standby mode. After a power-on reset or recovery from software standby mode, the reset interrupt flag (IRR0) is automatically set. Since this bit cannot be masked in the interrupt mask register (IMR), an HCAN interrupt will be initiated immediately after an HCAN interrupt is enabled by the interrupt controller without clearing the flag. IRR0 should therefore be cleared at initialization.

### 15.8.3 HCAN Sleep Mode

The bus operation interrupt flag (IRR12) in the interrupt register (IRR) is set by CAN bus operation in HCAN sleep mode. Therefore, this flag is not used by the HCAN to indicate sleep mode release. Note that the reset status bit (GSR3) in the general status register (GSR) is set even in sleep mode.

### 15.8.4 Interrupts

When the mailbox interrupt mask register (MBIMR) is set, the interrupt registers (IRR8, 2, 1) are not set by reception completion, transmission completion, or transmission cancellation of the set mailboxes.

### 15.8.5 Error Counters

In the case of error active and error passive, REC and TEC perform count up and down normally. In the bus-off state, 11-bit recessive sequences are counted (REC + 1) using REC. When REC reaches 96 during the count, IRR4 and GSR1 are set.

### 15.8.6 Register Access

Byte or word access can be performed for all HCAN registers. Longword access should be avoided.

### 15.8.7 HCAN Medium-Speed Mode

In medium-speed mode, the HCAN registers cannot be read/written.

### 15.8.8 Register Hold in Standby Modes

All HCAN registers are initialized in hardware standby mode and software standby mode.

### 15.8.9 Use on Bit Manipulation Instructions

Since the HCAN status flag is cleared by writing 1, do not use the bit manipulation instructions to clear the flag. To clear the flag, use the MOV instructions and write 1 only to the bit to be cleared.

### 15.8.10 HCAN TXCR Operation

1. When the transmit wait cancel register (TXCR) is used to cancel a transmit wait message in a transmit wait mailbox, the corresponding bit to TXCR and the transmit wait register (TXPR) may not be cleared even if transmission is canceled. This occurs when the following conditions are all satisfied.

- The HRxD pin is stacked to 1 because of a CAN bus error, etc.
- There is at least one mailbox waiting for transmission or being transmitted.
- The message transmission in a mailbox being transmitted is canceled by TXCR.

  If this occurs, transmission is canceled. However, since TXPR and TXCR states are indicated wrongly that a message is being cancelled, transmission cannot be restarted even if the stack state of the HRxD pin is canceled and the CAN bus recovers the normal state. If there are at least two transmission messages, a message which is not being transmitted is canceled and a message being transmitted retains its state.

To avoid this, one of the following countermeasures must be executed.

- Transmission must not be canceled by TXCR. When transmission is normally completed after the CAN bus has recovered, TXPR is cleared and the HCAN recovers the normal state.
- To cancel transmission, the corresponding bit to TXCR must be written to 1 continuously until the bit becomes 0. TXPR and TXCR are cleared and the HCAN recovers the normal state.

2. When the bus-off state is entered while TXPR is set and the transmit wait state is entered, the internal state machine does not operate even if TXCR is set during the bus-off state. Therefore transmission cannot be canceled. The message can be canceled when one message is transmitted or a transmission error occurs after the bus-off state is recovered. To clear a message after the bus-off state is recovered, the following countermeasure must be executed.

- A transmit wait message must be cleared by resetting the HCAN during the bus-off period.

  To reset the HCAN, the module stop bit (MSTPC3 in MSTPCRC) must be set or cleared. In this case, the HCAN is entirely reset. Therefore the initial settings must be made again.

# Section 16 Synchronous Serial Communication Unit (SSU)

This LSI has two independent synchronous serial communication unit (SSU) channels. The SSU has master mode in which this LSI outputs clocks as a master device for synchronous serial communication and slave mode in which clocks are input from an external device for synchronous serial communication. Synchronous serial communication can be performed with devices having different clock polarity and clock phase. In addition, synchronous serial communication can also be performed between multiple processors (multi-processor communication). Figure 16.1 is a block diagram of the SSU.

## 16.1 Features

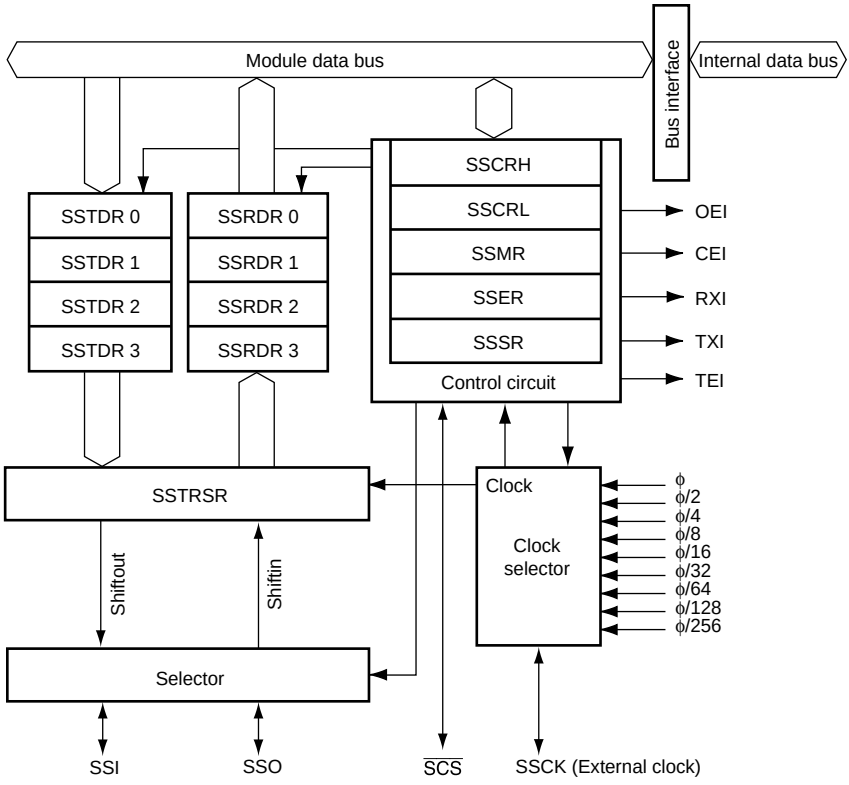
- Choice of master mode and slave mode
- Choice of standard mode and bidirectional mode
- Synchronous serial communication with devices with different clock polarity and clock phase
- Choice of 8/16/32-bit width of transmit/receive data
- Full-duplex communication capability

  The shift register is incorporated, enabling transmission and reception to be executed simultaneously.
- Continuous serial communication
- Choice of LSB-first or MSB-first transfer
- Choice of a clock source

  $\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, $\phi/32$, $\phi/64$, $\phi/128$, $\phi/256$, or external clock
- Five interrupt sources

  transmit-end, transmit-data-register-empty, receive-data-full, overrun-error, and conflict error
- Module stop mode can be set

Figure 16.1 shows a block diagram of the SSU.




[Legend]

| | |
|---|---|
| SSCRH | : SS control register H |
| SSCRL | : SS control register L |
| SSMR | : SS mode register |
| SSER | : SS enable register |
| SSSR | : SS status register |
| SSTDR0 to SSTDR3 | : SS transmit data register |
| SSRDR0 to SSRDR3 | : SS receive data register |
| SSTRSR | : SS transmit/recive shift register |

**Figure 16.1 Block Diagram of SSU**

## 16.2 Input/Output Pins

Table 16.1 shows the SSU pin configuration.

**Table 16.1 Pin Configuration**

| Name | Symbol | I/O | Function |
|---|---|---|---|
| SSU clock | SSCK | I/O | SSU clock input/output |
| SSU receive data input | SSI | I/O | SSU receive data input/output |
| SSU transmit data output | SSO | I/O | SSU transmit data input/output |
| SSU chip select input/output | $\overline{\text{SCS}}$ | I/O | SSU chip select input/output |

## 16.3 Register Descriptions

The SSU has the following registers.

- SS control register H (SSCRH)
- SS control register L (SSCRL)
- SS mode register (SSMR)
- SS enable register (SSER)
- SS status register (SSSR)
- SS transmit data register 0 to 3 (SSTDR0 to SSTDR3)
- SS receive data register 0 to 3 (SSRDR0 to SSRDR3)

### 16.3.1 SS Control Register H (SSCRH)

SSCRH specifies master/slave device selection, bidirectional mode enable, SSO pin output value selection, SSCK pin selection, and $\overline{\text{SCS}}$ pin selection.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | MSS | 0 | R/W | Master/Slave Device Selection<br>Selects that this module is used in master mode or slave mode. When master mode is selected, transfer clocks are output from the SSCK pin. When the CE bit in SSSR is set, this bit is automatically cleared.<br>0: Slave mode is selected.<br>1: Master mode is selected. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | BIDE | 0 | R/W | Bidirectional Mode Enable<br>Selects that both serial data input pin and output pin are used or one of them is used. However, transmission and reception are not performed simultaneously when bidirectional mode is selected. For details, section 16.5.3, Relationship between Data I/O Pins and Shift Register.<br>0: Standard mode (two pins are used as data input and output)<br>1: Bidirectional mode (one pin is used for data input and output) |
| 5 | — | 0 | — | Reserved<br>The write value should always be 0. |
| 4 | SOL | 0 | R/W | Serial Data Output Value Selection<br>The output level of serial data, which retains that of the last bit, can be modified by operating this bit before or after transmission. When modifying the output level, use the MOV instruction after clearing the SOLP bit to 0. Since writing to this bit during data transmission causes malfunctions, this bit should not be modified.<br>0: Serial data output is modified to low level.<br>1: Serial data output is modified to high level. |
| 3 | SOLP | 0 | R | SOL Bit Write Protect<br>When modifying the output level of serial data, use the MOV instruction after setting SOL to 1 and clearing SOLP to 0, or by clearing SOL and SOLP to 0.<br>0: Output level can be modified by the SOL value<br>1: This bit is always read as 1 and cannot be modified. |
| 2 | SCKS | 0 | R/W | SSCK Pin Selection<br>Selects that the SSCK pin functions as a port or a serial clock pin. When MSS = 1, this bit must be set to 1.<br>0: Functions as an I/O port.<br>1: Functions as a serial clock. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | CSS1 | 0 | R/W | $\overline{\text{SCS}}$ Pin Selection |
| 0 | CSS0 | 0 | R/W | Select that the $\overline{\text{SCS}}$ pin functions as a port or $\overline{\text{SCS}}$ input or output. However, when MSS = 0, the $\overline{\text{SCS}}$ pin functions as an input pin regardless of the CSS1 and CSS0 settings.<br>00: I/O port<br>01: Functions as $\overline{\text{SCS}}$ input<br>10: Functions as $\overline{\text{SCS}}$ automatic input/output (however, functions as $\overline{\text{SCS}}$ input before and after transfer and outputs a low level during transfer)<br>11: Functions as $\overline{\text{SCS}}$ automatic output (however, outputs a high level before and after transfer and outputs a low level during transfer) |

### 16.3.2 SS Control Register L (SSCRL)

SSCRL selects software reset and transmit/receive data width.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 5 | SRES | 0 | R/W | Software Reset<br>Setting this bit to 1 forcibly resets the SSU internal sequencer. After that, this bit is automatically cleared. The ORER, TEND, TDRE, RDRF, and CE bits in SSSR and the TE and RE bits in SSER are also initialized. Values of other bits for SSU registers are held.<br>To stop transfer, set this bit to 1 to reset the SSU internal sequencer. |
| 4 to 2 | — | All 0 | — | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 1 | DATS1 | 0 | R/W | Transmit/Receive Data Length Selection |
| 0 | DATS0 | 0 | R/W | Select serial data length from 8, 16, and 32 bits.<br>00: 8 bits<br>01: 16 bits<br>10: 32 bits<br>11: Setting invalid |

### 16.3.3 SS Mode Register (SSMR)

SSMR selects the MSB first/LSB first, clock phase, clock polarity, and clock rate of synchronous serial communication.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | MLS | 0 | R/W | MSB First/LSB First<br>Selects the serial data is transmitted in MSB first or LSB first.<br>0: LSB first<br>1: MSB first |
| 6 | CPOS | 0 | R/W | Clock Polarity Selection<br>Selects SSCK clock polarity.<br>0: High output in idle mode, and low output in active mode<br>1: Low output in idle mode, and high output in active mode |
| 5 | CPHS | 0 | R/W | Clock Phase Selection<br>Selects SSCK clock phase.<br>0: Data changes at the first edge.<br>1: Data is latched at the first edge. |
| 4, 3 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 2 | CKS2 | 0 | R/W | Transfer Clock Rate Selection |
| 1 | CKS1 | 0 | R/W | Select the transfer clock rate (prescaler division rate) when an internal clock is selected. |
| 0 | CKS0 | 0 | R/W | 000: $\phi$/2<br>001: $\phi$/4<br>010: $\phi$/8<br>011: $\phi$/16<br>100: $\phi$/32<br>101: $\phi$/64<br>110: $\phi$/128<br>111: $\phi$/256 |

### 16.3.4 SS Enable Register (SSER)

SSER performs transfer/receive control of synchronous serial communication and setting of interrupt enable.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TE | 0 | R/W | Transmit Enable<br>When this bit is set to 1, transmission is enabled. |
| 6 | RE | 0 | R/W | Receive Enable<br>When this bit is set to 1, reception is enabled. |
| 5, 4 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 3 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>When this bit is set to 1, TEI interrupt request is enabled. |
| 2 | TIE | 0 | R/W | Transmit Interrupt Enable<br>When this bit is set to 1, TXI interrupt request is enabled. |
| 1 | RIE | 0 | R/W | Receive Interrupt Enable<br>When this bit is set to 1, RXI interrupt request is enabled. |
| 0 | CEIE | 0 | R/W | Conflict Error Interrupt Enable<br>When this bit is set to 1, CEI interrupt request is enabled. |

### 16.3.5 SS Status Register (SSSR)

SSSR is a status flag register for interrupts.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved<br>The write value should always be 0. |
| 6 | ORER | 0 | R/W | Overrun Error<br>If the next data is received while RDRF = 1, an overrun error occurs, indicating abnormal termination. SSRDR stores 1-frame receive data before an overrun error occurs and loses data received later. While ORER = 1, continuous serial reception cannot be continued. Serial transmission cannot be continued, either.<br>[Setting condition]<br>• When the next reception data is transferred to SSRDR while RDRF = 1<br>[Clearing condition]<br>• When 0 is written to ORER after reading ORER = 1 |
| 5, 4 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 3 | TEND | 1 | R | Transmit End<br>[Setting condition]<br>• When the last bit of transmit data is transmitted with TDRE = 1<br>[Clearing conditions]<br>• When 0 is written to the TEND bit after reading TEND = 1<br>• When data is written to SSTDR<br>• When data is transferred by the DTC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | TDRE | 1 | R/W | Transmit Data Empty<br>Indicates whether or not SSTDR contains transmit data.<br>[Setting conditions]<br>• When the TE bit in SSER is 0<br>• When data is transferred from SSTDR to SSTRSR and SSTDR is ready to be written to.<br>[Clearing conditions]<br>• When 0 is written to the TDRE bit after reading TDRE = 1<br>• When data is written to SSTDR with TE = 1<br>• When data is transferred by the DTC |
| 1 | RDRF | 0 | R/W | Receive Data Register Full<br>Indicates whether or not SSRDR contains received data.<br>[Setting condition]<br>• When receive data is transferred from SSTRSR to SSRDR after successful data reception<br>[Clearing conditions]<br>• When 0 is written to RDRF after reading RDRF = 1<br>• When received data is read from SSRDR<br>• When data is transferred by the DTC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | CE | 0 | R/W | Conflict/Incomplete Error<br>Indicates that a conflict error has occurred when 0 is externally input via the $\overline{SCS}$ pin with MSS = 1.<br>If the $\overline{SCS}$ pin level changes to 1 during slave operation, an incomplete error occurs because it is determined that a master device has terminated the transfer. Data reception does not continue while the CE bit is set to 1. Reset the SSU internal sequencer by setting the SRES bit in SSCRL to 1 before resuming transfer after incomplete error.<br>[Setting condition]<br>• When a low level is input to the $\overline{SCS}$ pin in master device mode (MSS in SSCRH = 1)<br>• When a 1 is input to the $\overline{SCS}$ pin during slave device mode (MSS in SSCRH = 0) transfer<br>[Clearing condition]<br>• When 0 is written to the CE bit after reading CE = 1 |

### 16.3.6 SS Transmit Data Register 0 to 3 (SSTDR0 to SSTDR3)

SSTDR is an 8-bit register that stores transmit data. When 8-bit data length is selected by bits DATS1 and DATS0 in SSCRL, SSTDR0 is valid. When 16-bit data length is selected, SSTDR0 and SSTDR1 are valid. When 32-bit data length is selected, SSTDR0 to SSTDR3 are valid.

When the SSU detects that SSTRSR is empty, it transfers the transmit data written in SSTDR to SSTRSR and starts transmission. If the next transmit data has already been written to SSTDR during serial transmission, the SSU transfers the written data to SSTRSR to continue transmission. Although SSTDR can be read or written to by the CPU and DTC at all times, to achieve reliable serial transmission, write transmit data to SSTDR after confirming that the TDRE bit in SSSR is set to 1. The initial value of this register is H′00.

### 16.3.7 SS Receive Data Register 0 to 3 (SSRDR0 to SSRDR3)

SSRDR is an 8-bit register that stores receive data. When 8-bit data length is selected by bits DATS1 and DATS0 in SSCRL, SSRDR0 is valid. When 16-bit data length is selected, SSRDR0 and SSRDR1 are valid. When 32-bit data length is selected, SSRDR0 to SSRDR3 are valid.

When the SSU has received 1-byte data, it transfers the received serial data from SSTRSR to SSRDR where it is stored. After this, SSTRSR is receive-enabled. Since SSTRSR and SSRDR function as a double buffer in this way, continuous receive operations can be performed. Read SSRDR after confirming that the RDRF bit in SSSR is set to 1. SSRDR cannot be written to by the CPU. The initial value of this register is H′00.

### 16.3.8 SS Shift Register (SSTRSR)

SSTRSR is a shift register that transmits and receives serial data.

When data from SSTDR to SSTRSR is transferred with MLS = 0, bit 0 of transmit data is bit 0 in the SSTDR contents (LSB first communication). When data from SSTDR to SSTRSR is transferred with MLS = 1, bit 0 of transmit data is bit 7 in the SSTDR contents (MSB first communication). To perform serial data transmission, the SSU transfers data starting from LSB (bit 0) in SSTRSR to the SSO pin.

In reception, the SSU sets serial data that has been input from the SSI pin to SSTRSR starting from LSB (bit 0) and converts it into parallel data. When 1-byte data has been received, the SSTRSR contents are automatically transferred to SSRDR. SSTRSR cannot be directly accessed by the CPU.

## 16.4 Operation

### 16.4.1 Transfer Clock

A transfer clock can be selected from eight internal clocks and an external clock. When using this module, set SCKS in SSCRH to 1 to select the SSCK pin as a serial clock. When MSS in SSCRH is 1, an internal clock is selected and the SSCK pin is used as an output pin. When transfer is started, the clock with the transfer rate set by bits CKS2 to CKS0 in SSMR is output from the SSCK pin. When MSS = 0, an external clock is selected and the SSCK pin is used as an input pin.

### 16.4.2 Relationship of Clock Phase, Polarity, and Data

The relationship of clock phase, polarity, and transfer data depends on the combination of CPOS and CPHS in SSMR. Figure 16.2 shows the relationship.

Setting the MLS bit specifies that MSB or LSB first communication. When MLS = 0, data is transferred from the LSB to MSB. When MLS = 1, data is transferred from the MSB to LSB.

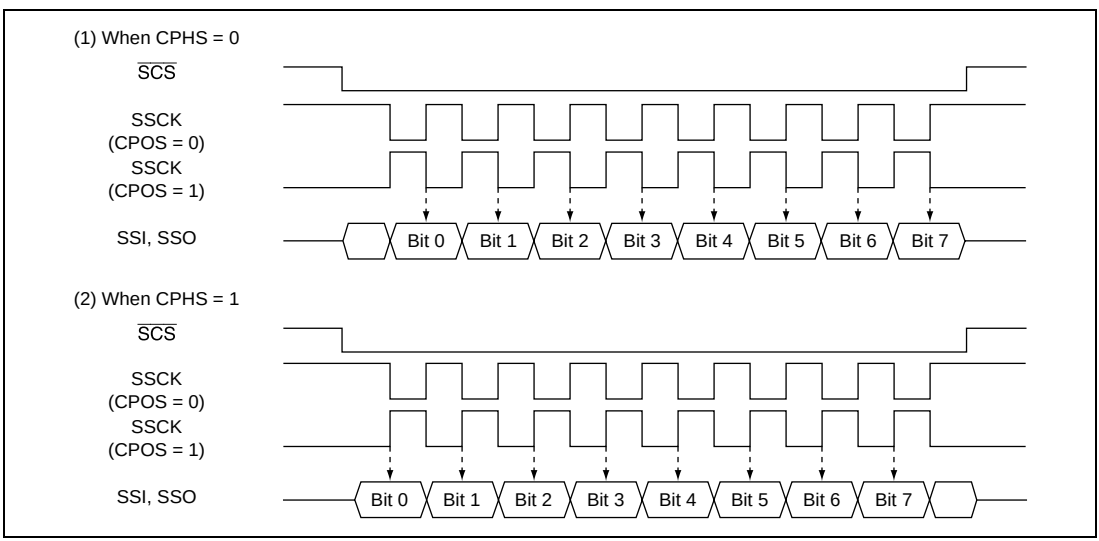



**Figure 16.2 Relationship of Clock Phase, Polarity, and Data**

### 16.4.3 Relationship between Data I/O Pins and the Shift Register

The connection between data I/O pins and the shift register (SSTRSR) depends on the combination of the MSS and BIDE bits in SSCRH. The data input pin and data output pin can be set independently by the values of TE and RE.

The SSU transmits serial data from the SSO pin and receives serial data from the SSI pin when operating with BIDE = 0 and MSS = 1 (standard, master mode) (see figure 16.3 (1)). The SSU transmits serial data from the SSI pin and receives serial data from the SSO pin when operating with BIDE = 0 and MSS = 0 (standard, slave mode) (see figure 16.3 (2)).

The SSU transmits and receives serial data from the SSO pin regardless of master or slave mode when operating with BIDE = 1 (bidirectional mode) (see figure 16.3 (3) and (4)).

However, even if both the TE and RE bits are set to 1, transmission and reception are not performed simultaneously. Either the TE or RE bit must be selected.

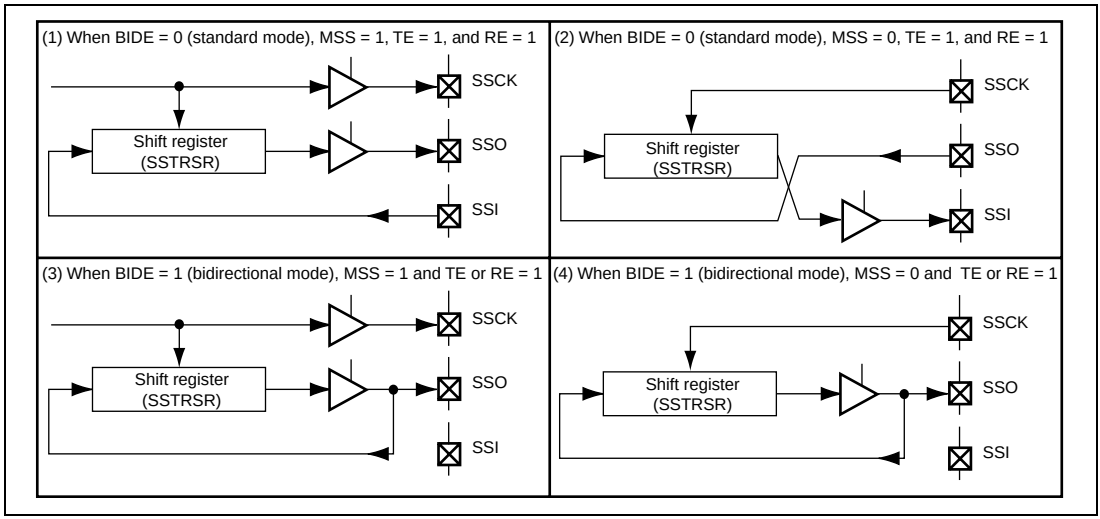



**Figure 16.3 Relationship between Data I/O Pins and the Shift Register**

### 16.4.4 Data Transmission and Data Reception

The SSU performs data communications using the bus with four lines: the clock line (SSCK), data input (SSI or SSO), data output (SSI or SSO), and chip select ($\overline{\text{SCS}}$).

The SSU also supports bidirectional mode in which the data is output and input using one pin.

- SSU Initialization

Figure 16.4 shows an example of the SSU initialization. Before transmitting and receiving data, first clear the TE and RE bits in SSER to 0, then initialize the SSU.

Note: When the operating mode or transfer format is changed for example, the TE and RE bits must be cleared to 0. When the TE bit is cleared to 0, the TDRE bit is set to 1. Note that clearing the RE bit to 0 does not initialize the values of the RDRF and ORER bits or the contents of SSRDR.

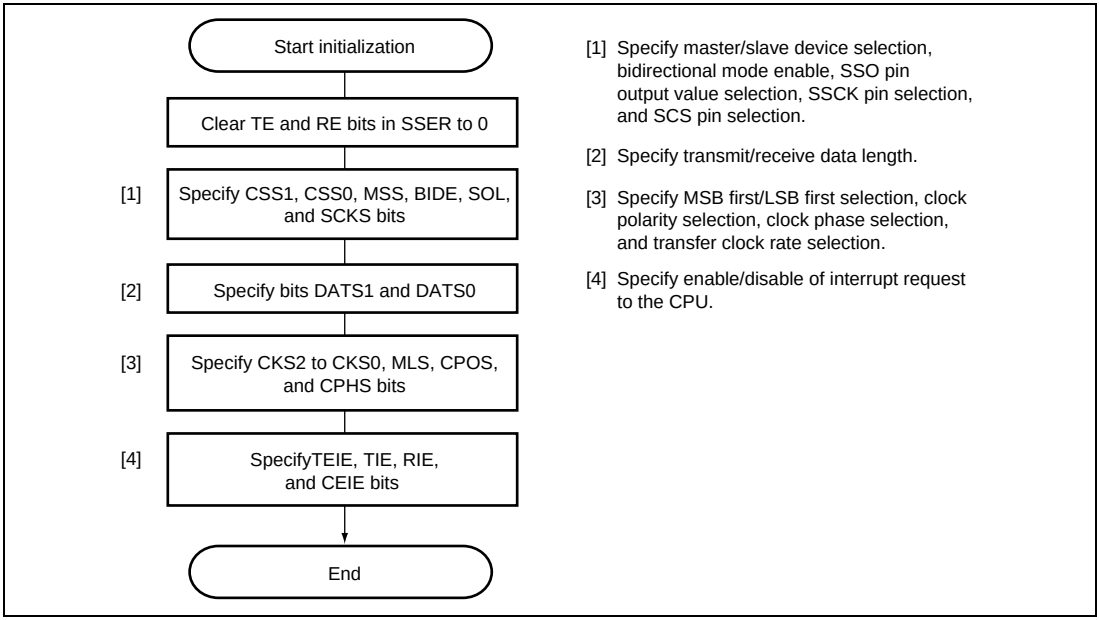



**Figure 16.4 Example of SSU Initialization**

- Data Transmission

Figure 16.5 shows an example of transmission operation, and figure 16.6 shows an example of data transmission flowchart.

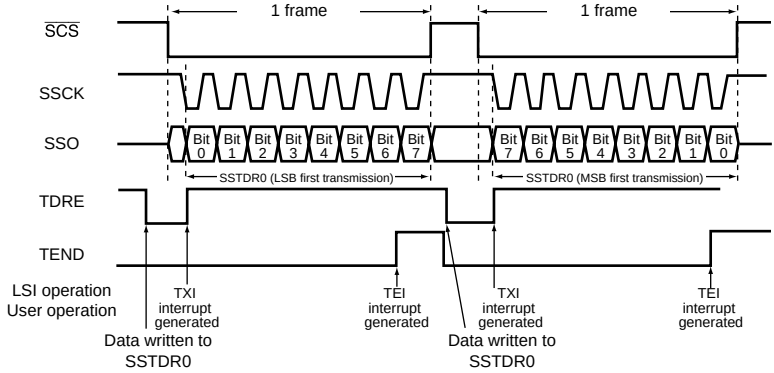
When transmitting data, the SSU operates as shown below.
In master device mode, the SSU outputs a transfer clock and data. In slave device mode, when a low level signal is input to the $\overline{\text{SCS}}$ pin and a transfer clock is input to the SSCK pin, the SSU outputs data in synchronization with the transfer clock.

Writing transmit data to SSTDR after the TE bit is set to 1 clears the TDRE bit to 0, and the SSTDR contents is transferred to SSTRSR. After that, the SSU sets the TDRE bit to 1 and starts transmission. At this time, if the TIE bit in SSER is set to 1, a TXI interrupt is generated.

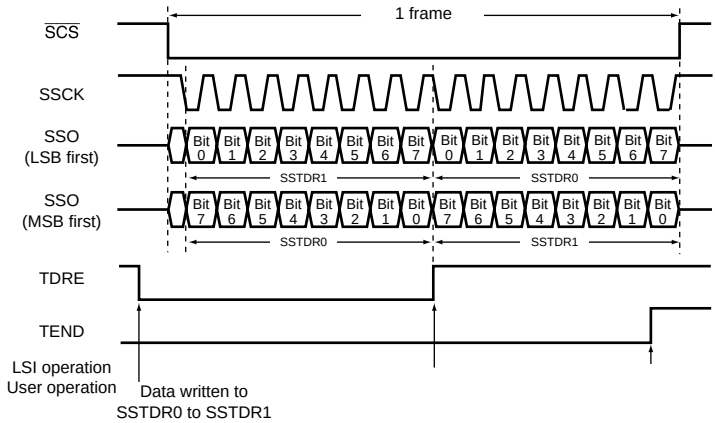
When 1-frame data has been transferred with the TDRE bit cleared to 0, the SSTDR contents are transferred to SSTRSR to start the next transmission. When the 8th bit of transmit data has been transferred with the TDRE bit set to 1, the TEND bit in SSSR is set to 1 and the state is retained. At this time, if the TEIE bit is set to 1, a TEI interrupt is generated. After transmission, the output level of the SSCK pin is fixed at a high level when CPOS = 0 and at a low level when CPOS = 1.

While the ORER bit in SSSR is set to 1, transmission is not performed. Check that the ORER bit is cleared to 0.

(1) When 8-bit data length is selected (SSTDR0 is valid) with CPOS = 0 and CPHS = 0




(2) When 16-bit data length is selected (SSTDR0 and SSTDR1 are valid) with CPOS = 0 and CPHS = 0




(3) When 32-bit data length is selected (SSTDR0 and SSTDR3 are valid) with CPOS = 0 and CPHS = 0

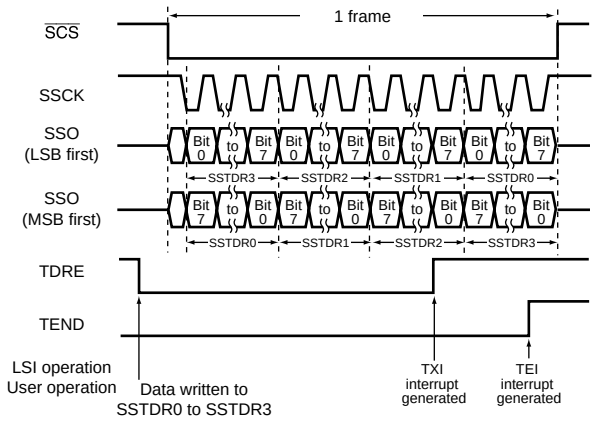



**Figure 16.5 Example of Transmission Operation**

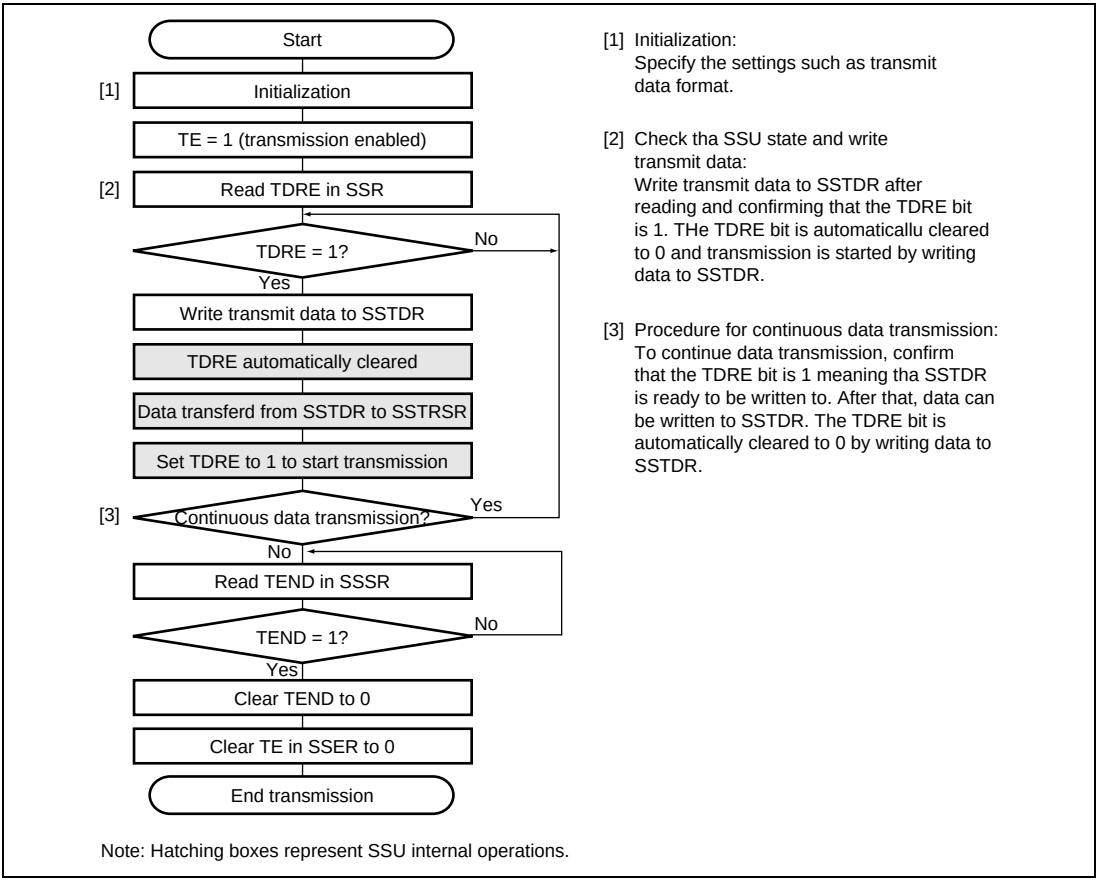



**Figure 16.6 Example of Data Transmission Flowchart**

- Data Reception

Figure 16.7 shows an example of reception operation, and figure 16.8 shows an example of data reception flowchart.

When receiving data, the SSU operates as shown below.

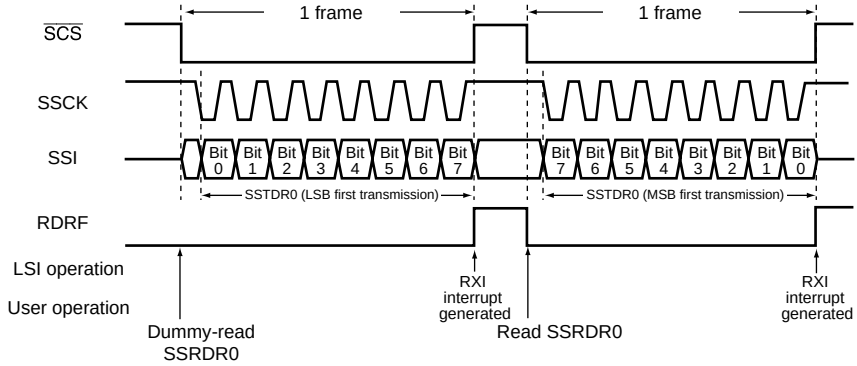
After the SSU sets the RE bit to 1 and dummy-reads SSRDR, data reception is started.

In master device mode, the SSU outputs a transfer clock and receives data. In slave device mode, when a low level signal is input to the $\overline{\text{SCS}}$ pin and a transfer clock is input to the SSCK pin, the SSU receives data in synchronization with the transfer clock.

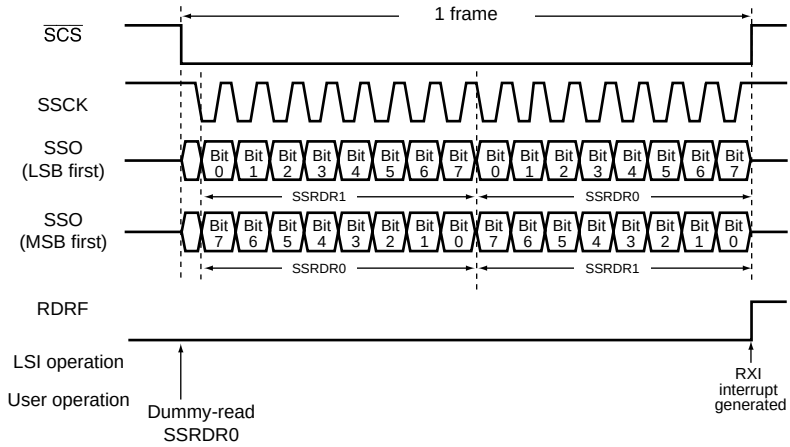
When 1-frame data has been received, the received data is stored in SSRDR. At this time, if the RIE bit is set to 1, an RXI interrupt is generated. The RDRF bit is automatically cleared to 0 by reading SSRDR.

When the RDRF bit has been set to 1 at the 8th rising edge of the transfer clock, the ORER bit in SSSR is set to 1. This indicates that an overrun error (OEI) has occurred. At this time, data reception is stopped. While the ORER bit in SSSR is set to 1, reception is not performed. To resume the reception, clear the ORER bit to 0.

(1) When 8-bit data length is selected (SSRDR0 is valid) with CPOS = 0 and CPHS 0




(2) When 16-bit data length is selected (SSRDR0 and SSRDR1 are valid) with CPOS = 0 and CPHS 0




(3) When 32-bit data length is selected (SSRDR0 and SSRDR3 are valid) with CPOS = 0 and CPHS 0

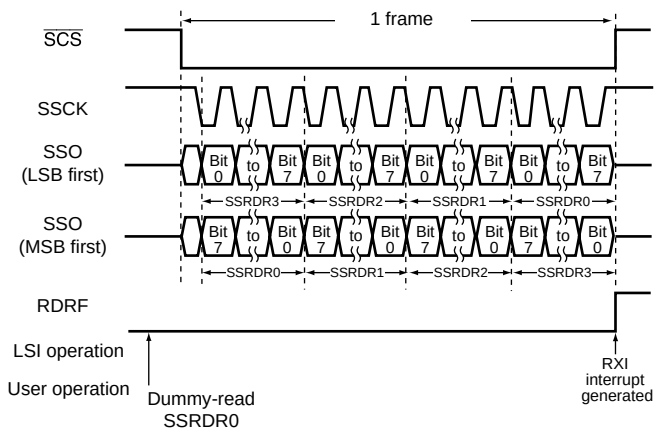
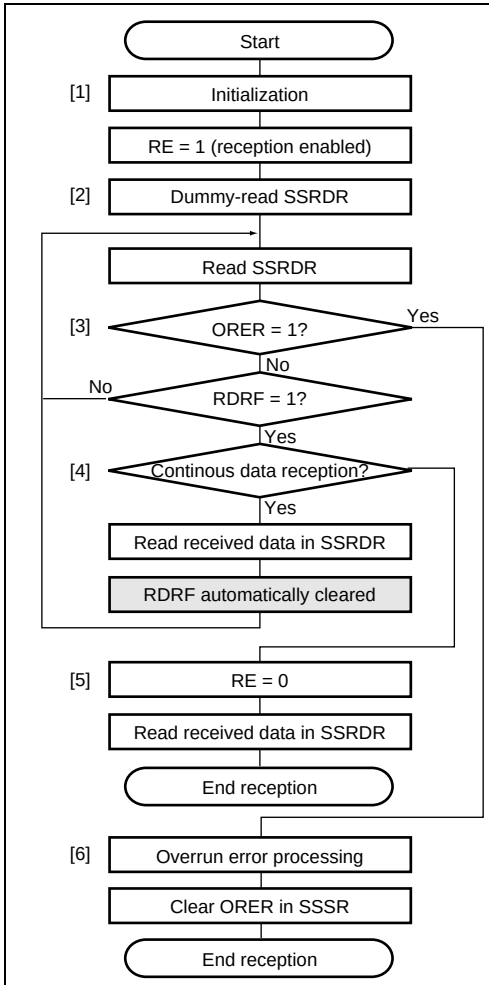



**Figure 16.7 Example of Reception Operation**




[1] Initialization:
Specify the settings such as receive data format.

[2] Start reception:
When SSRDR is dummy-read with RE = 1, reception is started.

[3], [6] Receive error processing:
When a receive error occurs execute the designated error processing after reading the ORER bit in SSSR. After that, Clear the ORER bit to 0. While the ORER bit is set to 1, transmission or reception is not resumed.

[4] To continue single reception:
When continueing single reception, the next single reception starts after reading received data in SSRDR.

[5] To complete reception:
To complete reception, read received data after clearing the RE bit to 0. When reading SSRDR without clearing the RE bit, reception is resumed.

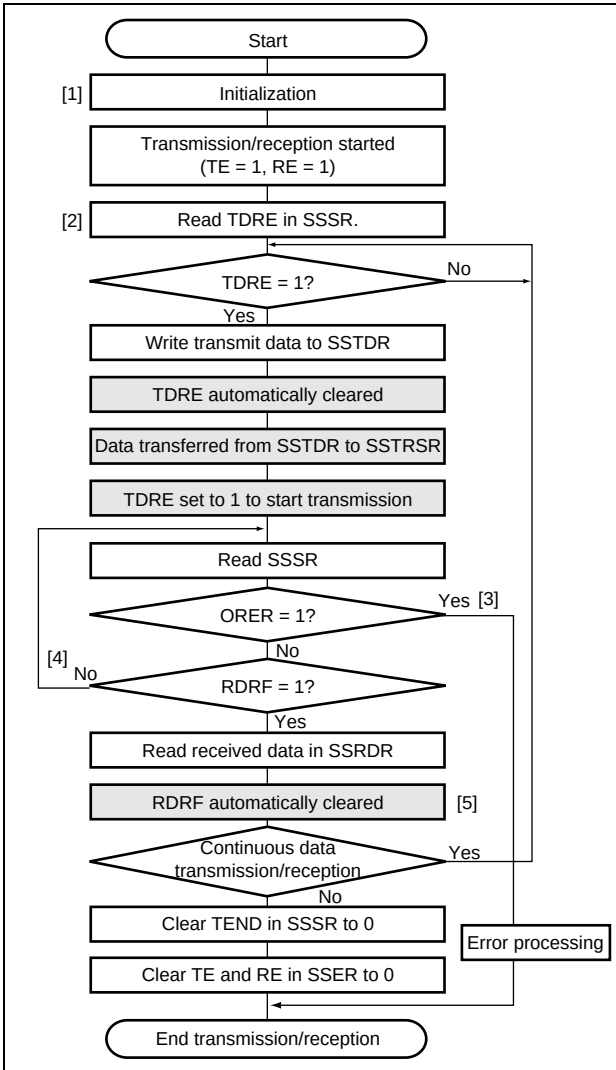
Note: Hatching boxes represent SSU internal operations.

**Figure 16.8 Example of Data Reception Flowchart**

- Data Transmission/Reception

Figure 16.9 shows an example of simultaneous transmission/reception operation. The data transmission/reception is performed combining the data transmission and data reception as mentioned above. The data transmission/reception is started by writing transmit data to SSTDR with TE = RE = 1.

Before switching transmission mode (TE = 1) or reception mode (RE = 1) to transmission/reception mode (TE = RE = 1), clear the TE and RE bits to 0. When starting the transfer, confirm that the TEND, RDRF, and ORER bits are cleared to 0 before setting the TE and RE bits to 1.




Note: Hatching boxes represent SSU internal operations.

[1] Initialization:
Specify the settings such as transmit/receive data format

[2] Check the SSU state and write transmit data:
Write transmit data to SSTDR after reading and confirming that the TDRE bit is 1. The TDRE bit is automatically cleared to 0 and transmission is started by writing data to SSTDR.

[3] Receive error processing:
When a receive error occurs, execute the designated error processing after reading the ORER bit in SSSR. After that, clear the ORER bit to 0. While the ORER bit is set to 1, transmission or reception is not resumed.

[4] Check the SSU state and read receibe data:
Read receibe data in SSRDR after reading and confirming that the RDRF bit is 1. The RDRF bit is cleared. A change of the RDRF bit (from 0 to 1) canbe notified by RXI interrupt.

[5] Procedure for continuous data transmission/ reception:
To continue setial data transmission/reception, confirm that the TDRE bit 1meaning that SSTDR is ready to be written to. After that, data can be written to SSTDR. The TDRE bit is automatically cleared to 0 by writing data to SSTDR.

**Figure 16.9 Example of Simultaneous Transmission/Reception Flowchart**

### 16.4.5 $\overline{\text{SCS}}$ Pin Control and Arbitration

When bits CSS1 and CSS0 in SSCRH are specified to B′10, the $\overline{\text{SCS}}$ pin functions as an input to detect arbitration. The arbitration detection period starts when setting the MSS bit in SSCRH to 1 and ends when starting serial transfer. When a low level signal is input to the $\overline{\text{SCS}}$ pin within the period, a conflict error occurs. At this time, the CE bit in SSSR is set to 1 and the MSS bit is cleared to 0.

Note: While the CE bit is set to 1, transmission or reception is not resumed. Clear the CE bit to 0 before resuming the transmission or reception.

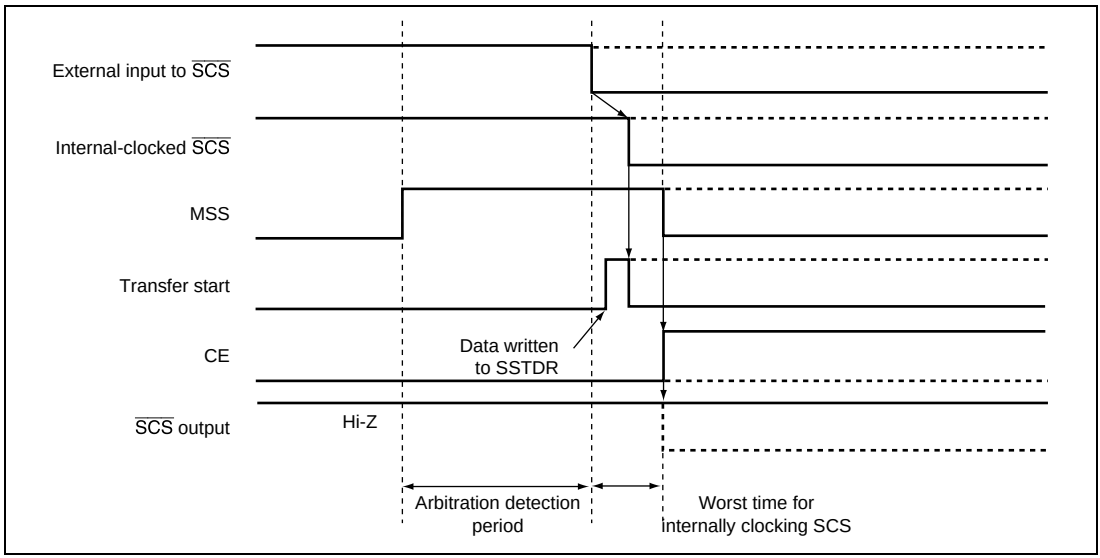



**Figure 16.10 Arbitration Detection Timing (Before Transfer)**

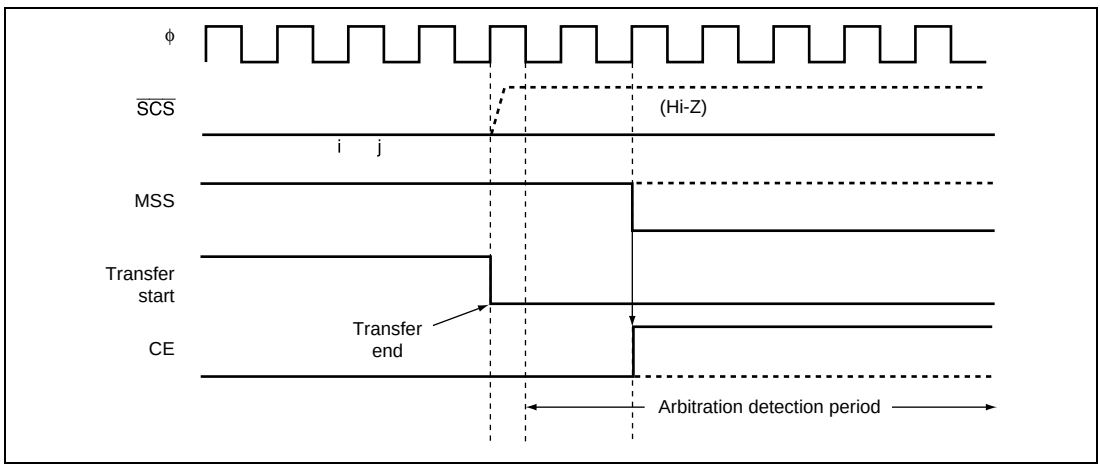



**Figure 16.11 Arbitration Detection Timing (After Transfer End)**

## 16.5 Interrupt Requests

The SSU interrupt requests consist of transmit data register empty, transmit end, receive data register full, overrun error, and conflict error. Of these interrupt sources, transmit data register empty, transmit end, receive data register full can activate the DTC for data transfer.

The TDRE, TEND, and RDRF bits are automatically cleared to 0 by the DTC data transfer. Since these interrupt requests are allocated to four vector addresses: SSEr_i0, SSRx_i0, SSTx_i0 and SSERT_i1, the interrupt sources must be distinguished by flags. Table 16.2 lists interrupt sources.

**Table 16.2 Interrupt Souses**

| Channel | Abbreviation | Interrupt Request | Symbol | Interrupt Condition |
|---|---|---|---|---|
| 0 | SSEr_i0 | Overrun error | OEI | (RIE = 1) • (ORER = 1) |
| | | Conflict error | CEI | (CEIE = 1) • (CE = 1) |
| | SSRx_i0 | Receive data register full | RXI | (RIE = 1) • (RDRF = 1) |
| | SSTx_i0 | Transmit data register empty | TXI | (TIE = 1) • (TDRE = 1) |
| | | Transmit end | TEI | (TEIE = 1) • (TEND = 1) |
| 1 | SSERT_i1 | Overrun error | OEI | (RIE = 1) • (ORER = 1) |
| | | Conflict error | CEI | (CEIE = 1) • (CE = 1) |
| | | Receive data register full | RXI | (RIE = 1) • (RDRF = 1) |
| | | Transmit data register empty | TXI | (TIE = 1) • (TDRE = 1) |
| | | Transmit end | TEI | (TEIE = 1) • (TEND = 1) |

When interrupt conditions shown in table 16.2 are satisfied and the I bit in CCR is 0, the CPU executes interrupt exception processing. Clear each interrupt source in the exception processing.

## 16.6 Usage Note

### 16.6.1 Setting of Module Stop Mode

The SSU can be enabled/disabled by the module stop control register setting and is disabled by the initial value. Canceling module stop mode enables to access the SSU registers. For details, see section 21, Power-Down Modes.

# Section 17 A/D Converter

This LSI includes a successive approximation type 10-bit A/D converter that allows up to sixteen analog input channels to be selected. The block diagram of the A/D converter is shown in figure 17.1.

## 17.1 Features

- 10-bit resolution
- Sixteen input channels
- Conversion time: 11.08 µs per channel (at 24 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
- Three conversion start methods
  - Software
  - 16-bit timer pulse unit (TPU) conversion start trigger
  - External trigger signal
- Interrupt request
  - An A/D conversion end interrupt request (ADI) can be generated
- Module stop mode can be set

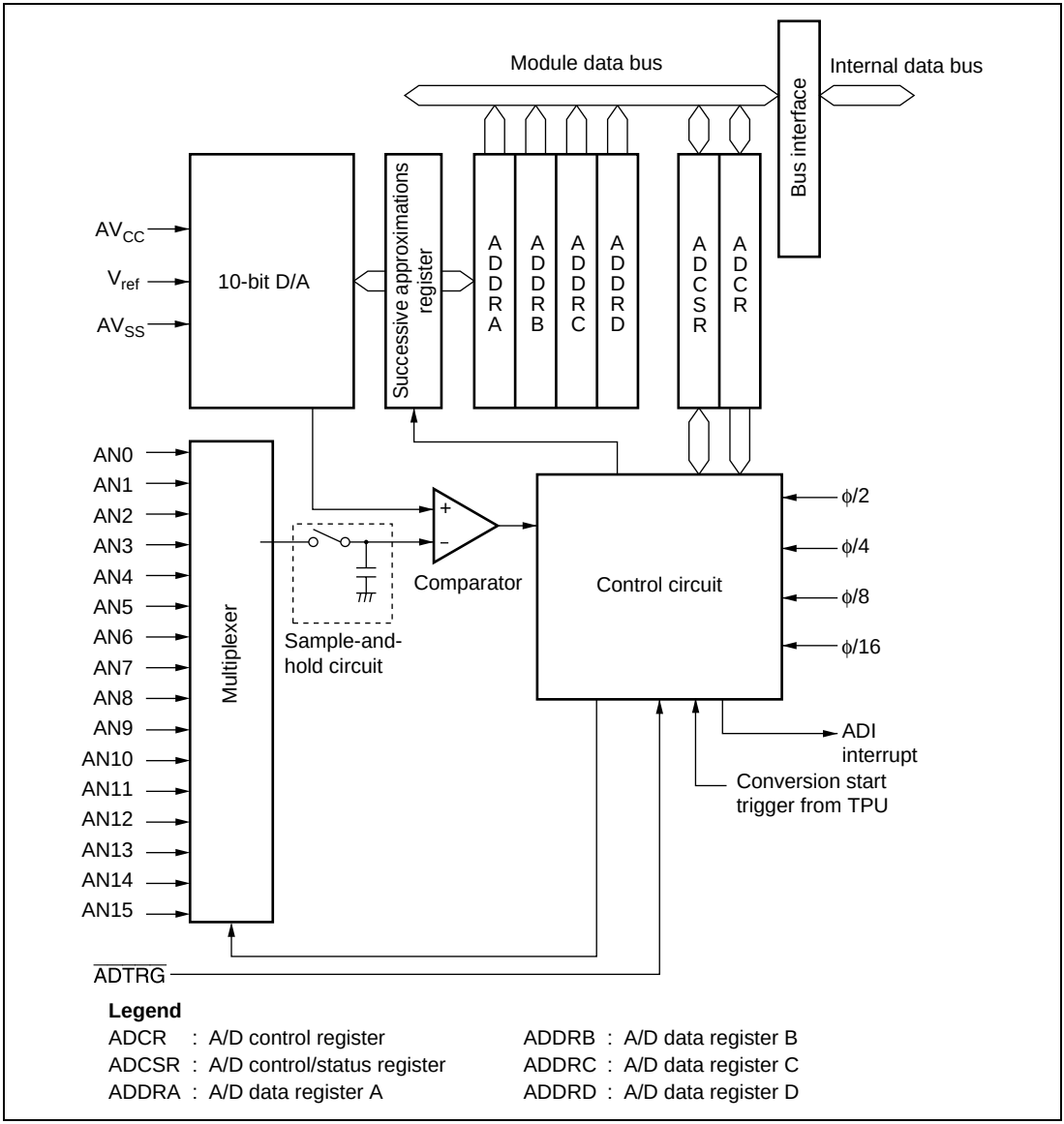



**Figure 17.1 Block Diagram of A/D Converter**

## 17.2 Input/Output Pins

Table 17.1 summarizes the input pins used by the A/D converter. 16 analog input pins are divided into four groups, each of which includes four channels; analog input pins 0 to 3 (AN0 to AN3) comprising group 0, analog input pins 4 to 7 (AN4 to AN7) comprising group 1, analog input pins 8 to 11 (AN8 to AN11) comprising group 2, and analog input pins 12 to 15 (AN12 to AN15) comprising group 3. The AVcc and AVss pins are the power supply pins for the A/D converter analog section. The Vref pin is the A/D conversion reference voltage pin.

**Table 17.1 Pin Configuration**

| Pin Name | Symbol | I/O | Function |
|---|---|---|---|
| Analog power supply pin | $AV_{CC}$ | Input | Analog section power supply and reference voltage |
| Analog ground pin | $AV_{SS}$ | Input | Analog section ground and reference voltage |
| Reference voltage pin | Vref | Input | Reference voltage of A/D conversion |
| Analog input pin 0 | AN0 | Input | Group 0 analog input pins |
| Analog input pin 1 | AN1 | Input | |
| Analog input pin 2 | AN2 | Input | |
| Analog input pin 3 | AN3 | Input | |
| Analog input pin 4 | AN4 | Input | Group 1 analog input pins |
| Analog input pin 5 | AN5 | Input | |
| Analog input pin 6 | AN6 | Input | |
| Analog input pin 7 | AN7 | Input | |
| Analog input pin 8 | AN8 | Input | Group 2 analog input pins |
| Analog input pin 9 | AN9 | Input | |
| Analog input pin 10 | AN10 | Input | |
| Analog input pin 11 | AN11 | Input | |
| Analog input pin 12 | AN12 | Input | Group 3 analog input pins |
| Analog input pin 13 | AN13 | Input | |
| Analog input pin 14 | AN14 | Input | |
| Analog input pin 15 | AN15 | Input | |
| A/D external trigger input pin | $\overline{\text{ADTRG}}$ | Input | External trigger input pin for starting A/D conversion |

## 17.3 Register Description

The A/D converter has the following registers. Module stop mode for the A/D converter is specified with the MSTPA1 bit in the module stop control register (MSTPCRA). For details on the module stop control register A (MSTPCRA), refer to section 21.1.2, Module Stop Control Register A to C (MSTPCRA to MSTPCRC).

- 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)

### 17.3.1 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 to store conversion results for each channel are shown in table 17.2.

The converted 10-bit data is stored in bits 6 to 15 in ADDR. 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. When reading the ADDR, always read the upper byte first, and then read the lower byte, or read in word unit. Otherwise, the read contents are not guaranteed.

**Table 17.2 Analog Input Channels and Corresponding ADDR Registers**

| Analog Input Channel | | | | |
|---|---|---|---|---|
| CH3 = 0 | | CH3 = 1 | | A/D Data Register to Store the A/D Conversion Results |
| Group 0 (CH2 = 0) | Group 1 (CH2 = 1) | Group 2 (CH2 = 0) | Group 3 (CH2 = 1) | |
| AN0 | AN4 | AN8 | AN12 | ADDRA |
| AN1 | AN5 | AN9 | AN13 | ADDRB |
| AN2 | AN6 | AN10 | AN14 | ADDRC |
| AN3 | AN7 | AN11 | AN15 | ADDRD |

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

ADCSR controls A/D conversion operations.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | ADF | 0 | R/(W) | A/D End Flag<br>A status flag that indicates the end of A/D conversion.<br>[Setting conditions]<br>• When A/D conversion ends<br>• When A/D conversion ends on all specified channels<br><br>[Clearing conditions]<br>• When 0 is written after reading ADF = 1<br>• When the DTC is activated by an ADI interrupt and ADDR is read |
| 6 | ADIE | 0 | R/W | A/D Interrupt Enable<br>A/D conversion end interrupt (ADI) is enabled when this bit is set to 1. |
| 5 | ADST | 0 | R/W | A/D Start<br>Clearing this bit to 0 stops A/D conversion, and the A/D converter enters the wait state.<br>Setting this bit to 1 starts A/D conversion. In single mode, this bits is automatically cleared to 0 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 software standby mode, hardware standby mode or module stop mode. |

| Bit | Bit Name | Initial Value | R/W | Description | |
|---|---|---|---|---|---|
| 4 | SCAN | 0 | R/W | Scan Mode | |
| | | | | Selects the A/D conversion operating mode. | |
| | | | | 0: Single mode | |
| | | | | 1: Scan mode | |
| 3 | CH3 | 0 | R/W | Channel Select 0 to 3 | |
| 2 | CH2 | 0 | R/W | Select analog input channels. | |
| 1 | CH1 | 0 | R/W | When SCAN = 0 | When SCAN = 1 |
| 0 | CH0 | 0 | R/W | 0000: AN0 | 0000: AN0 |
| | | | | 0001: AN1 | 0001: AN0, AN1 |
| | | | | 0010: AN2 | 0010: AN0 to AN2 |
| | | | | 0011: AN3 | 0011: AN0 to AN3 |
| | | | | 0100: AN4 | 0100: AN4 |
| | | | | 0101: AN5 | 0101: AN4, AN5 |
| | | | | 0110: AN6 | 0110: AN4 to AN6 |
| | | | | 0111: AN7 | 0111: AN4 to AN7 |
| | | | | 1000: AN8 | 1000: AN8 |
| | | | | 1001: AN9 | 1001: AN8, AN9 |
| | | | | 1010: AN10 | 1010: AN8 to AN10 |
| | | | | 1011: AN11 | 1011: AN8 to AN11 |
| | | | | 1100: AN12 | 1100: AN12 |
| | | | | 1101: AN13 | 1101: AN12, AN13 |
| | | | | 1110: AN14 | 1110: AN12 to AN14 |
| | | | | 1111: AN15 | 1111: AN12 to AN15 |

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

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

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TRGS1 | 0 | R/W | Timer Trigger Select 0 and 1 |
| 6 | TRGS0 | 0 | R/W | Enable the start of A/D conversion by a trigger signal. Bits TRGS0 and TRGS1 should be set while A/D conversion is stopped (ADST = 0).<br>00: A/D conversion is started by software<br>01: A/D conversion is started by TPU conversion start trigger<br>10: Setting prohibited<br>11: A/D conversion is started by the $\overline{\text{ADTRG}}$ pin |
| 5, 4 | — | All 1 | — | Reserved<br>These bits are always read as 1. |
| 3 | CKS1 | 0 | R/W | Clock Select 0 and 1 |
| 2 | CKS0 | 0 | R/W | Specify the A/D conversion time. The conversion time should be changed only when ADST = 0. Specify a value within the range shown in table 23.7 in section 23, Electrical Characteristics.<br>00: Conversion time = 530 states (max.)<br>01: Conversion time = 266 states (max.)<br>10: Conversion time = 134 states (max.)<br>11: Conversion time = 68 states (max.) |
| 1, 0 | — | All 1 | — | Reserved<br>These bits are always read as 1. |

## 17.4 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, clear the ADST bit in ADCSR to 0 first in order to prevent incorrect operation. The ADST bit can be set at the same time as the operating mode or analog input channel is changed.

### 17.4.1 Single Mode

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

1. A/D conversion is started when the ADST bit is set to 1 by software or external trigger input.
2. When A/D conversion is completed, the result is transferred to the A/D data register corresponding 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 retains 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. If the ADST bit is cleared to 0 during A/D conversion, the conversion is stopped and the A/D converter enters the wait state.

### 17.4.2 Scan Mode

In scan mode, A/D conversion is to be performed sequentially on the specified channels up to four channels as follows.

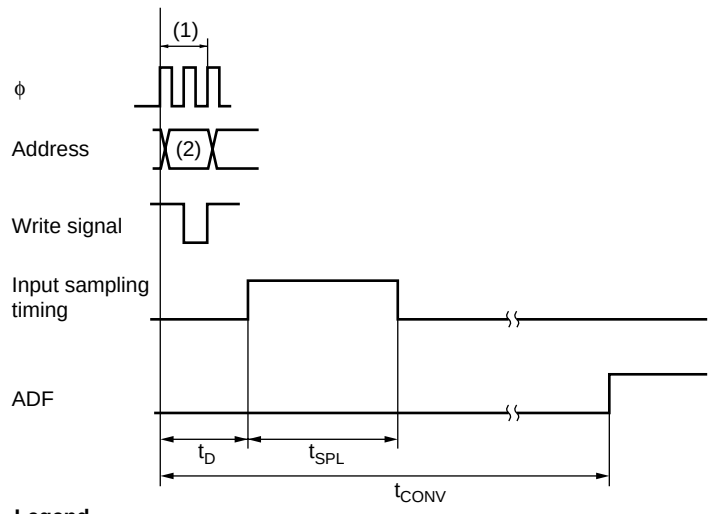
1. When the ADST bit is set to 1 by software, TPU or external trigger input, A/D conversion starts on the first channel in the group (for example, AN0 when CH3 and CH2 = 00, AN4 when CH3 and CH2 = 01, AN8 when CH3 and CH2 = 10, or AN12 when CH3 and CH2 = 11).
2. When the A/D conversion is completed on one channel, the result is sequentially transferred to the A/D data register corresponding to the channel.
3. When the conversion is completed on all the selected channels, the ADF bit in ADCSR is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt is requested after A/D conversion ends. Then, the A/D converter restarts the conversion from the first channel in the group.
4. Steps 2 to 3 are repeated as long as the ADST bit is set to 1. When the ADST bit is cleared to 0, the A/D conversion stops and the A/D converter enters the wait state.

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

The A/D converter includes the 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, and then conversion is started. Figure 17.2 shows the A/D conversion timing. Table 17.3 shows the A/D conversion time.

As shown in figure 17.2, the A/D conversion time ($t_{CONV}$) includes $t_D$ and input sampling time ($t_{SPL}$). The length of $t_D$ varies depending on the timing of the write access to ADCSR. Therefore, the total conversion time varies within the range shown in table 17.3.

In scan mode, the values given in table 17.3 indicate the first conversion time. The second and subsequent conversion time is shown in table 17.4. In both cases, set bits CKS1 and CKS0 in ADCR within the range shown in table 23.7 in section 23, Electrical Characteristics.




**Legend**
(1) : ADCSR write cycle
(2) : ADCSR address
$t_D$ : A/D conversion start delay
$t_{SPL}$ : Input sampling time
$t_{CONV}$ : A/D conversion time

**Figure 17.2 A/D Conversion Timing**

**Table 17.3 A/D Conversion Time (Single Mode)**

| | | CKS1 = 0 | | | | | | CKS1 = 1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CKS0 = 0 | | | CKS0 = 1 | | | CKS0 = 0 | | | CKS0 = 1 | | |
| **Item** | **Symbol** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** | **Min** | **Typ** | **Max** |
| A/D conversion start delay | $t_D$ | 18 | — | 33 | 10 | — | 17 | 6 | — | 9 | 4 | — | 5 |
| Input sampling time | $t_{SPL}$ | — | 127 | — | — | 63 | — | — | 31 | — | — | 15 | — |
| A/D conversion time | $t_{CONV}$ | 515 | — | 530 | 259 | — | 266 | 131 | — | 134 | 67 | — | 68 |

Note: All values represent the number of states.

**Table 17.4 A/D Conversion Time (Scan Mode)**

| CKS1 | CKS0 | Conversion Time (State) |
|---|---|---|
| 0 | 0 | 512 (Fixed) |
| | 1 | 256 (Fixed) |
| 1 | 0 | 128 (Fixed) |
| | 1 | 64 (Fixed) |

### 17.4.4 External Trigger Input Timing

A/D conversion can be externally triggered. When bits TRGS0 and TRGS1 in ADCR are set to 11, an external trigger is input on the $\overline{\text{ADTRG}}$ pin. At the falling edge of the $\overline{\text{ADTRG}}$ pin, the ADST bit in ADCSR is set to 1, and the A/D conversion starts. Other operations are the same as when the ADST bit has been set to 1 by software in both single and scan modes. Figure 17.3 shows the timing.

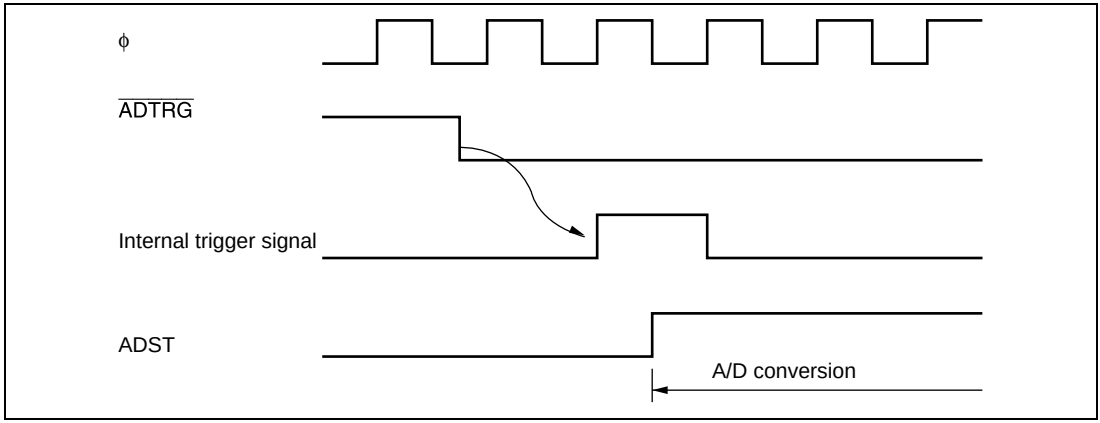



**Figure 17.3 External Trigger Input Timing**

## 17.5 Interrupt Source

When A/D conversion is completed, the A/D converter generates an A/D conversion end interrupt (ADI). The ADI interrupt request is enabled when the ADIE bit is set to 1 while the ADF bit in ADCSR is set to 1 after A/D conversion is completed. The DTC can be activated by an ADI interrupt. Having the converted data read by the DTC in response to an ADI interrupt enables continuous conversion without imposing a load on software.

**Table 17.5 A/D Converter Interrupt Source**

| Name | Interrupt Source | Interrupt Source Flag | DTC Activation |
|---|---|---|---|
| ADI | A/D conversion completed | ADF | Possible |

## 17.6 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 17.4).
- 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 B′0000000000 (H′000) to B′0000000001 (H′001) (see figure 17.5).
- Full-scale error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from B′1111111110 (H′3FE) to B′1111111111 (H′3FF) (see figure 17.5).
- Nonlinearity error

  The error with respect to the ideal A/D conversion characteristic between zero voltage and full-scale voltage. Does not include offset error, full-scale error, or quantization error (see figure 17.5).
- Absolute accuracy

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

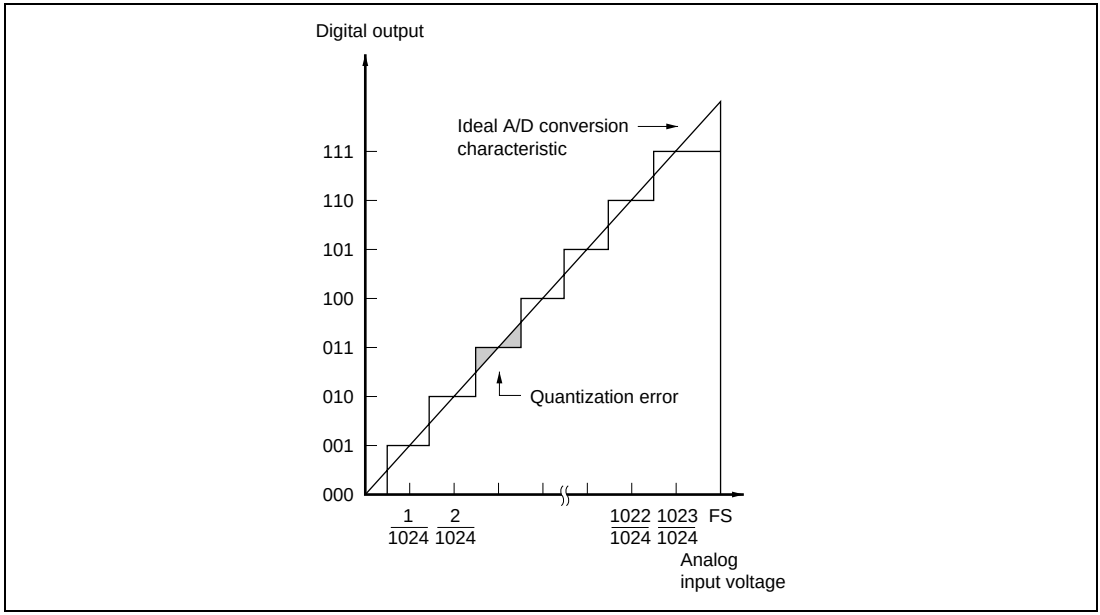



**Figure 17.4 A/D Conversion Accuracy Definitions**

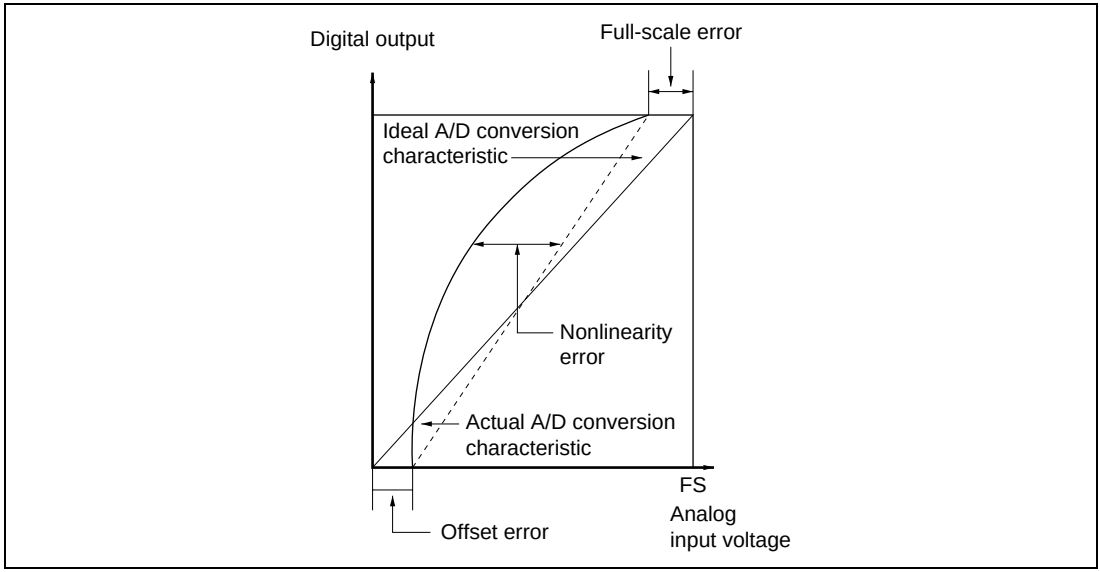



**Figure 17.5 A/D Conversion Accuracy Definitions**

## 17.7 Usage Notes

### 17.7.1 Module Stop Mode Setting

Operation of the A/D converter can be disabled or enabled using the module stop control register. The initial setting is for operation of the A/D converter to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 21, Power-Down Modes.

### 17.7.2 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 17.6). When converting a high-speed analog signal or converting in scan mode, a low-impedance buffer should be inserted.

### 17.7.3 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 such as AVss.

Care is also required to insure that filter circuits do not communicate with digital signals on the mounting board (i.e., acting as antennas).

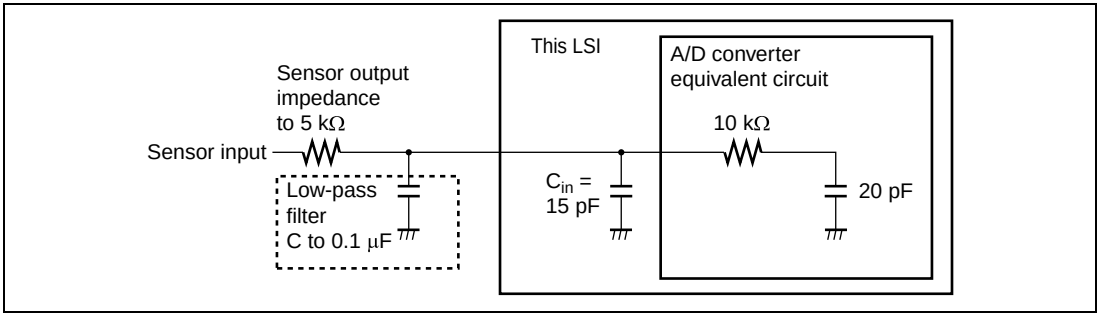



**Figure 17.6 Example of Analog Input Circuit**

### 17.7.4 Range of Analog Power Supply and Other Pin Settings

If the conditions below are not met, the reliability of the device may be adversely affected.

- Analog input voltage range
  The voltage applied to analog input pin ANn during A/D conversion should be in the range AVss ≤ VNn ≤ AVcc.
- Relationship between AVcc, AVss and Vcc, Vss
  Set AVss = Vss as the relationship between AVcc, AVss and Vcc, Vss. If the A/D converter is not used, the AVcc and AVss pins must not be left open.
- Setting range of the Vref pin
  The reference voltage set by the Vref pin should be in the range Vref ≤ AVcc.

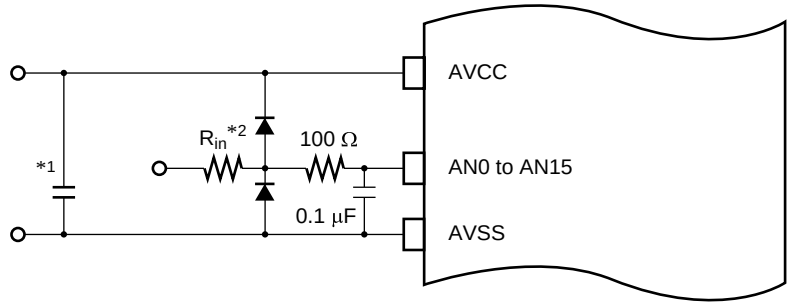
### 17.7.5 Notes on Board Design

In board design, digital circuitry and analog circuitry should be as mutually isolated as possible, and layout in which digital circuit signal lines and analog circuit signal lines cross or are in close proximity should be avoided as far as possible. Failure to do so may result in incorrect operation of the analog circuitry due to inductance, adversely affecting A/D conversion values. Also, digital circuitry must be isolated from the analog input signals (AN0 to AN15) and analog power supply (AVcc) by the analog ground (AVss). Also, the analog ground (AVss) should be connected at one point to a stable digital ground (Vss) on the board.

### 17.7.6 Notes on Noise Countermeasures

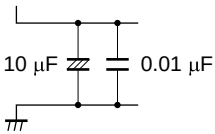
A protection circuit should be connected in order to prevent damage due to abnormal voltage, such as an excessive surge at the analog input pins (AN0 to AN15), between AVcc and AVss, as shown in figure 17.7. Also, the bypass capacitors connected to AVcc and the filter capacitor connected to AN0 to AN15 must be connected to AVss.

If a filter capacitor is connected, the input currents at the analog input pins (AN0 to AN15) are averaged, and so an error may arise. Also, when A/D conversion is performed frequently, as in scan mode, if the current charged and discharged by the capacitance of the sample-and-hold circuit in the A/D converter exceeds the current input via the input impedance ($R_{in}$), an error will arise in the analog input pin voltage. Careful consideration is therefore required when deciding circuit constants.




Notes: Values are reference values.

1.

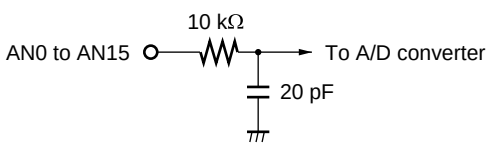



2. $R_{in}$: Input impedance

**Figure 17.7 Example of Analog Input Protection Circuit**

**Table 17.6 Analog Pin Specifications**

| Item | Min | Max | Unit |
|---|---|---|---|
| Analog input capacitance | — | 20 | pF |
| Permissible signal source impedance | — | 5 | kΩ |




Note: Values are reference values.

**Figure 17.8 Analog Input Pin Equivalent Circuit**

# Section 18 RAM

The H8S/2628 has 8 kbytes, and the H8S/2627 has 6 kbytes of on-chip high-speed static RAM. The RAM is connected to the CPU by a 16-bit data bus, enabling one-state access by the CPU to both byte data and word data.

The on-chip RAM can be enabled or disabled by means of the RAME bit in the system control register (SYSCR). For details on SYSCR, refer to section 3.2.2, System Control Register (SYSCR).

| Product Model | | ROM Type | Capacity | Address |
|---|---|---|---|---|
| H8S/2628 Series | HD64F2628 | Flash memory version | 8 kbytes | H′FFD000 to H′FFEFBF |
| | HD6432628 | Masked ROM version | 8 kbytes | H′FFD000 to H′FFEFBF |
| | HD6432627 | | 6 kbytes | H′FFD800 to H′FFEFBF |

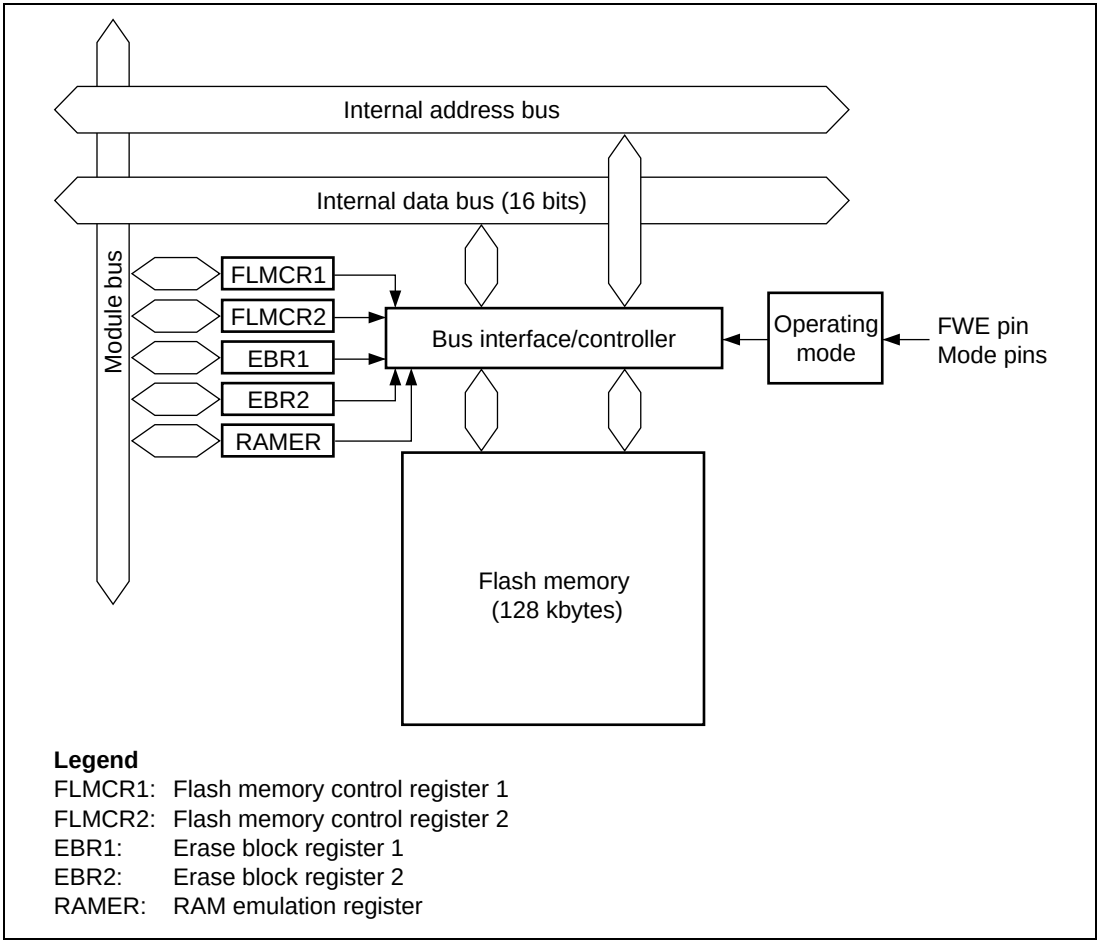
# Section 19 ROM

The features of the flash memory are summarized below.

The block diagram of the flash memory is shown in figure 19.1.

## 19.1 Features

- Size: 128 kbytes
- 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: 32 kbytes × 2 blocks, 28 kbytes × 1 block, 16 kbytes × 1 block, 8 kbytes × 2 blocks, and 1 kbyte × 4 blocks. To erase the entire flash memory, each block must be erased in turn.
- Reprogramming capability
  - The flash memory can be reprogrammed up to 100 times.
- Three programming modes
  - Boot mode
  - User mode
  - Programmer mode
  - On-board programming/erasing can be done in boot mode, in which the boot program built into the chip is started to erase or program of the entire flash memory. In normal user program mode, individual blocks can be erased or programmed.
- Programmer mode
  - Flash memory can be programmed/erased in programmer mode using a PROM programmer, as well as in on-board programming mode.
- Automatic bit rate adjustment
  - For data transfer in boot mode, this LSI's bit rate can be automatically adjusted to match the transfer bit rate of the host.
- Programming/erasing protection
  - Sets software protection against flash memory programming/erasing.




**Legend**

FLMCR1: Flash memory control register 1
FLMCR2: Flash memory control register 2
EBR1: Erase block register 1
EBR2: Erase block register 2
RAMER: RAM emulation register

**Figure 19.1 Block Diagram of Flash Memory**

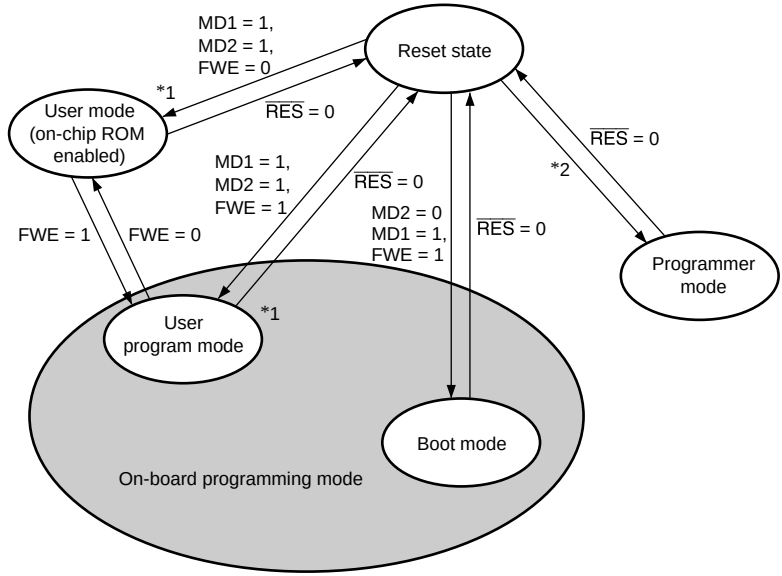
## 19.2 Mode Transitions

When the mode pins and the FWE pin are set in the reset state and a reset-start is executed, this LSI enters an operating mode as shown in figure 19.2. In user mode, flash memory can be read but not programmed or erased.

The boot, user program and programmer modes are provided as modes to write and erase the flash memory.

The differences between boot mode and user program mode are shown in table 19.1.

Figure 19.3 shows the operation flow for boot mode and figure 19.4 shows that for user program mode.




Notes: Only make a transition between user mode and user program mode when the CPU is not accessing the flash memory.

1. RAM emulation possible
2. This LSI transits to programmer mode by using the dedicated PROM programmer.

**Figure 19.2 Flash Memory State Transitions**

**Table 19.1 Differences between Boot Mode and User Program Mode**

| | Boot Mode | User Program Mode |
|---|---|---|
| Total erase | Yes | Yes |
| Block erase | No | Yes |
| Programming control program* | (2) | (1) (2) (3) |

(1) Erase/erase-verify

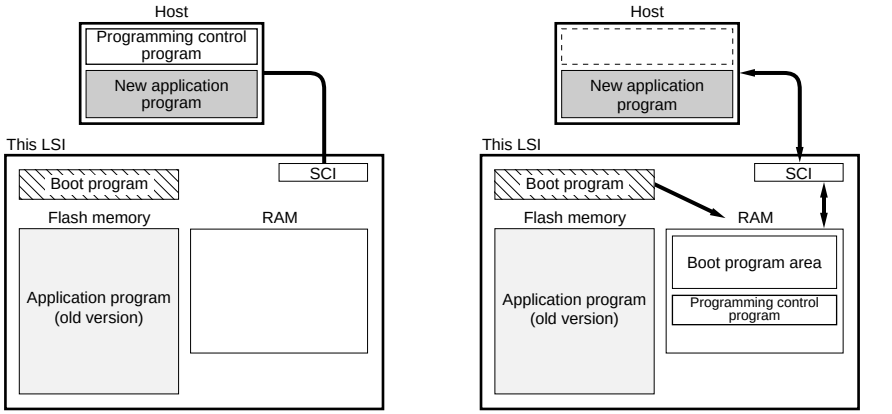
(2) Program/program-verify

(3) Emulation

Note: * To be provided by the user, in accordance with the recommended algorithm.

1. Initial state
   The old program version or data remains written in the flash memory. The user should prepare the programming control program and new application program beforehand in the host.

2. Programming control program transfer
   When boot mode is entered, the boot program in this LSI (originally incorporated in the chip) is started and the programming control program in the host is transferred to RAM via SCI communication. The boot program required for flash memory erasing is automatically transferred to the RAM boot program area.




3. Flash memory initialization
   The erase program in the boot program area (in RAM) is executed, and the flash memory is initialized (to H'FF). In boot mode, total flash memory erasure is performed, without regard to blocks.

4. Writing new application program
   The programming control program transferred from the host to RAM is executed, and the new application program in the host is written into the flash memory.

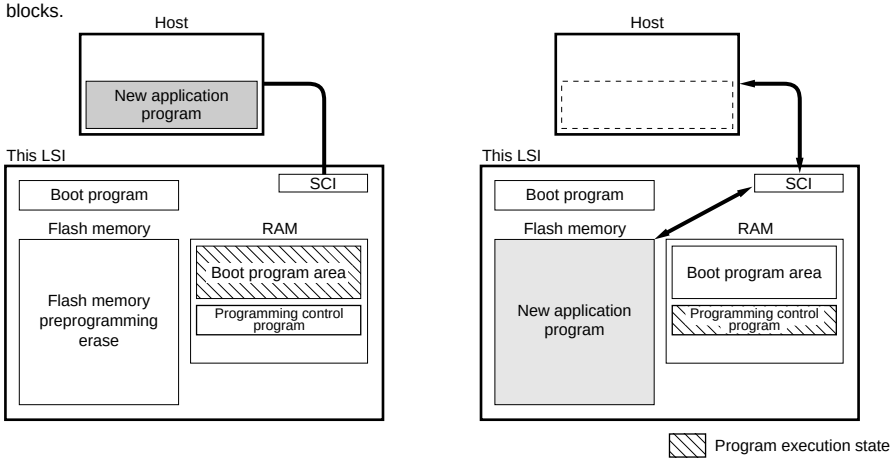
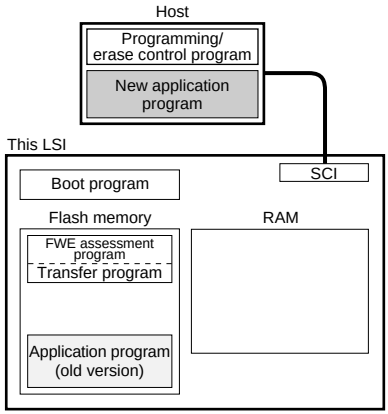
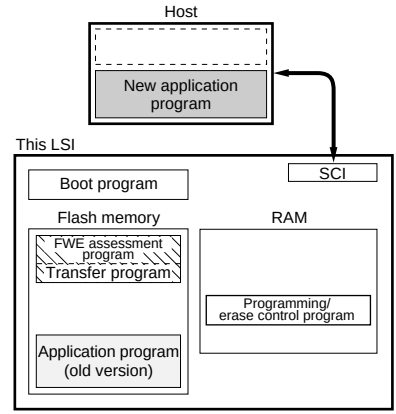



**Figure 19.3 Boot Mode**

1. Initial state
   The FWE assessment program that confirms that user program mode has been entered, and the program that will transfer the programming/erase control program from flash memory to on-chip RAM should be written into the flash memory by the user beforehand. The programming/erase control program should be prepared in the host or in the flash memory.

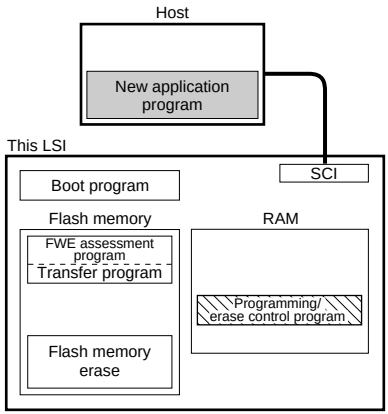
2. Programming/erase control program transfer
   When user program mode is entered, user software confirms this fact, executes transfer program in the flash memory, and transfers the programming/erase control program to RAM.

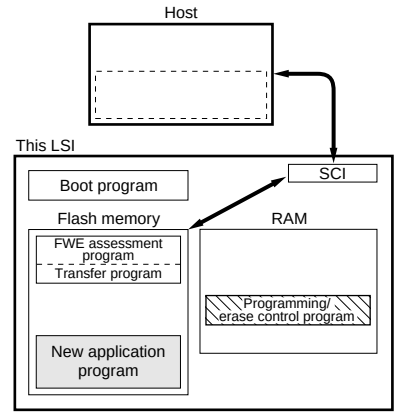



3. Flash memory initialization
   The programming/erase program in RAM is executed, and the flash memory is initialized (to H'FF). Erasing can be performed in block units, but not in byte units.

4. Writing new application program
   Next, the new application program in the host is written into the erased flash memory blocks. Do not write to unerased blocks.




Program execution state

**Figure 19.4 User Program Mode**

## 19.3 Block Configuration

Figure 19.5 shows the block configuration of 128-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 32 kbytes (2 blocks), 28 kbytes (1 block), 16 kbytes (1 block), 8 kbytes (2 blocks), and 1 kbyte (4 blocks). 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.

| | | | | | |
|---|---|---|---|---|---|
| EB0<br>Erase unit<br>1 kbyte | H'000000 | H'000001 | H'000002 | ← Programming unit: 128 bytes → | H'00007F |
| | | | | | |
| | H'000380 | H'000381 | H'000382 | | H'0003FF |
| EB1<br>Erase unit<br>1 kbyte | H'000400 | H'000401 | H'000402 | ← Programming unit: 128 bytes → | H'00047F |
| | | | | | |
| | H'000780 | H'000781 | H'000782 | | H'0007FF |
| EB2<br>Erase unit<br>1 kbyte | H'000800 | H'000801 | H'000802 | ← Programming unit: 128 bytes → | H'00087F |
| | | | | | |
| | H'000B80 | H'000B81 | H'000B82 | | H'000BFF |
| EB3<br>Erase unit<br>1 kbyte | H'000C00 | H'000C01 | H'000C02 | ← Programming unit: 128 bytes → | H'000C7F |
| | | | | | |
| | H'000F80 | H'000F81 | H'000F82 | | H'000FFF |
| EB4<br>Erase unit<br>28 kbytes | H'001000 | H'001001 | H'001002 | ← Programming unit: 128 bytes → | H'00107F |
| | | | | | |
| | H'007F80 | H'007F81 | H'007F82 | | H'007FFF |
| EB5<br>Erase unit<br>16 kbytes | H'008000 | H'008001 | H'008002 | ← Programming unit: 128 bytes → | H'00807F |
| | | | | | |
| | H'00BF80 | H'00BF81 | H'00BF82 | | H'00BFFF |
| EB6<br>Erase unit<br>8 kbytes | H'00C000 | H'00C001 | H'00C002 | ← Programming unit: 128 bytes → | H'00C07F |
| | | | | | |
| | H'00DF80 | H'00DF81 | H'00DF82 | | H'00DFFF |
| EB7<br>Erase unit<br>8 kbytes | H'00E000 | H'00E001 | H'00E002 | ← Programming unit: 128 bytes → | H'00E07F |
| | | | | | |
| | H'00FF80 | H'00FF81 | H'00FF82 | | H'00FFFF |
| EB8<br>Erase unit<br>32 kbytes | H'010000 | H'010001 | H'010002 | ← Programming unit: 128 bytes → | H'01007F |
| | | | | | |
| | H'017F80 | H'017F81 | H'017F82 | | H'017FFF |
| EB9<br>Erase unit<br>32 kbytes | H'018000 | H'018001 | H'018002 | ← Programming unit: 128 bytes → | H'01807F |
| | | | | | |
| | H'01FF80 | H'01FF81 | H'01FF82 | | H'01FFFF |

**Figure 19.5 Flash Memory Block Configuration**

## 19.4 Input/Output Pins

The flash memory is controlled by means of the pins shown in table 19.2.

**Table 19.2 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| $\overline{\text{RES}}$ | Input | Reset |
| FWE | Input | Flash program/erase protection by hardware |
| MD2 | Input | Sets this LSI's operating mode |
| MD1 | Input | Sets this LSI's operating mode |
| MD0 | Input | Sets this LSI's operating mode |
| TxD2 | Output | Serial transmit data output |
| RxD2 | Input | Serial receive data input |

## 19.5 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)
- Erase block register 2 (EBR2)
- RAM emulation register (RAMER)

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

FLMCR1 makes the flash memory enter program mode, program-verify mode, erase mode, or erase-verify mode. For details on the register setting, refer to section 19.8, Flash Memory Programming/Erasing.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | FWE | — | R | Reflects the input level at the FWE pin. It is set to 1 when a low level is input to the FWE pin, and cleared to 0 when a high level is input. |
| 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 and EBR2 bits cannot be set. |
| 5 | ESU1 | 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. |
| 4 | PSU1 | 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 P1 bit in FLMCR1. |
| 3 | EV1 | 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 | PV1 | 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 | E1 | 0 | R/W | Erase<br>When this bit is set to 1 while the SWE1 and ESU1 bits are 1, the flash memory changes to erase mode. When it is cleared to 0, erase mode is cancelled. |
| 0 | P1 | 0 | R/W | Program<br>When this bit is set to 1 while the SWE1 and PSU1 bits are 1, the flash memory changes to program mode. When it is cleared to 0, program mode is cancelled. |

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

FLMCR2 indicates 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 | Indicates that an error has occurred during flash memory programming or erasing. When the flash memory enters the error-protection state, this bit is set to 1.<br>See 19.9.3, Error Protection, for details. |
| 6 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0. |

### 19.5.3 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, otherwise, all the bits in EBR1 are automatically cleared to 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | EB7 | 0 | R/W | When this bit is set to 1, 8 kbytes of EB7 (H′00E000 to H′00FFFF) will be erased. |
| 6 | EB6 | 0 | R/W | When this bit is set to 1, 8 kbytes of EB6 (H′00C000 to H′00DFFF) will be erased. |
| 5 | EB5 | 0 | R/W | When this bit is set to 1, 16 kbytes of EB5 (H′008000 to H′00BFFF) will be erased. |
| 4 | EB4 | 0 | R/W | When this bit is set to 1, 28 kbytes of EB4 (H′001000 to H′007FFF) will be erased. |
| 3 | EB3 | 0 | R/W | When this bit is set to 1, 1 kbyte of EB3 (H′000C00 to H′000FFF) will be erased. |
| 2 | EB2 | 0 | R/W | When this bit is set to 1, 1 kbyte of EB2 (H′000800 to H′000BFF) will be erased. |
| 1 | EB1 | 0 | R/W | When this bit is set to 1, 1 kbyte of EB1 (H′000400 to H′0007FF) will be erased. |
| 0 | EB0 | 0 | R/W | When this bit is set to 1, 1 kbyte of EB0 (H′000000 to H′0003FF) will be erased. |

### 19.5.4 Erase Block Register 2 (EBR2)

EBR2 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, otherwise, all the bits in EBR1 are be automatically cleared to 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 2 | — | All 0 | — | Reserved<br>These bits are always read as 0. |
| 1 | EB9 | 0 | R/W | When this bit is set to 1, 32 kbytes of EB9 (H′018000 to H′01FFFF) will be erased. |
| 0 | EB8 | 0 | R/W | When this bit is set to 1, 32 kbytes of EB8 (H′010000 to H′017FFF) will be erased. |

### 19.5.5 RAM Emulation Register (RAMER)

RAMER specifies the area of flash memory to be overlapped with part of RAM when emulating real-time flash memory programming. RAMER settings should be made in user mode or user program mode. To ensure correct operation of the emulation function, the ROM for which RAM emulation is performed should not be accessed immediately after this register has been modified. If accessed, normal access execution is not guaranteed.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | — | Reserved |
| 6 | — | 0 | — | These bits are always read as 0. |
| 5 | — | 0 | R/W | Reserved |
| 4 | — | 0 | | Only 0 should be written to these bits. |
| 3 | RAMS | 0 | R/W | RAM Select<br>Specifies selection or non-selection of flash memory emulation in RAM. When RAMS = 1, the flash memory is overlapped with part of RAM, and all flash memory blocks are program/erase-protected. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | RAM2 | 0 | R/W | Flash Memory Area Selection |
| 1 | RAM1 | 0 | R/W | Specifies one of the following flash memory areas to overlap the RAM area of H′FFE000 to H′FFE3FF when the RAMS bit is set to 1. The areas correspond with 1-kbyte erase blocks. |
| 0 | RAM0 | 0 | R/W | 00X: H′000000 to H′0003FF (EB0) |
| | | | | 01X: H′000400 to H′0007FF (EB1) |
| | | | | 10X: H′000800 to H′000BFF (EB2) |
| | | | | 11X: H′000C00 to H′000FFF (EB3) |
| | | | | Note: X: Don’t care |

## 19.6 On-Board Programming Modes

There are two modes for programming/erasing of the flash memory; boot mode enabling on-board programming/erasing and programmer mode enabling programming/erasing with a PROM programmer. On-board programming/erasing can also be performed in user program mode. At reset-start in reset mode, this LSI changes to a mode depending on the MD pin settings and FWE pin setting, as shown in table 19.3. The input level of each pin must be defined four states before the reset ends.

When boot mode is entered, 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 SCI_2. 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 in case that programming/erasing cannot be performed 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 19.3 Setting On-Board Programming Modes**

| MD2 | MD1 | MD0 | FWE | LSI State after Reset End |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | User Mode |
| 0 | 1 | 1 | 1 | Boot Mode |

### 19.6.1 Boot Mode

Table 19.4 shows the boot mode operations from a reset end to a branch to the programming control program.

1. In boot mode, 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 19.8, Flash Memory Programming/Erasing.
2. SCI_2 should be set to asynchronous mode with the transfer format of 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 SCI_2 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. When the bit rate matching is completed, the chip transmits 1-byte data H′00 to the host to indicate the end of bit rate adjustment. The host should confirm that this adjustment end indication (H′00) has been received normally, and transmit 1-byte data H′55 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 19.5.
5. In boot mode, a part of the on-chip RAM area is used by the boot program. The area H′FFE800 to H′FFEFBF is used to store the programming control program to be transferred from the host. The boot program area cannot be used until the execution is shifted to the programming control program.
6. Before branching to the programming control program, the chip terminates transfer operations by SCI_2 (by clearing the RE and TE bits in SCR to 0), however the adjusted bit rate value is retained in BRR. Therefore, the programming control program can still use it for transfer of write data or verify data with the host. At this time, the TxD pin is in the high level output state. 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, since 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 by driving the reset pin low, waiting at least 20 states, and then setting the mode (MD) pins. Boot mode is also cleared when a WDT overflow occurs.
8. Do not change the MD pin input level in boot mode.
9. All interrupts are disabled during programming or erasing of the flash memory.

**Table 19.4 Boot Mode Operation**

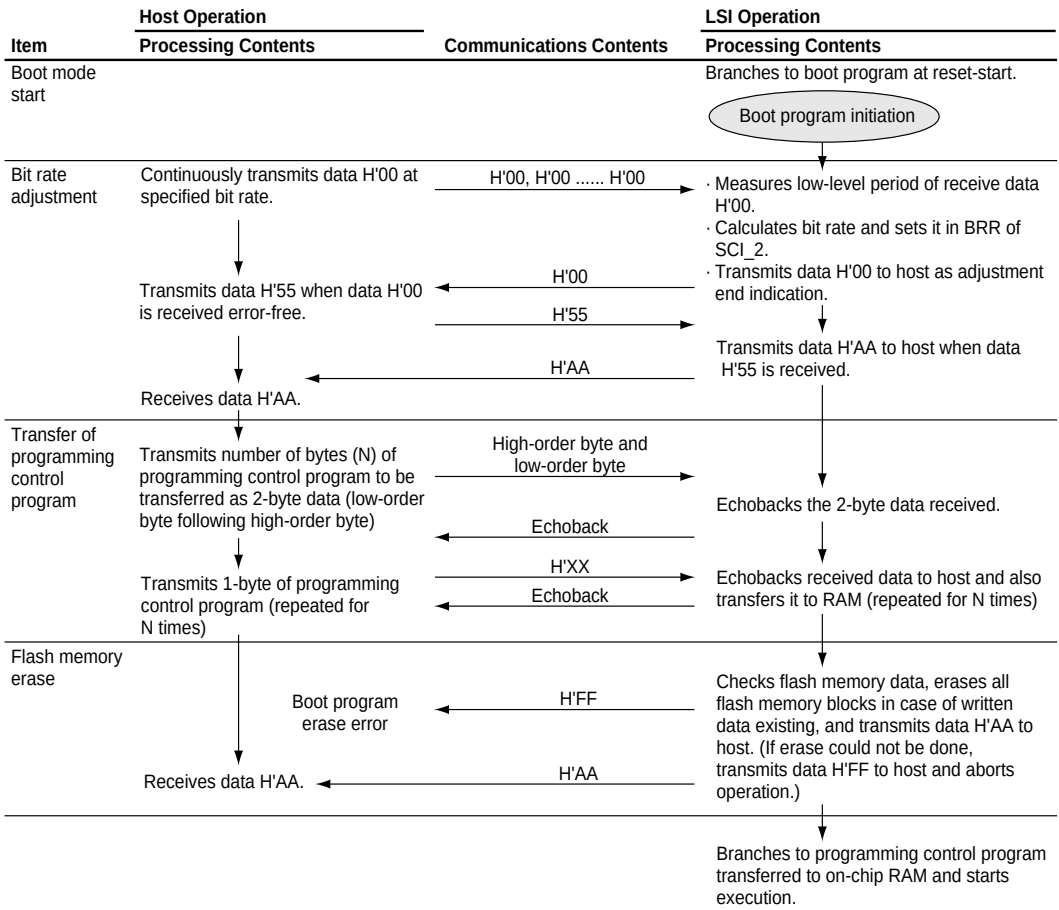
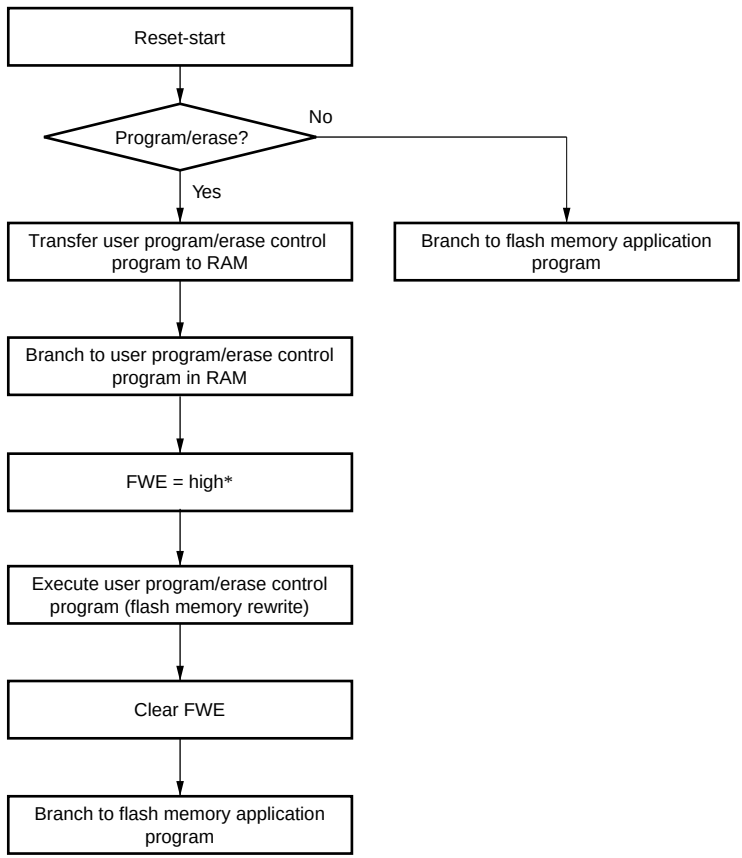



**Table 19.5 System Clock Frequencies for which Automatic Adjustment of LSI Bit Rate is Possible**

| Host Bit Rate | System Clock Frequency Range of LSI |
|---|---|
| 19,200 bps | 24 MHz |
| 9,600 bps | 8 to 24 MHz |
| 4,800 bps | 4 to 24 MHz |

### 19.6.2 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. Since 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 19.6 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 19.8, Flash Memory Programming/Erasing.




Note: * Do not constantly apply a high level to the FWE pin. Only apply a high level to the FWE pin when programming or erasing the flash memory. To prevent excessive programming or erasing, while a high level is being applied to the FWE pin, activate the watchdog timer in case of handling CPU runaways.

**Figure 19.6 Programming/Erasing Flowchart Example in User Program Mode**

## 19.7 Flash Memory Emulation in RAM

A setting in the RAM emulation register (RAMER) enables part of RAM to be overlapped onto the flash memory area so that data to be written to flash memory can be emulated in RAM in real time. Emulation can be performed in user mode or user program mode. Figure 19.7 shows an example of emulation of real-time flash memory programming.

1. Set RAMER to overlap part of RAM onto the area for which real-time programming is required.
2. Emulation is performed using the overlapping RAM.
3. After the program data has been confirmed, the RAMS bit is cleared, thus releasing the RAM overlap.
4. The data written in the overlapping RAM is written into the flash memory space (EB0).

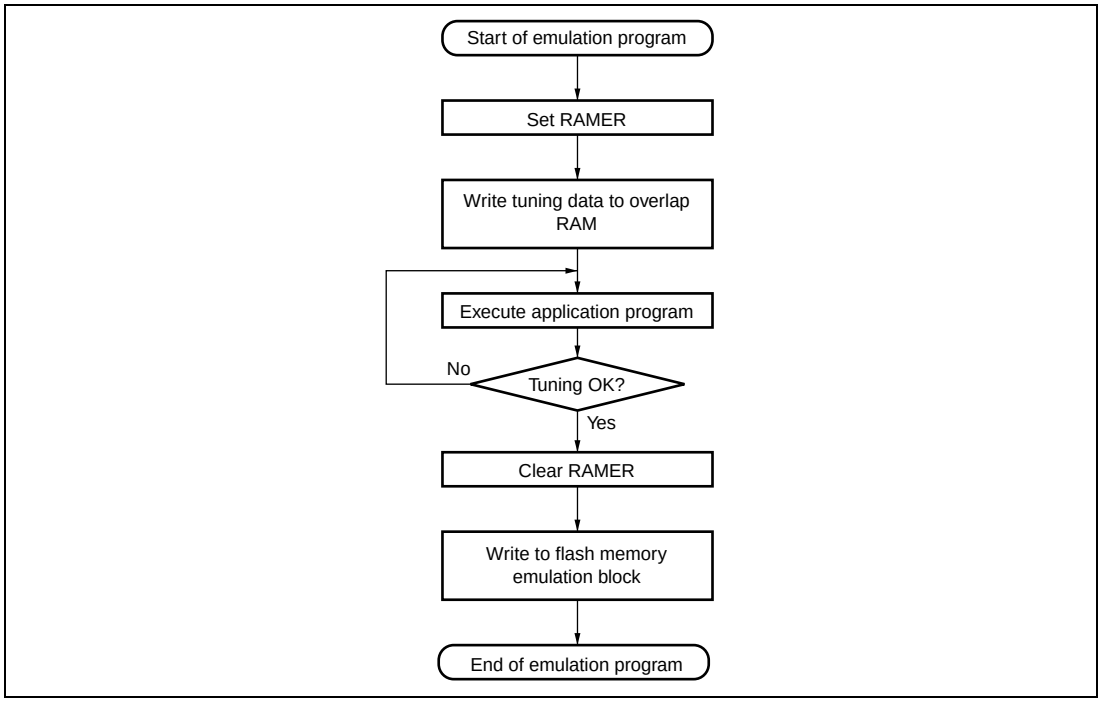



**Figure 19.7 Flowchart for Flash Memory Emulation in RAM**

An example in which flash memory block area EB0 is overlapped is shown in figure 19.8.

1. The RAM area to be overlapped is fixed at a 1-kbyte area in the range H′FFE000 to H′FFE3FF.
2. The flash memory area to overlap is selected by RAMER from a 1-kbyte area of the EB0 to EB3 blocks.
3. The overlapped RAM area can be accessed from both the flash memory addresses and RAM addresses.
4. When the RAMS bit in RAMER is set to 1, program/erase protection is enabled for all flash memory blocks (emulation protection). In this state, setting the P1 or E1 bit in FLMCR1 to 1 does not make a transition to program mode or erase mode.
5. A RAM area cannot be erased by execution of software in accordance with the erase algorithm.
6. Block area EB0 contains the vector table. When performing RAM emulation, the vector table is needed in the overlap RAM.

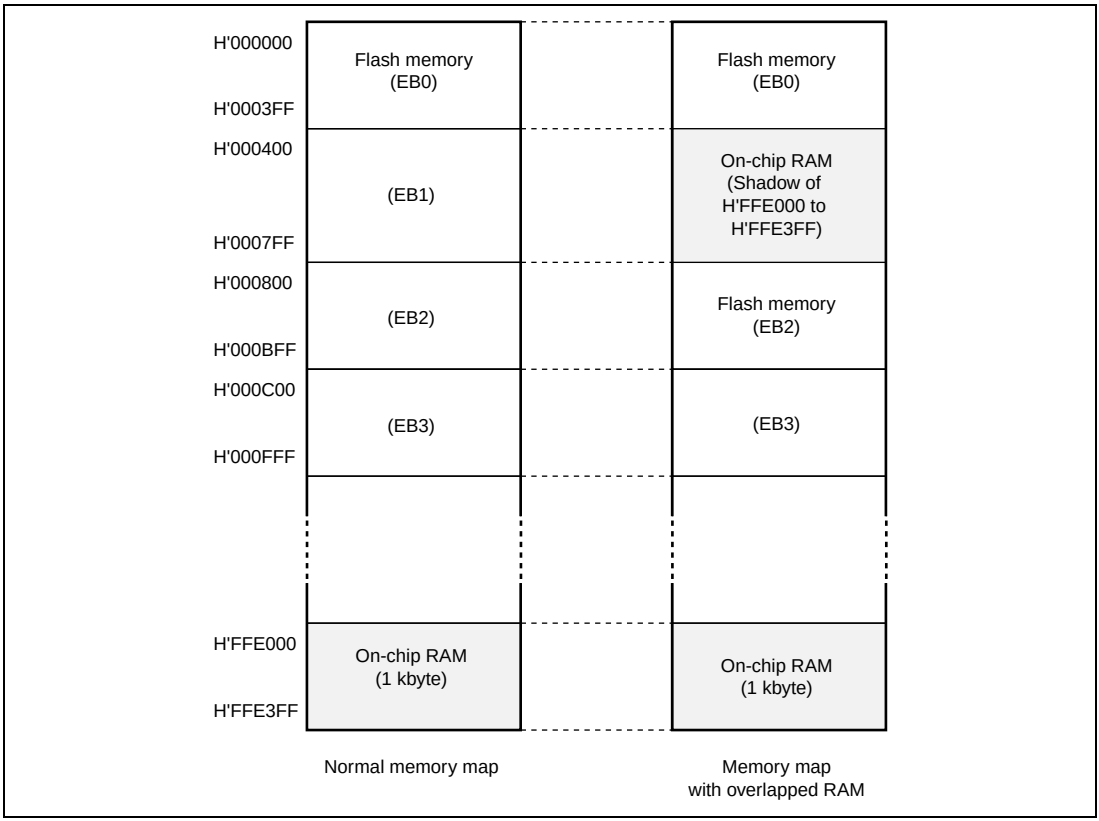



**Figure 19.8 Example of RAM Overlap Operation**

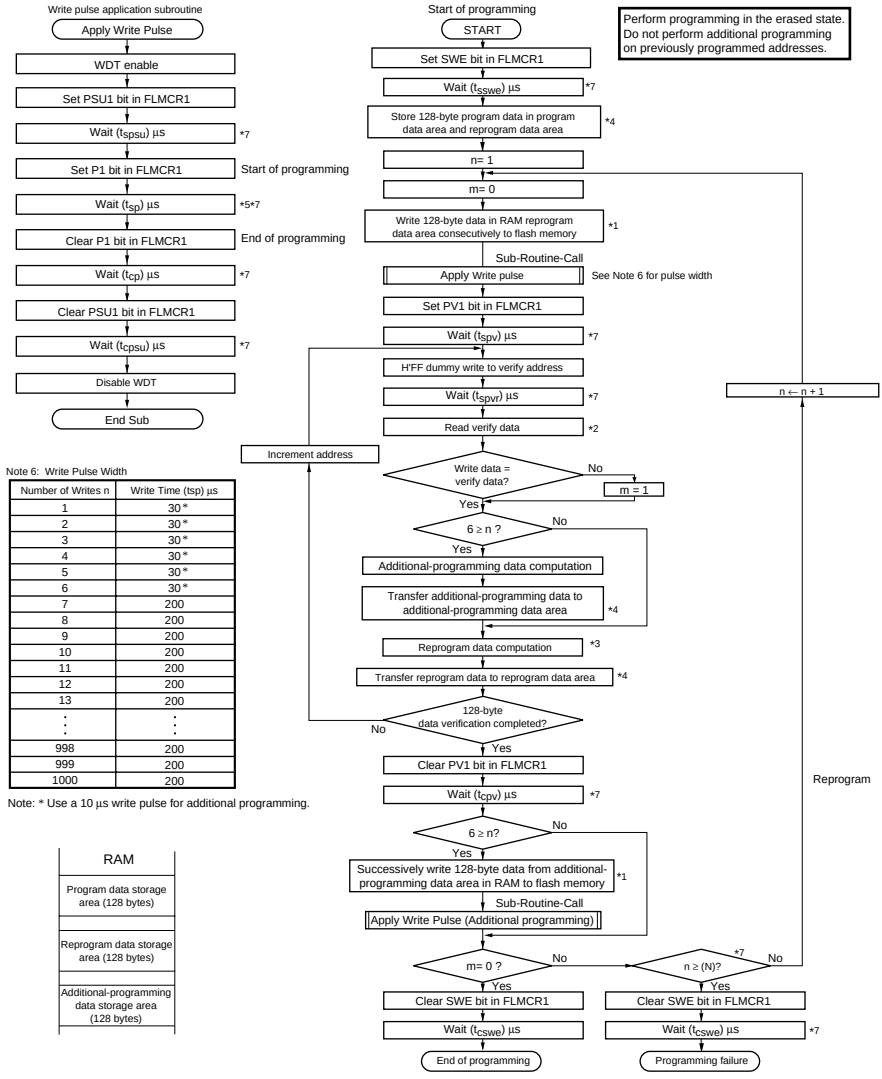
## 19.8 Flash Memory Programming/Erasing

The flash memory is programmed or erased in on-board programming mode by a software method using the CPU. 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 perform programming/erasing in combination with these modes. Flash memory programming and erasing should be performed in accordance with the descriptions in section 19.8.1, Program/Program-Verify and section 19.8.2, Erase/Erase-Verify, respectively.

### 19.8.1 Program/Program-Verify

When writing data or programs to the flash memory, the program/program-verify flowchart shown in figure 19.9 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 on erased addresses. Do not perform additional programming or previously programmed addresses.
2. Programming should be performed in units of 128 bytes. A 128-byte data must be transferred even if data to be written is 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 and additional programming data computation according to Figure 19.9.
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 P1 bit is set to 1 is the programming time. Figure 19.9 shows the allowable programming times.
6. The watchdog timer (WDT) is set to prevent overprogramming due to program runaway, etc. Set the overflow cycle to approximately 6.6 ms.
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 longwords from the address to which a dummy write was performed.
8. The number of repetitions of the program/program-verify sequence for the same bit should be less than 1,000.




Note 6: Write Pulse Width

| Number of Writes n | Write Time (tsp) µs |
|---|---|
| 1 | 30 * |
| 2 | 30 * |
| 3 | 30 * |
| 4 | 30 * |
| 5 | 30 * |
| 6 | 30 * |
| 7 | 200 |
| 8 | 200 |
| 9 | 200 |
| 10 | 200 |
| 11 | 200 |
| 12 | 200 |
| 13 | 200 |
| ⋮ | ⋮ |
| 998 | 200 |
| 999 | 200 |
| 1000 | 200 |

Note: * Use a 10 µs write pulse for additional programming.

Notes:
1. Data transfer is performed by byte transfer. The lower 8 bits of the first address written to must be H'00 or H'80. 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.
2. Verify data is read in 16-bit (word) units.
3. Reprogram data is determined by the operation shown in the table below (comparison between the data stored in the program data area and the verify data). Bits for which the reprogram data is 0 are programmed in the next reprogramming loop. Therefore, even bits for which programming has been completed will be subjected to programming once again if the result of the subsequent verify operation is NG.
4. A 128-byte area for storing program data, a 128-byte area for storing reprogram data, and a 128-byte area for storing additional data must be provided in RAM. The contents of the reprogram data area and additional data are modified as programming proceeds.
5. A write pulse of 30 µs or 200 µs is applied according to the progress of the programming operation. See Note 6 for details of the pulse widths. When writing of additional-programming data is executed, a 10 µs write pulse should be applied. Reprogram data X' means reprogram data when the write pulse is applied.
7. The wait times and value of N are shown in section 23.5, Flash Memory characteristics.

Reprogram Data Computation Table

| Original Data (D) | Verify Data (V) | Reprogram Data (X) | Comments |
|---|---|---|---|
| 0 | 0 | 1 | Programming completed |
| 0 | 1 | 0 | Programming incomplete; reprogram |
| 1 | 0 | 1 | |
| 1 | 1 | 1 | Still in erased state; no action |

Additional-Programming Data Computation Table

| Reprogram Data (X') | Verify Data (V) | Additional-Programming Data (Y) | Comments |
|---|---|---|---|
| 0 | 0 | 0 | Additional programming to be executed |
| 0 | 1 | 1 | Additional programming not to be executed |
| 1 | 0 | 1 | Additional programming not to be executed |
| 1 | 1 | 1 | Additional programming not to be executed |

**Figure 19.9 Program/Program-Verify Flowchart**

### 19.8.2 Erase/Erase-Verify

When erasing flash memory, the erase/erase-verify flowchart shown in figure 19.10 should be followed.

1. Prewriting (setting erase block data to all 0s) is not necessary.
2. Erasing is performed in block units. Specify a single block o be erased with the erase block registers (EBR1 and EBR2). 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. Set the overflow cycle to approximately 19.8 ms.
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. Note that the number of repetitions of the erase/erase-verify sequence should be less than 100.

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

All interrupts, including the $\overline{\text{NMI}}$ interrupt, should be disabled while flash memory is being programmed, erased, or the boot program is being executed, for the following three reasons:

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

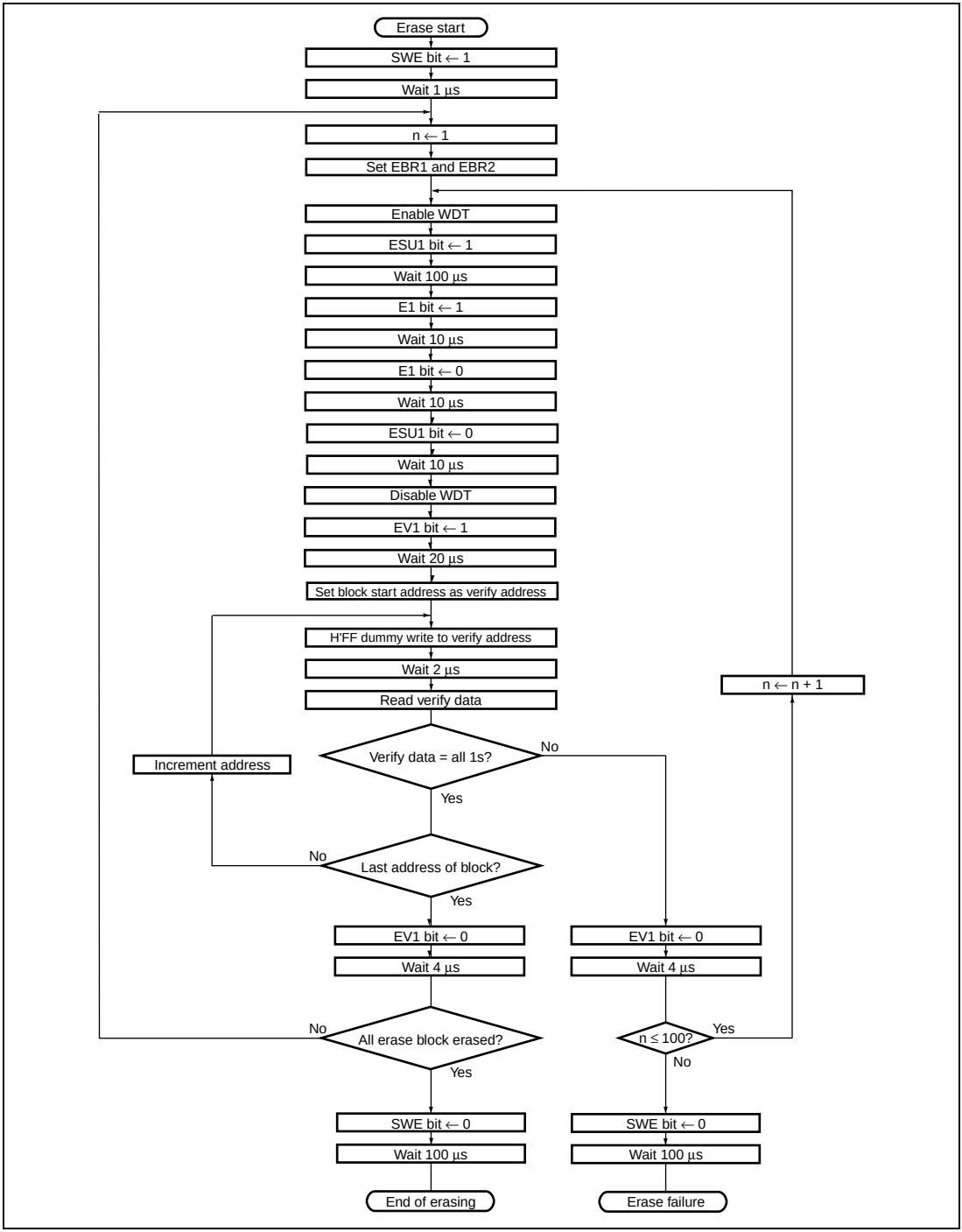



**Figure 19.10 Erase/Erase-Verify Flowchart**

## 19.9 Program/Erase Protection

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

### 19.9.1 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 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 settles 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.

### 19.9.2 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 P1 or E1 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.

### 19.9.3 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 P1 or E1 bit. However, PV1 and EV1 bit setting is enabled, and a transition can be made to verify mode. Error protection can be cleared only by a power-on reset.

## 19.10 Programmer Mode

In programmer mode, a PROM programmer can be used to perform programming/erasing via a socket adapter, just as for a discrete flash memory. Use a PROM programmer that supports the Hitachi 128-kbyte flash memory on-chip MCU device type (FZTAT128V5A).

## 19.11 Power-Down States for Flash Memory

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

- Normal operating mode
  The flash memory can be read and written to.
- Standby mode
  All flash memory circuits are halted.

Table 19.6 shows the correspondence between the operating modes of the H8S/2628 Series and the flash memory. When the flash memory returns to its normal operating state from standby mode, a period to settle the power supply circuits that were stopped is needed. When the flash memory returns to its normal operating state, bits STS2 to STS0 in SBYCR must be set to provide a wait time of at least 20 µs, even when the external clock is being used.

**Table 19.6 Flash Memory Operating States**

| LSI Operating State | Flash Memory Operating State |
|---|---|
| Active mode | Normal operating mode |
| Standby mode | Standby mode |

# Section 20 Clock Pulse Generator

This LSI has an on-chip clock pulse generator that generates the system clock (ϕ), the bus master clock, and internal clocks. The clock pulse generator consists of an oscillator, PLL circuit, clock selection circuit, medium-speed clock divider, and bus master clock selection circuit. A block diagram of the clock pulse generator is shown in figure 20.1.

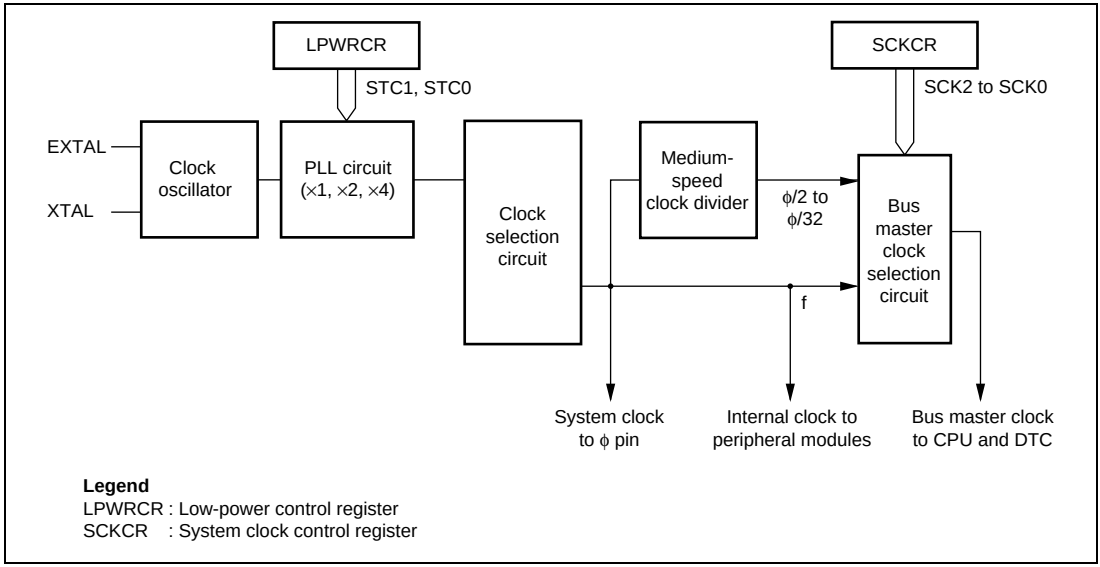



**Figure 20.1 Block Diagram of Clock Pulse Generator**

The frequency can be changed by means of the PLL circuit. Frequency changes are performed by software by settings in the low-power control register (LPWRCR) and system clock control register (SCKCR).

## 20.1 Register Descriptions

The on-chip clock pulse generator has the following registers.

- System clock control register (SCKCR)
- Low-power control register (LPWRCR)

### 20.1.1 System Clock Control Register (SCKCR)

SCKCR performs φ clock output control, selection of operation when the PLL circuit frequency multiplication factor is changed, and medium-speed mode control.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | PSTOP | 0 | R/W | φ Clock Output Disable<br>Controls φ output.<br>High-speed Mode, Medium-Speed Mode<br>0: φ output<br>1: Fixed high<br>Sleep Mode<br>0: φ output<br>1: Fixed high<br>Software Standby Mode<br>0: Fixed high<br>1: Fixed high<br>Hardware Standby Mode<br>0: High impedance<br>1: High impedance |
| 6 to 4 | — | All 0 | — | Reserved<br>These bits are always read as 0. |
| 3 | STCS | 0 | R/W | Frequency Multiplication Factor Switching Mode Select<br>Selects the operation when the PLL circuit frequency multiplication factor is changed.<br>0: Specified multiplication factor is valid after transition to software standby mode<br>1: Specified multiplication factor is valid immediately after STC1 bit and STC0 bit are rewritten |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | SCK2 | 0 | R/W | System Clock Select 0 to 2 |
| 1 | SCK1 | 0 | R/W | These bits select the bus master clock. |
| 0 | SCK0 | 0 | R/W | 000: High-speed mode |
| | | | | 001: Medium-speed clock is $\phi$/2 |
| | | | | 010: Medium-speed clock is $\phi$/4 |
| | | | | 011: Medium-speed clock is $\phi$/8 |
| | | | | 100: Medium-speed clock is $\phi$/16 |
| | | | | 101: Medium-speed clock is $\phi$/32 |
| | | | | 11X: Setting prohibited |

**Legend**
X: Don't care

### 20.1.2 Low-Power Control Register (LPWRCR)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | — | All 0 | — | Reserved<br>The write value should always be 0. |
| 3, 2 | — | All 0 | R/W | Reserved<br>These bits can be read from and write to, but should not be set to 1. |
| 1 | STC1 | 0 | R/W | Frequency Multiplication Factor |
| 0 | STC0 | 0 | R/W | The STC bits specify the frequency multiplication factor of the PLL circuit. |
| | | | | 00: $\times$1 |
| | | | | 01: $\times$2 |
| | | | | 10: $\times$4 |
| | | | | 11: Setting prohibited |

## 20.2 Oscillator

Clock pulses can be supplied by connecting a crystal resonator, or by input of an external clock. In either case, the input clock should not exceed 20 MHz.

### 20.2.1 Connecting a Crystal Resonator

Circuit Configuration: A crystal resonator can be connected as shown in the example in figure 20.2. Select the damping resistance $R_d$ according to table 20.1. An AT-cut parallel-resonance crystal should be used.

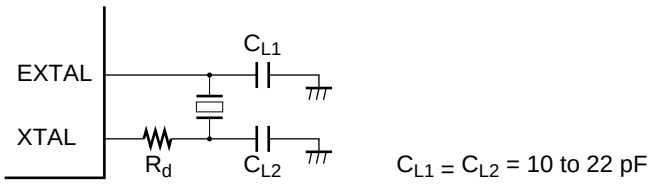



**Figure 20.2 Connection of Crystal Resonator (Example)**

**Table 20.1 Damping Resistance Value**

| Frequency (MHz) | 4 | 8 | 12 | 16 | 20 | 24 |
|---|---|---|---|---|---|---|
| $R_d$ (Ω) | 500 | 200 | 0 | 0 | 0 | 0 |

Figure 20.3 shows the equivalent circuit of the crystal resonator. Use a crystal resonator that has the characteristics shown in table 20.2.

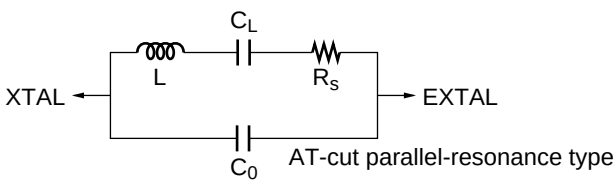



**Figure 20.3 Crystal Resonator Equivalent Circuit**

**Table 20.2 Crystal Resonator Characteristics**

| Frequency (MHz) | 4 | 8 | 12 | 16 | 20 | 24 |
|---|---|---|---|---|---|---|
| $R_S$ max (Ω) | 120 | 80 | 60 | 50 | 40 | 40 |
| $C_0$ max (pF) | 7 | 7 | 7 | 7 | 7 | 7 |

### 20.2.2 External Clock Input

Circuit Configuration: An external clock signal can be input as shown in the examples in figure 20.4. If the XTAL pin is left open, ensure that stray capacitance does not exceed 10 pF. When complementary clock is input to the XTAL pin, the external clock input should be fixed high in standby mode.

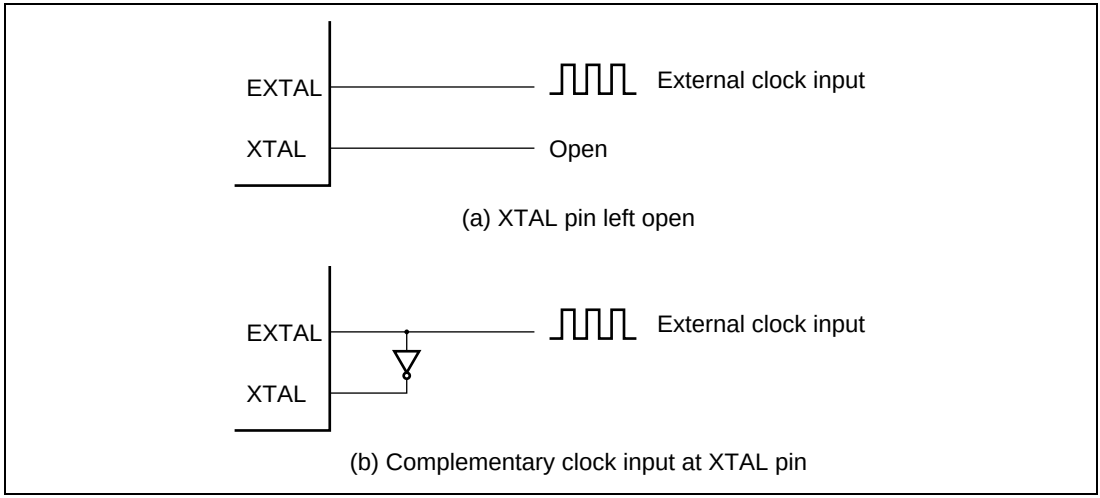



**Figure 20.4 External Clock Input (Examples)**

Table 20.3 shows the input conditions for the external clock.

**Table 20.3 External Clock Input Conditions**

| | | $V_{cc}$ = 5.0 V ± 10% | | | |
|---|---|---|---|---|---|
| **Item** | **Symbol** | **Min** | **Max** | **Unit** | **Test Conditions** |
| External clock input low pulse width | $t_{EXL}$ | 20.8 | — | ns | Figure 20.5 |
| External clock input high pulse width | $t_{EXH}$ | 20.8 | — | ns | |
| External clock rise time | $t_{EXr}$ | — | 5 | ns | |
| External clock fall time | $t_{EXf}$ | — | 5 | ns | |

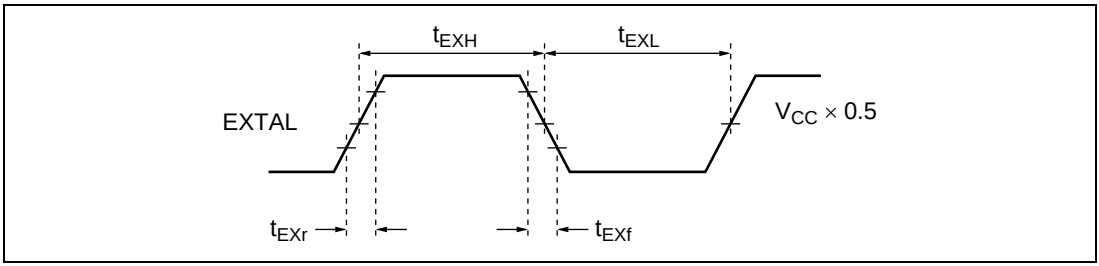



**Figure 20.5 External Clock Input Timing**

## 20.3 PLL Circuit

The PLL circuit multiplies the frequency of the clock from the oscillator by a factor of 1, 2, or 4. The multiplication factor is set by the STC0 bit and the STC1 bit in LPWRCR. The phase of the rising edge of the internal clock is controlled so as to match that at the EXTAL pin.

When the multiplication factor of the PLL circuit is changed, the operation varies according to the setting of the STCS bit in SCKCR.

When STCS = 0, the setting becomes valid after a transition to software standby mode. The transition time count is performed in accordance with the setting of bits STS0 to STS2 in SBYCR. For details on SBYCR, refer to section 21.1.1, Standby Control Register (SBYCR).

1. The initial PLL circuit multiplication factor is 1.
2. STS0 to STS2 are set to give the specified transition time.
3. The target value is set in STC0 and STC1, and a transition is made to software standby mode.
4. The clock pulse generator stops and the value set in STC0 and STC1 becomes valid.
5. Software standby mode is cleared, and a transition time is secured in accordance with the setting in STS0 to STS2.
6. After the set transition time has elapsed, this LSI resumes operation using the target multiplication factor.

If a PC break is set for the SLEEP instruction, software standby mode is entered and break exception handling is executed after the oscillation settling time. In this case, the instruction following the SLEEP instruction is executed after execution of the RTE instruction. When STCS = 1, this LSI operates on the changed multiplication factor immediately after bits STC0 and STC1 are rewritten.

## 20.4 Medium-Speed Clock Divider

The medium-speed clock divider divides the system clock to generate $\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, and $\phi/32$.

## 20.5 Bus Master Clock Selection Circuit

The bus master clock selection circuit selects the clock supplied to the bus master by setting the bits SCK 2 to 0 in SCKCR. The bus master clock can be selected from high-speed mode, or medium-speed clocks ($\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, $\phi/32$).

## 20.6 Usage Notes

### 20.6.1 Note on Crystal Resonator

As various characteristics related to the crystal resonator are closely linked to the user's board design, thorough evaluation is necessary on the user's part, using the resonator connection examples shown in this section as a guide. As the resonator circuit ratings will depend on the floating capacitance of the resonator and the mounting circuit, the ratings should be determined in consultation with the resonator manufacturer. The design must ensure that a voltage exceeding the maximum rating is not applied to the oscillator pin.

### 20.6.2 Note on Board Design

When designing the board, place the crystal resonator and its load capacitors as close as possible to the XTAL and EXTAL pins. Other signal lines should be routed away from the oscillator circuit, as shown in figure 20.6. This is to prevent induction from interfering with correct oscillation.

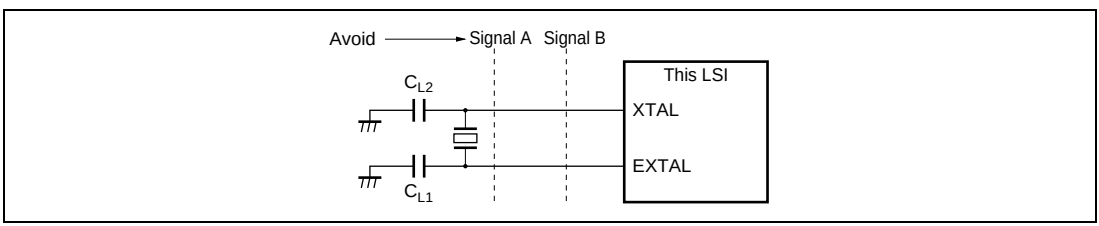



**Figure 20.6 Note on Board Design of Oscillator Circuit**

Figure 20.7 shows external circuitry recommended to be provided around the PLL circuit. Place oscillation settling capacitor C1 and resistor R1 close to the PLLCAP pin, and ensure that no other signal lines cross this line. Separate PLLVss from the other Vcc and Vss lines at the board power supply source, and be sure to insert bypass capacitors CB close to the pins.

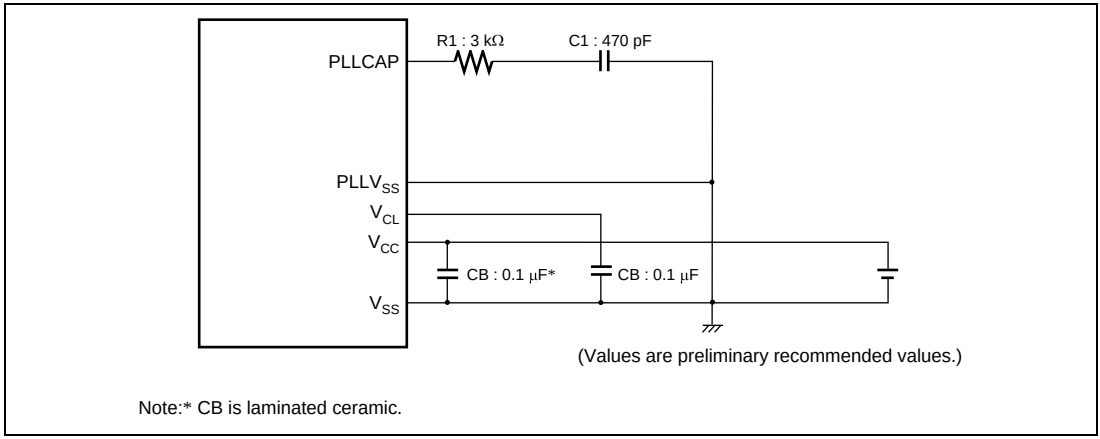



**Figure 20.7 External Circuitry Recommended for PLL Circuit**

# Section 21 Power-Down Modes

In addition to the normal program execution state, this LSI has five power-down modes in which operation of the CPU and oscillator is halted and power consumption is reduced. Low-power operation can be achieved by individually controlling the CPU, on-chip peripheral modules, and so on.

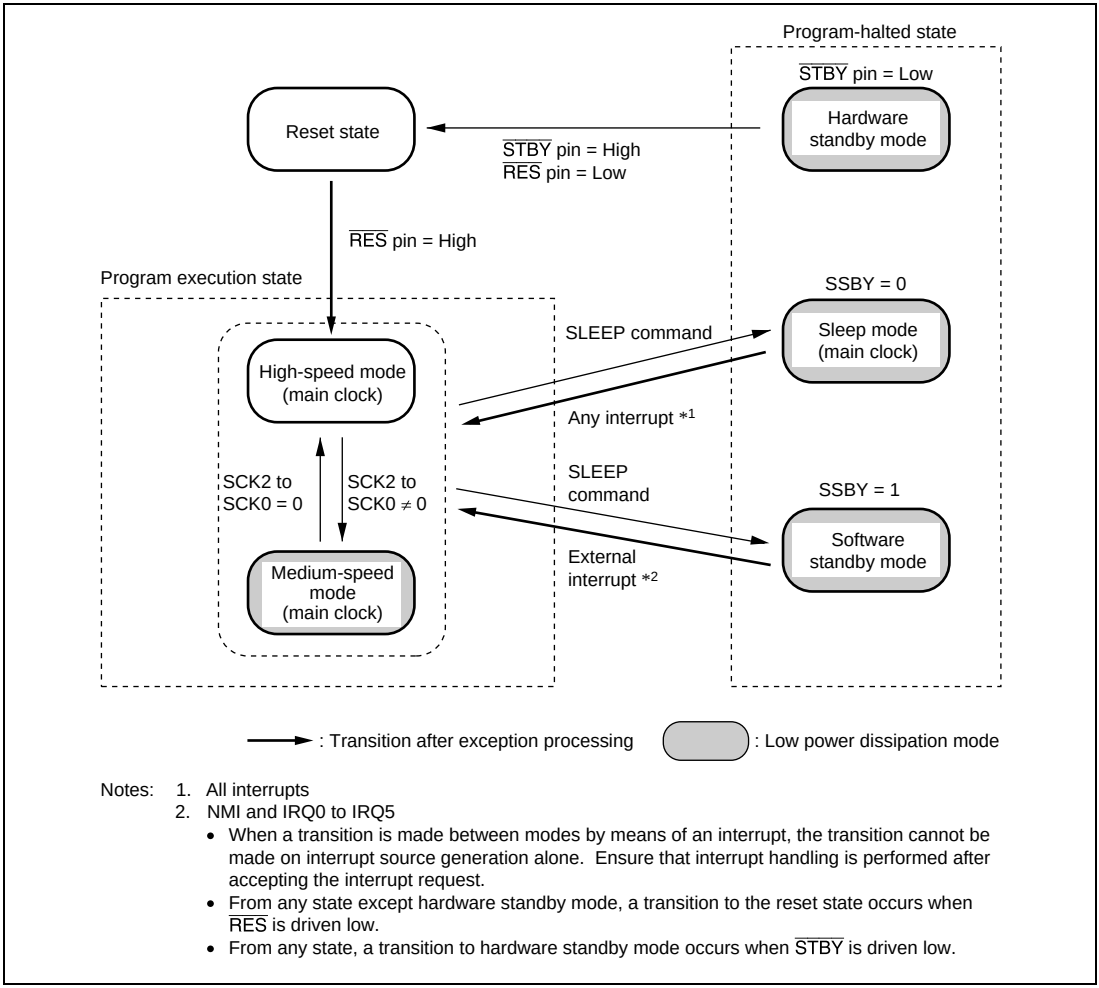
This LSI's operating modes are as follows:

1. High-speed mode
2. Medium-speed mode
3. Sleep mode
4. Module stop mode
5. Software standby mode
6. Hardware standby mode

2. to 6. are power-down modes. Sleep mode is a CPU state, medium-speed mode is a CPU and bus master state, and module stop mode is an internal peripheral function (including bus masters other than the CPU) state. Some of these states can be combined.

After a reset, the LSI is in high-speed mode.

Figure 21.1 shows possible transitions between modes. Table 21.1 shows the conditions of transition made by the SLEEP instruction and recovery from power-down mode by an interrupt. Table 21.2 shows the internal states in each mode.




Notes: 1. All interrupts
2. NMI and IRQ0 to IRQ5
- When a transition is made between modes by means of an interrupt, the transition cannot be made on interrupt source generation alone. Ensure that interrupt handling is performed after accepting the interrupt request.
- From any state except hardware standby mode, a transition to the reset state occurs when $\overline{RES}$ is driven low.
- From any state, a transition to hardware standby mode occurs when $\overline{STBY}$ is driven low.

**Figure 21.1 Mode Transition Diagram**

**Table 21.1 Low Power Consumption Mode Transition Conditions**

| Pre-Transition State | Status of Control Bit at Transition | State after Transition Invoked by SLEEP Command | State after Transition Back from Low Power Mode Invoked by Interrupt |
|---|---|---|---|
| | SSBY | | |
| High-speed/ Medium-speed | 0 | Sleep | High-speed/Medium-speed |
| | 1 | Software standby | High-speed/Medium-speed |

**Table 21.2 LSI Internal States in Each Mode**

| Function | | High-Speed | Medium-Speed | Sleep | Module Stop | Software Standby | Hardware Standby |
|---|---|---|---|---|---|---|---|
| System clock pulse generator | | Operate | Operate | Operate | Operate | Halted | Halted |
| CPU | Instructions Registers | Operate | Medium-speed operation | Halted (retained) | High/ medium-speed operation | Halted (retained) | Halted (undefined) |
| External interrupts | NMI | Operate | Operate | Operate | Operate | Operate | Halted |
| | IRQ0–IRQ5 | | | | | | |
| Peripheral functions | PBC | Operate | Medium-speed operation | Operate | Halted (retained) | Halted (retained) | Halted (reset) |
| | DTC | | | | | | |
| | I/O | Operate | Operate | Operate | Operate | Retained | High impedance |
| | TPU | Operate | Operate | Operate | Halted (retained) | Halted (retained) | Halted (reset) |
| | PPG | | | | | | |
| | WDT | Operate | Operate | Operate | Operate | Halted (retained) | Halted (reset) |
| | SCI | Operate | Operate | Operate | Halted (reset) | Halted (reset) | Halted (reset) |
| | HCAN | | | | | | |
| | A/D | | | | | | |
| | RAM | Operate | Medium-speed operation | Operate (DTC) | Operate | Retained | Retained |
| | SSU | Operate | Operate | Operate | Halted (reset) | Halted (reset) | Halted (reset) |

Notes: Halted (retained) means that internal register values are retained. The internal state is in the operation suspended state.

Halted (reset) means that internal register values and internal states are initialized.

In module stop mode, only modules for which a stop setting has been made are halted (reset or retained).

## 21.1 Register Descriptions

Registers related to the power down mode are shown below. For details on the system clock control register (SCKCR), refer to section 20.1.1, System Clock Control Register (SCKCR).

- System clock control register (SCKCR)
- Standby control register (SBYCR)
- Module stop control register A (MSTPCRA)
- Module stop control register B (MSTPCRB)
- Module stop control register C (MSTPCRC)

### 21.1.1 Standby Control Register (SBYCR)

SBYCR is an 8-bit readable/writable register that performs software standby mode control.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | SSBY | 0 | R/W | Software Standby<br>This bit specifies the transition mode after executing the SLEEP instruction<br>0: Shifts to sleep mode when the SLEEP instruction is executed<br>1: Shifts to software standby mode when the SLEEP instruction is executed<br>This bit does not change when clearing the software standby mode by using external interrupts and shifting to normal operation. This bit should be written with 0 when clearing. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | STS2 | 0 | R/W | Standby Timer Select 0 to 2 |
| 5 | STS1 | 0 | R/W | These bits select the MCU wait time for clock settling when software standby mode is cancelled by an external interrupt. With a crystal oscillator (Table 21.3), select a wait time of 8ms (oscillation settling time) or more, depending on the operating frequency. With an external clock, select a wait time of 2 ms or more.<br>000: Standby time = 8192 states<br>001: Standby time = 16384 states<br>010: Standby time = 32768 states<br>011: Standby time = 65536 states<br>100: Standby time = 131072 states<br>101: Standby time = 262144 states<br>110: Reserved<br>111: Standby time = 16 states |
| 4 | STS0 | 0 | R/W | |
| 3 | — | 1 | R/W | Reserved<br>The write value should always be 0. |
| 2 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0 and cannot be modified. |

### 21.1.2 Module Stop Control Registers A to C (MSTPCRA to MSTPCRC)

MSTPCR is comprised of three 8-bit readable/writable registers, and performs module stop mode control. Setting a bit to 1 causes the corresponding module to enter module stop mode. Clearing the bit to 0 clears the module stop mode.

- MSTPCRA

| Bit | Bit Name | Initial Value | R/W | Module |
|---|---|---|---|---|
| 7 | MSTPA7* | 0 | R/W | |
| 6 | MSTPA6 | 0 | R/W | Data transfer controller (DTC) |
| 5 | MSTPA5 | 1 | R/W | 16-bit timer pulse unit (TPU) |
| 4 | MSTPA4 | 1 | R/W | 8-bit timer (TMR_0, TMR_1) |
| 3 | MSTPA3 | 1 | R/W | Programmable pulse generator (PPG) |
| 2 | MSTPA2* | 1 | R/W | |
| 1 | MSTPA1 | 1 | R/W | A/D converter |
| 0 | MSTPA0 | 1 | R/W | 8-bit timer (TMR_2, TMR_3) |

- MSTPCRB

| Bit | Bit Name | Initial Value | R/W | Module |
|---|---|---|---|---|
| 7 | MSTPB7 | 1 | R/W | Serial communication interface 0 (SCI0) |
| 6 | MSTPB6* | 1 | R/W | |
| 5 | MSTPB5 | 1 | R/W | Serial communication interface 2 (SCI2) |
| 4 | MSTPB4* | 1 | R/W | |
| 3 | MSTPB3* | 1 | R/W | |
| 2 | MSTPB2* | 1 | R/W | |
| 1 | MSTPB1* | 1 | R/W | |
| 0 | MSTPB0* | 1 | R/W | |

- MSTPCRC

| Bit | Bit Name | Initial Value | R/W | Module |
|---|---|---|---|---|
| 7 | MSTPC7* | 1 | R/W | |
| 6 | MSTPC6* | 1 | R/W | |
| 5 | MSTPC5* | 1 | R/W | |
| 4 | MSTPC4 | 1 | R/W | PC break controller (PBC) |
| 3 | MSTPC3 | 1 | R/W | Hitachi Controller Area Network (HCAN) |
| 2 | MSTPC2 | 1 | R/W | Synchronous serial communication unit (SSU) |
| 1 | MSTPC1* | 1 | R/W | |
| 0 | MSTPC0* | 1 | R/W | |

Note:* MSTPA7 is a readable/writable bit with an initial value of 0. The write value should always be 0.

MSTPA2, MSTPB6, MSTPB4 to MSTPB0, MSTPC7 to MSTPC5, MSTPC1, and MSTPC0 are readable/writable bits with an initial value of 1. The write value should always be 1.

## 21.2 Medium-Speed Mode

When the SCK0 to SCK2 bits in SCKCR are set to 1, the operating mode changes to medium-speed mode as soon as the current bus cycle ends. In medium-speed mode, the CPU operates on the operating clock ($\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$, or $\phi/32$) specified by the SCK0 to SCK2 bits. Bus masters (DTC) other than the CPU also operate in medium-speed mode. On-chip peripheral modules other than bus masters always operate on the high-speed clock ($\phi$).

In medium-speed mode, a bus access is executed in the specified number of states with respect to the bus master operating clock. For example, if $\phi/4$ is selected as the operating clock, on-chip memory is accessed in 4 states, and internal I/O registers in 8 states.

Medium-speed mode is cleared by clearing all of bits SCK0 to SCK2 to 0. A transition is made to high-speed mode and medium-speed mode is cleared at the end of the current bus cycle.

If a SLEEP instruction is executed when the SSBY bit in SBYCR is cleared to 0, a transition is made to sleep mode. When sleep mode is cleared by an interrupt, medium-speed mode is restored.

When the SLEEP instruction is executed with the SSBY bit = 1, operation shifts to the software standby mode. When software standby mode is cleared by an external interrupt, medium-speed mode is restored.

When the $\overline{\text{RES}}$ pin is set low and medium-speed mode is cancelled, operation shifts to the reset state. The same applies in the case of a reset caused by overflow of the watchdog timer.

When the $\overline{\text{STBY}}$ pin is driven low, a transition is made to hardware standby mode.

Figure 21.2 shows the timing for transition to and clearance of medium-speed mode.

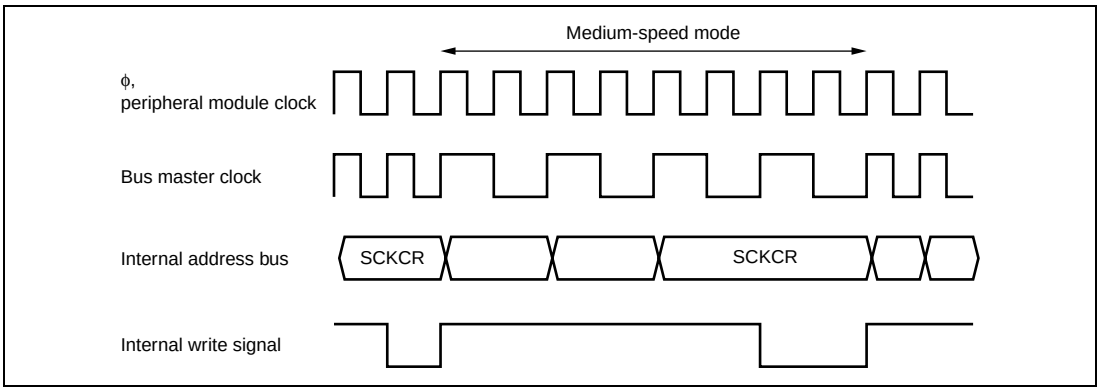



**Figure 21.2 Medium-Speed Mode Transition and Clearance Timing**

## 21.3 Sleep Mode

### 21.3.1 Transition to Sleep Mode

If SLEEP instruction is executed when the SBYCR SSBY bit = 0, the CPU enters the sleep mode. In sleep mode, CPU operation stops, however the contents of the CPU's internal registers are retained. Other peripheral modules do not stop.

### 21.3.2 Clearing Sleep Mode

Sleep mode is cleared by any interrupt, or signals at the $\overline{\text{RES}}$, or $\overline{\text{STBY}}$ pins.

- Exiting Sleep Mode by Interrupts:
  When an interrupt occurs, sleep mode is exited and interrupt exception processing starts. Sleep mode is not exited if the interrupt is disabled, or if interrupts other than NMI are masked by the CPU.
- Exiting Sleep Mode by $\overline{\text{RES}}$ pin:
  Setting the $\overline{\text{RES}}$ pin low level selects the reset state. After the stipulated reset input duration, driving the $\overline{\text{RES}}$ pin high level restart the CPU performing reset exception processing.
- Exiting Sleep Mode by $\overline{\text{STBY}}$ Pin:
  When the $\overline{\text{STBY}}$ pin level is driven low, a transition is made to hardware standby mode.

## 21.4 Software Standby Mode

### 21.4.1 Transition to Software Standby Mode

A transition is made to software standby mode if the SLEEP instruction is executed when the SBYCR SSBY bit is set to 1. In this mode, the CPU, on-chip peripheral modules, and oscillator, all stop. However, the contents of the CPU's internal registers, on-chip RAM data, and the states of on-chip peripheral modules other than the SCI, A/D converter, and the states of I/O ports, are retained. In this mode, the oscillator stops, and therefore power consumption is significantly reduced.

### 21.4.2 Clearing Software Standby Mode

Software standby mode is cleared by an external interrupt (NMI pin, or pins $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ5}}$), or by means of the $\overline{\text{RES}}$ pin or $\overline{\text{STBY}}$ pin.

- Clearing with an interrupt

  When an NMI or IRQ0 to IRQ5 interrupt request signal is input, clock oscillation starts, and after the time set in bits STS0 to STS2 in SBYCR has elapsed, stable clocks are supplied to the entire chip, software standby mode is cleared, and interrupt exception handling is started.

  When clearing software standby mode with an IRQ0 to IRQ5 interrupt, set the corresponding enable bit to 1 and ensure that no interrupt with a higher priority than interrupts IRQ0 to IRQ5 is generated. Software standby mode cannot be cleared if the interrupt has been masked on the CPU side or has been designated as a DTC activation source.
- Clearing with the $\overline{\text{RES}}$ pin

  When the $\overline{\text{RES}}$ pin is driven low, clock oscillation is started. At the same time as clock oscillation starts, clocks are supplied to the entire chip. Note that the $\overline{\text{RES}}$ pin must be held low until clock oscillation settles. When the $\overline{\text{RES}}$ pin goes high, the CPU begins reset exception handling.
- Clearing with the $\overline{\text{STBY}}$ pin

  When the $\overline{\text{STBY}}$ pin is driven low, a transition is made to hardware standby mode.

### 21.4.3 Setting Oscillation Stabilization Time after Clearing Software Standby Mode

Bits STS2 to STS0 in SBYCR should be set as described below.

- Using a Crystal Oscillator:
  Set bits STS0 to STS2 so that the standby time is at least 8 ms (the oscillation settling time).
  Table 21.3 shows the standby times for different operating frequencies and settings of bits STS0 to STS2.
- Using an External Clock
  The PLL circuit requires a time for settling. Set bits STS0 to STS2 so that the standby time is at least 2 ms(the oscillation settling time).

**Table 21.3 Oscillation Stabilization Time Settings**

| STS2 | STS1 | STS0 | Standby Time | 24 MHz | 20 MHz | 16 MHz | 12 MHz | 10 MHz | 8 MHz | 6 MHz | 4 MHz | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 8192 states | 0.34 | 0.41 | 0.51 | 0.68 | 0.8 | 1.0 | 1.3 | 2.0 | ms |
| | | 1 | 16384 states | 0.68 | 0.82 | 1.0 | 1.3 | 1.6 | 2.0 | 2.7 | 4.1 | |
| | 1 | 0 | 32768 states | 1.4 | 1.6 | 2.0 | 2.7 | 3.3 | 4.1 | 5.5 | 8.2 | |
| | | 1 | 65536 states | 2.7 | 3.3 | 4.1 | 5.5 | 6.6 | 8.2 | 10.9 | 16.4 | |
| 1 | 0 | 0 | 131072 states | 5.5 | 6.6 | 8.2 | 10.9 | 13.1 | 16.4 | 21.8 | 32.8 | |
| | | 1 | 262144 states | 10.9 | 13.1 | 16.4 | 21.8 | 26.2 | 32.8 | 43.6 | 65.6 | |
| | 1 | 0 | Reserved | — | — | — | — | — | — | — | — | — |
| | | 1 | 16 states* | 0.7 | 0.8 | 1.0 | 1.3 | 1.6 | 2.0 | 1.7 | 4.0 | µs |

: Recommended time setting

Note:* Cannot be used in this LSI.

### 21.4.4 Software Standby Mode Application Example

Figure 21.3 shows an example in which a transition is made to software standby mode at the falling edge on the NMI pin, and software standby mode is cleared at the rising edge on the NMI pin.

In this example, an NMI interrupt is accepted with the NMIEG bit in SYSCR cleared to 0 (falling edge specification), then the NMIEG bit is set to 1 (rising edge specification), the SSBY bit is set to 1, and a SLEEP instruction is executed, causing a transition to software standby mode.

Software standby mode is then cleared at the rising edge on the NMI pin.

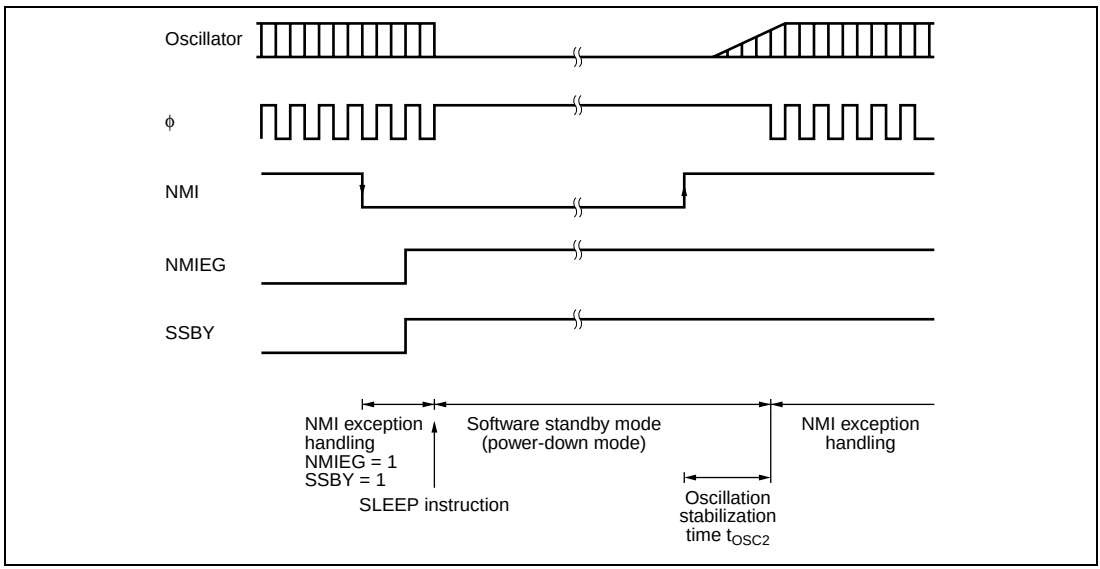



**Figure 21.3 Software Standby Mode Application Example**

## 21.5 Hardware Standby Mode

### 21.5.1 Transition to Hardware Standby Mode

When the $\overline{\text{STBY}}$ pin is driven low, a transition is made to hardware standby mode from any mode.

In hardware standby mode, all functions enter the reset state and stop operation, resulting in a significant reduction in power consumption. As long as the prescribed voltage is supplied, on-chip RAM data is retained. I/O ports are set to the high-impedance state.

In order to retain on-chip RAM data, the RAME bit in SYSCR should be cleared to 0 before driving the $\overline{\text{STBY}}$ pin low.

Do not change the state of the mode pins (MD0 to MD2) while this LSI is in hardware standby mode.

### 21.5.2 Clearing Hardware Standby Mode

Hardware standby mode is cleared by means of the $\overline{\text{STBY}}$ pin and the $\overline{\text{RES}}$ pin. When the $\overline{\text{STBY}}$ pin is driven high while the $\overline{\text{RES}}$ pin is low, the reset state is set and clock oscillation is started. Ensure that the $\overline{\text{RES}}$ pin is held low until the clock oscillator settles (at least 8 ms—the oscillation settling time—when using a crystal oscillator). When the $\overline{\text{RES}}$ pin is subsequently driven high, a transition is made to the program execution state via the reset exception handling state.

### 21.5.3 Hardware Standby Mode Timings

**Timing of Transition to Hardware Standby Mode**

1. To retain RAM contents with the RAME bit set to 1 in SYSCR
   Drive the $\overline{\text{RES}}$ signal low at least 10 states before the $\overline{\text{STBY}}$ signal goes low, as shown in figure 21.4. After $\overline{\text{STBY}}$ has gone low, $\overline{\text{RES}}$ has to wait for at least 0 ns before becoming high.

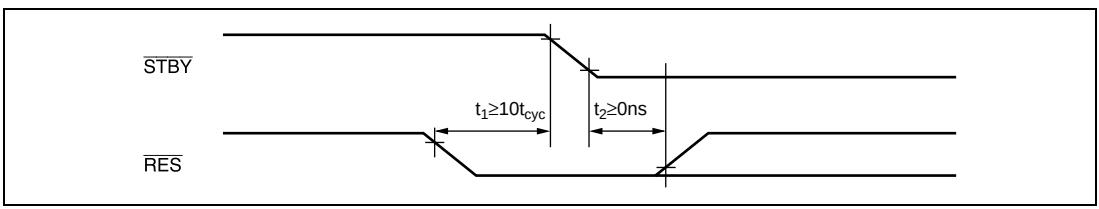



**Figure 21.4 Timing of Transition to Hardware Standby Mode**

2. To retain RAM contents with the RAME bit cleared to 0 in SYSCR, or when RAM contents do not need to be retained
   $\overline{\text{RES}}$ does not have to be driven low as in the above case.

**Timing of Recovery from Hardware Standby Mode**

Drive the $\overline{\text{RES}}$ signal low approximately 100 ns or more before $\overline{\text{STBY}}$ goes high to execute a power-on reset.

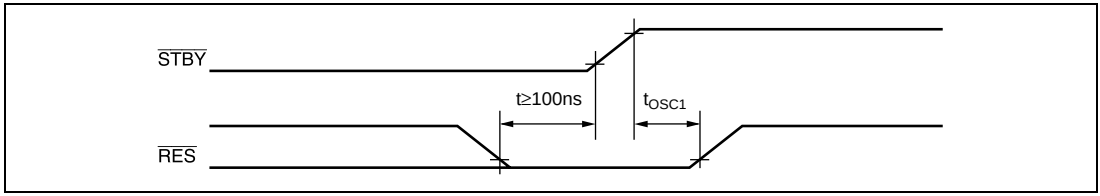



**Figure 21.5 Timing of Recovery from Hardware Standby Mode**

## 21.6 Module Stop Mode

### 21.6.1 Module Stop Mode

Module stop mode can be set for individual on-chip peripheral modules.

When the corresponding MSTP bit in MSTPCR is set to 1, module operation stops at the end of the bus cycle and a transition is made to module stop mode. The CPU continues operating independently.

When the corresponding MSTP bit is cleared to 0, module stop mode is cleared and the module starts operating at the end of the bus cycle. In module stop mode, the internal states of modules other than the SCI*, HCAN, and A/D converter are retained.

After reset clearance, all modules other than DTC are in module stop mode.

When an on-chip peripheral module is in module stop mode, read/write access to its registers is disabled.

Note: The internal states of some SCI registers are retained.

## 21.7 φ Clock Output Disabling Function

The output of the φ clock can be controlled by means of the PSTOP bit in SCKCR, and DDR for the corresponding port. When the PSTOP bit is set to 1, the φ clock stops at the end of the bus cycle, and φ output goes high. φ clock output is enabled when the PSTOP bit is cleared to 0. When DDR for the corresponding port is cleared to 0, φ clock output is disabled and input port mode is set. Table 21.4 shows the state of the φ pin in each processing state.

**Table 21.4 φ Pin State in Each Processing State**

| Register Settings | | | | | |
|---|---|---|---|---|---|
| DDR | PSTOP | Normal Mode | Sleep Mode | Software Standby Mode | Hardware Standby Mode |
| 0 | X | High impedance | High impedance | High impedance | High impedance |
| 1 | 0 | φ output | φ output | Fixed high | High impedance |
| 1 | 1 | Fixed high | Fixed high | Fixed high | High impedance |

## 21.8 Usage Notes

### 21.8.1 I/O Port Status

In software standby mode, I/O port states are retained. Therefore, there is no reduction in current consumption for the output current when a high-level signal is output.

### 21.8.2 Current Consumption during Oscillation Stabilization Wait Period

Current consumption increases during the oscillation settling wait period.

### 21.8.3 DTC Module Stop

Depending on the operating status of the DTC, MSTPA6 bit may not be set to 1. Setting of the DTC module stop mode should be carried out only when the respective module is not activated.

For details, refer to section 8, Data Transfer Controller (DTC).

### 21.8.4 On-Chip Peripheral Module Interrupt

Relevant interrupt operations cannot be performed in module stop mode. Consequently, if module stop mode is entered when an interrupt has been requested, it will not be possible to clear the CPU interrupt source or the DTC activation source.

Interrupts should therefore be disabled before entering module stop mode.

### 21.8.5 Writing to MSTPCR

MSTPCR should only be written to by the CPU.

# Section 22 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.
- When the address is 16-bit wide, the address of the upper byte is given in the list.
- Registers are classified by functional modules.
- The access size 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.
- Bit number in the bit-name column indicates that the whole register is allocated as a counter or for holding data.
- 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.

## 22.1 Register Addresses (Address Order)

The data-bus width column indicates the number of bits. The access-state column shows the number of states of the selected basic clock that is required for access to the register.

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Master control register | MCR | 8 | H′F800 | HCAN | 16 | 4 |
| General status register | GSR | 8 | H′F801 | HCAN | 16 | 4 |
| Bit configuration register | BCR | 16 | H′F802 | HCAN | 16 | 4 |
| Mailbox configuration register | MBCR | 16 | H′F804 | HCAN | 16 | 4 |
| Transmit wait register | TXPR | 16 | H′F806 | HCAN | 16 | 4 |
| Transmit wait cancel register | TXCR | 16 | H′F808 | HCAN | 16 | 4 |
| Transmit acknowledge register | TXACK | 16 | H′F80A | HCAN | 16 | 4 |
| Abort acknowledge register | ABACK | 16 | H′F80C | HCAN | 16 | 4 |
| Receive complete register | RXPR | 16 | H′F80E | HCAN | 16 | 4 |
| Remote request register | RFPR | 16 | H′F810 | HCAN | 16 | 4 |
| Interrupt register | IRR | 16 | H′F812 | HCAN | 16 | 4 |
| Mailbox interrupt mask register | MBIMR | 16 | H′F814 | HCAN | 16 | 4 |
| Interrupt mask register | IMR | 16 | H′F816 | HCAN | 16 | 4 |
| Receive error counter | REC | 8 | H′F818 | HCAN | 16 | 4 |
| Transmit error counter | TEC | 8 | H′F819 | HCAN | 16 | 4 |
| Unread message status register | UMSR | 16 | H′F81A | HCAN | 16 | 4 |
| Local acceptance filter mask L | LAFML | 16 | H′F81C | HCAN | 16 | 4 |
| Local acceptance filter mask H | LAFMH | 16 | H′F81E | HCAN | 16 | 4 |
| Message control 0[1] | MC0[1] | 8 | H′F820 | HCAN | 16 | 4 |
| Message control 0[2] | MC0[2] | 8 | H′F821 | HCAN | 16 | 4 |
| Message control 0[3] | MC0[3] | 8 | H′F822 | HCAN | 16 | 4 |
| Message control 0[4] | MC0[4] | 8 | H′F823 | HCAN | 16 | 4 |
| Message control 0[5] | MC0[5] | 8 | H′F824 | HCAN | 16 | 4 |
| Message control 0[6] | MC0[6] | 8 | H′F825 | HCAN | 16 | 4 |
| Message control 0[7] | MC0[7] | 8 | H′F826 | HCAN | 16 | 4 |
| Message control 0[8] | MC0[8] | 8 | H′F827 | HCAN | 16 | 4 |
| Message control 1[1] | MC1[1] | 8 | H′F828 | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message control 1[2] | MC1[2] | 8 | H′F829 | HCAN | 16 | 4 |
| Message control 1[3] | MC1[3] | 8 | H′F82A | HCAN | 16 | 4 |
| Message control 1[4] | MC1[4] | 8 | H′F82B | HCAN | 16 | 4 |
| Message control 1[5] | MC1[5] | 8 | H′F82C | HCAN | 16 | 4 |
| Message control 1[6] | MC1[6] | 8 | H′F82D | HCAN | 16 | 4 |
| Message control 1[7] | MC1[7] | 8 | H′F82E | HCAN | 16 | 4 |
| Message control 1[8] | MC1[8] | 8 | H′F82F | HCAN | 16 | 4 |
| Message control 2[1] | MC2[1] | 8 | H′F830 | HCAN | 16 | 4 |
| Message control 2[2] | MC2[2] | 8 | H′F831 | HCAN | 16 | 4 |
| Message control 2[3] | MC2[3] | 8 | H′F832 | HCAN | 16 | 4 |
| Message control 2[4] | MC2[4] | 8 | H′F833 | HCAN | 16 | 4 |
| Message control 2[5] | MC2[5] | 8 | H′F834 | HCAN | 16 | 4 |
| Message control 2[6] | MC2[6] | 8 | H′F835 | HCAN | 16 | 4 |
| Message control 2[7] | MC2[7] | 8 | H′F836 | HCAN | 16 | 4 |
| Message control 2[8] | MC2[8] | 8 | H′F837 | HCAN | 16 | 4 |
| Message control 3[1] | MC3[1] | 8 | H′F838 | HCAN | 16 | 4 |
| Message control 3[2] | MC3[2] | 8 | H′F839 | HCAN | 16 | 4 |
| Message control 3[3] | MC3[3] | 8 | H′F83A | HCAN | 16 | 4 |
| Message control 3[4] | MC3[4] | 8 | H′F83B | HCAN | 16 | 4 |
| Message control 3[5] | MC3[5] | 8 | H′F83C | HCAN | 16 | 4 |
| Message control 3[6] | MC3[6] | 8 | H′F83D | HCAN | 16 | 4 |
| Message control 3[7] | MC3[7] | 8 | H′F83E | HCAN | 16 | 4 |
| Message control 3[8] | MC3[8] | 8 | H′F83F | HCAN | 16 | 4 |
| Message control 4[1] | MC4[1] | 8 | H′F840 | HCAN | 16 | 4 |
| Message control 4[2] | MC4[2] | 8 | H′F841 | HCAN | 16 | 4 |
| Message control 4[3] | MC4[3] | 8 | H′F842 | HCAN | 16 | 4 |
| Message control 4[4] | MC4[4] | 8 | H′F843 | HCAN | 16 | 4 |
| Message control 4[5] | MC4[5] | 8 | H′F844 | HCAN | 16 | 4 |
| Message control 4[6] | MC4[6] | 8 | H′F845 | HCAN | 16 | 4 |
| Message control 4[7] | MC4[7] | 8 | H′F846 | HCAN | 16 | 4 |
| Message control 4[8] | MC4[8] | 8 | H′F847 | HCAN | 16 | 4 |
| Message control 5[1] | MC5[1] | 8 | H′F848 | HCAN | 16 | 4 |
| Message control 5[2] | MC5[2] | 8 | H′F849 | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message control 5[3] | MC5[3] | 8 | H′F84A | HCAN | 16 | 4 |
| Message control 5[4] | MC5[4] | 8 | H′F84B | HCAN | 16 | 4 |
| Message control 5[5] | MC5[5] | 8 | H′F84C | HCAN | 16 | 4 |
| Message control 5[6] | MC5[6] | 8 | H′F84D | HCAN | 16 | 4 |
| Message control 5[7] | MC5[7] | 8 | H′F84E | HCAN | 16 | 4 |
| Message control 5[8] | MC5[8] | 8 | H′F84F | HCAN | 16 | 4 |
| Message control 6[1] | MC6[1] | 8 | H′F850 | HCAN | 16 | 4 |
| Message control 6[2] | MC6[2] | 8 | H′F851 | HCAN | 16 | 4 |
| Message control 6[3] | MC6[3] | 8 | H′F852 | HCAN | 16 | 4 |
| Message control 6[4] | MC6[4] | 8 | H′F853 | HCAN | 16 | 4 |
| Message control 6[5] | MC6[5] | 8 | H′F854 | HCAN | 16 | 4 |
| Message control 6[6] | MC6[6] | 8 | H′F855 | HCAN | 16 | 4 |
| Message control 6[7] | MC6[7] | 8 | H′F856 | HCAN | 16 | 4 |
| Message control 6[8] | MC6[8] | 8 | H′F857 | HCAN | 16 | 4 |
| Message control 7[1] | MC7[1] | 8 | H′F858 | HCAN | 16 | 4 |
| Message control 7[2] | MC7[2] | 8 | H′F859 | HCAN | 16 | 4 |
| Message control 7[3] | MC7[3] | 8 | H′F85A | HCAN | 16 | 4 |
| Message control 7[4] | MC7[4] | 8 | H′F85B | HCAN | 16 | 4 |
| Message control 7[5] | MC7[5] | 8 | H′F85C | HCAN | 16 | 4 |
| Message control 7[6] | MC7[6] | 8 | H′F85D | HCAN | 16 | 4 |
| Message control 7[7] | MC7[7] | 8 | H′F85E | HCAN | 16 | 4 |
| Message control 7[8] | MC7[8] | 8 | H′F85F | HCAN | 16 | 4 |
| Message control 8[1] | MC8[1] | 8 | H′F860 | HCAN | 16 | 4 |
| Message control 8[2] | MC8[2] | 8 | H′F861 | HCAN | 16 | 4 |
| Message control 8[3] | MC8[3] | 8 | H′F862 | HCAN | 16 | 4 |
| Message control 8[4] | MC8[4] | 8 | H′F863 | HCAN | 16 | 4 |
| Message control 8[5] | MC8[5] | 8 | H′F864 | HCAN | 16 | 4 |
| Message control 8[6] | MC8[6] | 8 | H′F865 | HCAN | 16 | 4 |
| Message control 8[7] | MC8[7] | 8 | H′F866 | HCAN | 16 | 4 |
| Message control 8[8] | MC8[8] | 8 | H′F867 | HCAN | 16 | 4 |
| Message control 9[1] | MC9[1] | 8 | H′F868 | HCAN | 16 | 4 |
| Message control 9[2] | MC9[2] | 8 | H′F869 | HCAN | 16 | 4 |
| Message control 9[3] | MC9[3] | 8 | H′F86A | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message control 9[4] | MC9[4] | 8 | H′F86B | HCAN | 16 | 4 |
| Message control 9[5] | MC9[5] | 8 | H′F86C | HCAN | 16 | 4 |
| Message control 9[6] | MC9[6] | 8 | H′F86D | HCAN | 16 | 4 |
| Message control 9[7] | MC9[7] | 8 | H′F86E | HCAN | 16 | 4 |
| Message control 9[8] | MC9[8] | 8 | H′F86F | HCAN | 16 | 4 |
| Message control 10[1] | MC10[1] | 8 | H′F870 | HCAN | 16 | 4 |
| Message control 10[2] | MC10[2] | 8 | H′F871 | HCAN | 16 | 4 |
| Message control 10[3] | MC10[3] | 8 | H′F872 | HCAN | 16 | 4 |
| Message control 10[4] | MC10[4] | 8 | H′F873 | HCAN | 16 | 4 |
| Message control 10[5] | MC10[5] | 8 | H′F874 | HCAN | 16 | 4 |
| Message control 10[6] | MC10[6] | 8 | H′F875 | HCAN | 16 | 4 |
| Message control 10[7] | MC10[7] | 8 | H′F876 | HCAN | 16 | 4 |
| Message control 10[8] | MC10[8] | 8 | H′F877 | HCAN | 16 | 4 |
| Message control 11[1] | MC11[1] | 8 | H′F878 | HCAN | 16 | 4 |
| Message control 11[2] | MC11[2] | 8 | H′F879 | HCAN | 16 | 4 |
| Message control 11[3] | MC11[3] | 8 | H′F87A | HCAN | 16 | 4 |
| Message control 11[4] | MC11[4] | 8 | H′F87B | HCAN | 16 | 4 |
| Message control 11[5] | MC11[5] | 8 | H′F87C | HCAN | 16 | 4 |
| Message control 11[6] | MC11[6] | 8 | H′F87D | HCAN | 16 | 4 |
| Message control 11[7] | MC11[7] | 8 | H′F87E | HCAN | 16 | 4 |
| Message control 11[8] | MC11[8] | 8 | H′F87F | HCAN | 16 | 4 |
| Message control 12[1] | MC12[1] | 8 | H′F880 | HCAN | 16 | 4 |
| Message control 12[2] | MC12[2] | 8 | H′F881 | HCAN | 16 | 4 |
| Message control 12[3] | MC12[3] | 8 | H′F882 | HCAN | 16 | 4 |
| Message control 12[4] | MC12[4] | 8 | H′F883 | HCAN | 16 | 4 |
| Message control 12[5] | MC12[5] | 8 | H′F884 | HCAN | 16 | 4 |
| Message control 12[6] | MC12[6] | 8 | H′F885 | HCAN | 16 | 4 |
| Message control 12[7] | MC12[7] | 8 | H′F886 | HCAN | 16 | 4 |
| Message control 12[8] | MC12[8] | 8 | H′F887 | HCAN | 16 | 4 |
| Message control 13[1] | MC13[1] | 8 | H′F888 | HCAN | 16 | 4 |
| Message control 13[2] | MC13[2] | 8 | H′F889 | HCAN | 16 | 4 |
| Message control 13[3] | MC13[3] | 8 | H′F88A | HCAN | 16 | 4 |
| Message control 13[4] | MC13[4] | 8 | H′F88B | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message control 13[5] | MC13[5] | 8 | H′F88C | HCAN | 16 | 4 |
| Message control 13[6] | MC13[6] | 8 | H′F88D | HCAN | 16 | 4 |
| Message control 13[7] | MC13[7] | 8 | H′F88E | HCAN | 16 | 4 |
| Message control 13[8] | MC13[8] | 8 | H′F88F | HCAN | 16 | 4 |
| Message control 14[1] | MC14[1] | 8 | H′F890 | HCAN | 16 | 4 |
| Message control 14[2] | MC14[2] | 8 | H′F891 | HCAN | 16 | 4 |
| Message control 14[3] | MC14[3] | 8 | H′F892 | HCAN | 16 | 4 |
| Message control 14[4] | MC14[4] | 8 | H′F893 | HCAN | 16 | 4 |
| Message control 14[5] | MC14[5] | 8 | H′F894 | HCAN | 16 | 4 |
| Message control 14[6] | MC14[6] | 8 | H′F895 | HCAN | 16 | 4 |
| Message control 14[7] | MC14[7] | 8 | H′F896 | HCAN | 16 | 4 |
| Message control 14[8] | MC14[8] | 8 | H′F897 | HCAN | 16 | 4 |
| Message control 15[1] | MC15[1] | 8 | H′F898 | HCAN | 16 | 4 |
| Message control 15[2] | MC15[2] | 8 | H′F899 | HCAN | 16 | 4 |
| Message control 15[3] | MC15[3] | 8 | H′F89A | HCAN | 16 | 4 |
| Message control 15[4] | MC15[4] | 8 | H′F89B | HCAN | 16 | 4 |
| Message control 15[5] | MC15[5] | 8 | H′F89C | HCAN | 16 | 4 |
| Message control 15[6] | MC15[6] | 8 | H′F89D | HCAN | 16 | 4 |
| Message control 15[7] | MC15[7] | 8 | H′F89E | HCAN | 16 | 4 |
| Message control 15[8] | MC15[8] | 8 | H′F89F | HCAN | 16 | 4 |
| Message data 0[1] | MD0[1] | 8 | H′F8B0 | HCAN | 16 | 4 |
| Message data 0[2] | MD0[2] | 8 | H′F8B1 | HCAN | 16 | 4 |
| Message data 0[3] | MD0[3] | 8 | H′F8B2 | HCAN | 16 | 4 |
| Message data 0[4] | MD0[4] | 8 | H′F8B3 | HCAN | 16 | 4 |
| Message data 0[5] | MD0[5] | 8 | H′F8B4 | HCAN | 16 | 4 |
| Message data 0[6] | MD0[6] | 8 | H′F8B5 | HCAN | 16 | 4 |
| Message data 0[7] | MD0[7] | 8 | H′F8B6 | HCAN | 16 | 4 |
| Message data 0[8] | MD0[8] | 8 | H′F8B7 | HCAN | 16 | 4 |
| Message data 1[1] | MD1[1] | 8 | H′F8B8 | HCAN | 16 | 4 |
| Message data 1[2] | MD1[2] | 8 | H′F8B9 | HCAN | 16 | 4 |
| Message data 1[3] | MD1[3] | 8 | H′F8BA | HCAN | 16 | 4 |
| Message data 1[4] | MD1[4] | 8 | H′F8BB | HCAN | 16 | 4 |
| Message data 1[5] | MD1[5] | 8 | H′F8BC | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message data 1[6] | MD1[6] | 8 | H′F8BD | HCAN | 16 | 4 |
| Message data 1[7] | MD1[7] | 8 | H′F8BE | HCAN | 16 | 4 |
| Message data 1[8] | MD1[8] | 8 | H′F8BF | HCAN | 16 | 4 |
| Message data 2[1] | MD2[1] | 8 | H′F8C0 | HCAN | 16 | 4 |
| Message data 2[2] | MD2[2] | 8 | H′F8C1 | HCAN | 16 | 4 |
| Message data 2[3] | MD2[3] | 8 | H′F8C2 | HCAN | 16 | 4 |
| Message data 2[4] | MD2[4] | 8 | H′F8C3 | HCAN | 16 | 4 |
| Message data 2[5] | MD2[5] | 8 | H′F8C4 | HCAN | 16 | 4 |
| Message data 2[6] | MD2[6] | 8 | H′F8C5 | HCAN | 16 | 4 |
| Message data 2[7] | MD2[7] | 8 | H′F8C6 | HCAN | 16 | 4 |
| Message data 2[8] | MD2[8] | 8 | H′F8C7 | HCAN | 16 | 4 |
| Message data 3[1] | MD3[1] | 8 | H′F8C8 | HCAN | 16 | 4 |
| Message data 3[2] | MD3[2] | 8 | H′F8C9 | HCAN | 16 | 4 |
| Message data 3[3] | MD3[3] | 8 | H′F8CA | HCAN | 16 | 4 |
| Message data 3[4] | MD3[4] | 8 | H′F8CB | HCAN | 16 | 4 |
| Message data 3[5] | MD3[5] | 8 | H′F8CC | HCAN | 16 | 4 |
| Message data 3[6] | MD3[6] | 8 | H′F8CD | HCAN | 16 | 4 |
| Message data 3[7] | MD3[7] | 8 | H′F8CE | HCAN | 16 | 4 |
| Message data 3[8] | MD3[8] | 8 | H′F8CF | HCAN | 16 | 4 |
| Message data 4[1] | MD4[1] | 8 | H′F8D0 | HCAN | 16 | 4 |
| Message data 4[2] | MD4[2] | 8 | H′F8D1 | HCAN | 16 | 4 |
| Message data 4[3] | MD4[3] | 8 | H′F8D2 | HCAN | 16 | 4 |
| Message data 4[4] | MD4[4] | 8 | H′F8D3 | HCAN | 16 | 4 |
| Message data 4[5] | MD4[5] | 8 | H′F8D4 | HCAN | 16 | 4 |
| Message data 4[6] | MD4[6] | 8 | H′F8D5 | HCAN | 16 | 4 |
| Message data 4[7] | MD4[7] | 8 | H′F8D6 | HCAN | 16 | 4 |
| Message data 4[8] | MD4[8] | 8 | H′F8D7 | HCAN | 16 | 4 |
| Message data 5[1] | MD5[1] | 8 | H′F8D8 | HCAN | 16 | 4 |
| Message data 5[2] | MD5[2] | 8 | H′F8D9 | HCAN | 16 | 4 |
| Message data 5[3] | MD5[3] | 8 | H′F8DA | HCAN | 16 | 4 |
| Message data 5[4] | MD5[4] | 8 | H′F8DB | HCAN | 16 | 4 |
| Message data 5[5] | MD5[5] | 8 | H′F8DC | HCAN | 16 | 4 |
| Message data 5[6] | MD5[6] | 8 | H′F8DD | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message data 5[7] | MD5[7] | 8 | H′F8DE | HCAN | 16 | 4 |
| Message data 5[8] | MD5[8] | 8 | H′F8DF | HCAN | 16 | 4 |
| Message data 6[1] | MD6[1] | 8 | H′F8E0 | HCAN | 16 | 4 |
| Message data 6[2] | MD6[2] | 8 | H′F8E1 | HCAN | 16 | 4 |
| Message data 6[3] | MD6[3] | 8 | H′F8E2 | HCAN | 16 | 4 |
| Message data 6[4] | MD6[4] | 8 | H′F8E3 | HCAN | 16 | 4 |
| Message data 6[5] | MD6[5] | 8 | H′F8E4 | HCAN | 16 | 4 |
| Message data 6[6] | MD6[6] | 8 | H′F8E5 | HCAN | 16 | 4 |
| Message data 6[7] | MD6[7] | 8 | H′F8E6 | HCAN | 16 | 4 |
| Message data 6[8] | MD6[8] | 8 | H′F8E7 | HCAN | 16 | 4 |
| Message data 7[1] | MD7[1] | 8 | H′F8E8 | HCAN | 16 | 4 |
| Message data 7[2] | MD7[2] | 8 | H′F8E9 | HCAN | 16 | 4 |
| Message data 7[3] | MD7[3] | 8 | H′F8EA | HCAN | 16 | 4 |
| Message data 7[4] | MD7[4] | 8 | H′F8EB | HCAN | 16 | 4 |
| Message data 7[5] | MD7[5] | 8 | H′F8EC | HCAN | 16 | 4 |
| Message data 7[6] | MD7[6] | 8 | H′F8ED | HCAN | 16 | 4 |
| Message data 7[7] | MD7[7] | 8 | H′F8EE | HCAN | 16 | 4 |
| Message data 7[8] | MD7[8] | 8 | H′F8EF | HCAN | 16 | 4 |
| Message data 8[1] | MD8[1] | 8 | H′F8F0 | HCAN | 16 | 4 |
| Message data 8[2] | MD8[2] | 8 | H′F8F1 | HCAN | 16 | 4 |
| Message data 8[3] | MD8[3] | 8 | H′F8F2 | HCAN | 16 | 4 |
| Message data 8[4] | MD8[4] | 8 | H′F8F3 | HCAN | 16 | 4 |
| Message data 8[5] | MD8[5] | 8 | H′F8F4 | HCAN | 16 | 4 |
| Message data 8[6] | MD8[6] | 8 | H′F8F5 | HCAN | 16 | 4 |
| Message data 8[7] | MD8[7] | 8 | H′F8F6 | HCAN | 16 | 4 |
| Message data 8[8] | MD8[8] | 8 | H′F8F7 | HCAN | 16 | 4 |
| Message data 9[1] | MD9[1] | 8 | H′F8F8 | HCAN | 16 | 4 |
| Message data 9[2] | MD9[2] | 8 | H′F8F9 | HCAN | 16 | 4 |
| Message data 9[3] | MD9[3] | 8 | H′F8FA | HCAN | 16 | 4 |
| Message data 9[4] | MD9[4] | 8 | H′F8FB | HCAN | 16 | 4 |
| Message data 9[5] | MD9[5] | 8 | H′F8FC | HCAN | 16 | 4 |
| Message data 9[6] | MD9[6] | 8 | H′F8FD | HCAN | 16 | 4 |
| Message data 9[7] | MD9[7] | 8 | H′F8FE | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message data 9[8] | MD9[8] | 8 | H′F8FF | HCAN | 16 | 4 |
| Message data 10[1] | MD10[1] | 8 | H′F900 | HCAN | 16 | 4 |
| Message data 10[2] | MD10[2] | 8 | H′F901 | HCAN | 16 | 4 |
| Message data 10[3] | MD10[3] | 8 | H′F902 | HCAN | 16 | 4 |
| Message data 10[4] | MD10[4] | 8 | H′F903 | HCAN | 16 | 4 |
| Message data 10[5] | MD10[5] | 8 | H′F904 | HCAN | 16 | 4 |
| Message data 10[6] | MD10[6] | 8 | H′F905 | HCAN | 16 | 4 |
| Message data 10[7] | MD10[7] | 8 | H′F906 | HCAN | 16 | 4 |
| Message data 10[8] | MD10[8] | 8 | H′F907 | HCAN | 16 | 4 |
| Message data 11[1] | MD11[1] | 8 | H′F908 | HCAN | 16 | 4 |
| Message data 11[2] | MD11[2] | 8 | H′F909 | HCAN | 16 | 4 |
| Message data 11[3] | MD11[3] | 8 | H′F90A | HCAN | 16 | 4 |
| Message data 11[4] | MD11[4] | 8 | H′F90B | HCAN | 16 | 4 |
| Message data 11[5] | MD11[5] | 8 | H′F90C | HCAN | 16 | 4 |
| Message data 11[6] | MD11[6] | 8 | H′F90D | HCAN | 16 | 4 |
| Message data 11[7] | MD11[7] | 8 | H′F90E | HCAN | 16 | 4 |
| Message data 11[8] | MD11[8] | 8 | H′F90F | HCAN | 16 | 4 |
| Message data 12[1] | MD12[1] | 8 | H′F910 | HCAN | 16 | 4 |
| Message data 12[2] | MD12[2] | 8 | H′F911 | HCAN | 16 | 4 |
| Message data 12[3] | MD12[3] | 8 | H′F912 | HCAN | 16 | 4 |
| Message data 12[4] | MD12[4] | 8 | H′F913 | HCAN | 16 | 4 |
| Message data 12[5] | MD12[5] | 8 | H′F914 | HCAN | 16 | 4 |
| Message data 12[6] | MD12[6] | 8 | H′F915 | HCAN | 16 | 4 |
| Message data 12[7] | MD12[7] | 8 | H′F916 | HCAN | 16 | 4 |
| Message data 12[8] | MD12[8] | 8 | H′F917 | HCAN | 16 | 4 |
| Message data 13[1] | MD13[1] | 8 | H′F918 | HCAN | 16 | 4 |
| Message data 13[2] | MD13[2] | 8 | H′F919 | HCAN | 16 | 4 |
| Message data 13[3] | MD13[3] | 8 | H′F91A | HCAN | 16 | 4 |
| Message data 13[4] | MD13[4] | 8 | H′F91B | HCAN | 16 | 4 |
| Message data 13[5] | MD13[5] | 8 | H′F91C | HCAN | 16 | 4 |
| Message data 13[6] | MD13[6] | 8 | H′F91D | HCAN | 16 | 4 |
| Message data 13[7] | MD13[7] | 8 | H′F91E | HCAN | 16 | 4 |
| Message data 13[8] | MD13[8] | 8 | H′F91F | HCAN | 16 | 4 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Message data 14[1] | MD14[1] | 8 | H′F920 | HCAN | 16 | 4 |
| Message data 14[2] | MD14[2] | 8 | H′F921 | HCAN | 16 | 4 |
| Message data 14[3] | MD14[3] | 8 | H′F922 | HCAN | 16 | 4 |
| Message data 14[4] | MD14[4] | 8 | H′F923 | HCAN | 16 | 4 |
| Message data 14[5] | MD14[5] | 8 | H′F924 | HCAN | 16 | 4 |
| Message data 14[6] | MD14[6] | 8 | H′F925 | HCAN | 16 | 4 |
| Message data 14[7] | MD14[7] | 8 | H′F926 | HCAN | 16 | 4 |
| Message data 14[8] | MD14[8] | 8 | H′F927 | HCAN | 16 | 4 |
| Message data 15[1] | MD15[1] | 8 | H′F928 | HCAN | 16 | 4 |
| Message data 15[2] | MD15[2] | 8 | H′F929 | HCAN | 16 | 4 |
| Message data 15[3] | MD15[3] | 8 | H′F92A | HCAN | 16 | 4 |
| Message data 15[4] | MD15[4] | 8 | H′F92B | HCAN | 16 | 4 |
| Message data 15[5] | MD15[5] | 8 | H′F92C | HCAN | 16 | 4 |
| Message data 15[6] | MD15[6] | 8 | H′F92D | HCAN | 16 | 4 |
| Message data 15[7] | MD15[7] | 8 | H′F92E | HCAN | 16 | 4 |
| Message data 15[8] | MD15[8] | 8 | H′F92F | HCAN | 16 | 4 |
| HCAN monitor register | HCANMON | 8 | H′FA00 | HCAN | 16 | 4 |
| SS control register H_0 | SSCRH_0 | 8 | H′FB00 | SSU_0 | 16 | 3 |
| SS control register L_0 | SSCRL_0 | 8 | H′FB01 | SSU_0 | 16 | 3 |
| SS mode register_0 | SSMR_0 | 8 | H′FB02 | SSU_0 | 16 | 3 |
| SS enable register_0 | SSER_0 | 8 | H′FB03 | SSU_0 | 16 | 3 |
| SS status register_0 | SSSR_0 | 8 | H′FB04 | SSU_0 | 16 | 3 |
| SS transmit data register 0_0 | SSTDR0_0 | 8 | H′FB06 | SSU_0 | 16 | 3 |
| SS transmit data register 1_0 | SSTDR1_0 | 8 | H′FB07 | SSU_0 | 16 | 3 |
| SS transmit data register 2_0 | SSTDR2_0 | 8 | H′FB08 | SSU_0 | 16 | 3 |
| SS transmit data register 3_0 | SSTDR3_0 | 8 | H′FB09 | SSU_0 | 16 | 3 |
| SS receive data register 0_0 | SSRDR0_0 | 8 | H′FB0A | SSU_0 | 16 | 3 |
| SS receive data register 1_0 | SSRDR1_0 | 8 | H′FB0B | SSU_0 | 16 | 3 |
| SS receive data register 2_0 | SSRDR2_0 | 8 | H′FB0C | SSU_0 | 16 | 3 |
| SS receive data register 3_0 | SSRDR3_0 | 8 | H′FB0D | SSU_0 | 16 | 3 |
| SS control register H_1 | SSCRH_1 | 8 | H′FB10 | SSU_1 | 16 | 3 |
| SS control register L_1 | SSCRL_1 | 8 | H′FB11 | SSU_1 | 16 | 3 |
| SS mode register_1 | SSMR_1 | 8 | H′FB12 | SSU_1 | 16 | 3 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| SS enable register_1 | SSER_1 | 8 | H′FB13 | SSU_1 | 16 | 3 |
| SS status register_1 | SSSR_1 | 8 | H′FB14 | SSU_1 | 16 | 3 |
| SS transmit data register 0_1 | SSTDR0_1 | 8 | H′FB16 | SSU_1 | 16 | 3 |
| SS transmit data register 1_1 | SSTDR1_1 | 8 | H′FB17 | SSU_1 | 16 | 3 |
| SS transmit data register 2_1 | SSTDR2_1 | 8 | H′FB18 | SSU_1 | 16 | 3 |
| SS transmit data register 3_1 | SSTDR3_1 | 8 | H′FB19 | SSU_1 | 16 | 3 |
| SS receive data register 0_1 | SSRDR0_1 | 8 | H′FB1A | SSU_1 | 16 | 3 |
| SS receive data register 1_1 | SSRDR1_1 | 8 | H′FB1B | SSU_1 | 16 | 3 |
| SS receive data register 2_1 | SSRDR2_1 | 8 | H′FB1C | SSU_1 | 16 | 3 |
| SS receive data register 3_1 | SSRDR3_1 | 8 | H′FB1D | SSU_1 | 16 | 3 |
| Port D realtime input data register | PDRTIDR | 8 | H′FB40 | PORT | 8 | 3 |
| Timer control register_2 | TCR_2 | 8 | H′FDC0 | TMR_2 | 8 | 2 |
| Timer control register_3 | TCR_3 | 8 | H′FDC1 | TMR_3 | 8 | 2 |
| Timer control/status register_2 | TCSR_2 | 8 | H′FDC2 | TMR_2 | 8 | 2 |
| Timer control/status register_3 | TCSR_3 | 8 | H′FDC3 | TMR_3 | 8 | 2 |
| Timer constant register A_2 | TCORA_2 | 8 | H′FDC4 | TMR_2 | 8 | 2 |
| Timer constant register A_3 | TCORA_3 | 8 | H′FDC5 | TMR_3 | 8 | 2 |
| Timer constant register B_2 | TCORB_2 | 8 | H′FDC6 | TMR_2 | 8 | 2 |
| Timer constant register B_3 | TCORB_3 | 8 | H′FDC7 | TMR_3 | 8 | 2 |
| Timer counter_2 | TCNT_2 | 8 | H′FDC8 | TMR_2 | 8 | 2 |
| Timer counter_3 | TCNT_3 | 8 | H′FDC9 | TMR_3 | 8 | 2 |
| Standby control register | SBYCR | 8 | H′FDE4 | SYSTEM | 8 | 2 |
| System control register | SYSCR | 8 | H′FDE5 | SYSTEM | 8 | 2 |
| System clock control register | SCKCR | 8 | H′FDE6 | SYSTEM | 8 | 2 |
| Mode control register | MDCR | 8 | H′FDE7 | SYSTEM | 8 | 2 |
| Module stop control register A | MSTPCRA | 8 | H′FDE8 | SYSTEM | 8 | 2 |
| Module stop control register B | MSTPCRB | 8 | H′FDE9 | SYSTEM | 8 | 2 |
| Module stop control register C | MSTPCRC | 8 | H′FDEA | SYSTEM | 8 | 2 |
| Low-power control register | LPWRCR | 8 | H′FDEC | SYSTEM | 8 | 2 |
| Break address register A | BARA | 32 | H′FE00 | PBC | 32 | 2 |
| Break address register B | BARB | 32 | H′FE04 | PBC | 32 | 2 |
| Break control register A | BCRA | 8 | H′FE08 | PBC | 8 | 2 |
| Break control register B | BCRB | 8 | H′FE09 | PBC | 8 | 2 |
| IRQ sense control register H | ISCRH | 8 | H′FE12 | INT | 8 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| IRQ sense control register L | ISCRL | 8 | H′FE13 | INT | 8 | 2 |
| IRQ enable register | IER | 8 | H′FE14 | INT | 8 | 2 |
| IRQ status register | ISR | 8 | H′FE15 | INT | 8 | 2 |
| DTC enable register A | DTCERA | 8 | H′FE16 | DTC | 8 | 2 |
| DTC enable register B | DTCERB | 8 | H′FE17 | DTC | 8 | 2 |
| DTC enable register C | DTCERC | 8 | H′FE18 | DTC | 8 | 2 |
| DTC enable register D | DTCERD | 8 | H′FE19 | DTC | 8 | 2 |
| DTC enable register E | DTCERE | 8 | H′FE1A | DTC | 8 | 2 |
| DTC enable register F | DTCERF | 8 | H′FE1B | DTC | 8 | 2 |
| DTC enable register G | DTCERG | 8 | H′FE1C | DTC | 8 | 2 |
| DTC vector register | DTVECR | 8 | H′FE1F | DTC | 8 | 2 |
| PPG output control register | PCR | 8 | H′FE26 | PPG | 8 | 2 |
| PPG output mode register | PMR | 8 | H′FE27 | PPG | 8 | 2 |
| Next data enable register H | NDERH | 8 | H′FE28 | PPG | 8 | 2 |
| Next data enable register L | NDERL | 8 | H′FE29 | PPG | 8 | 2 |
| Output data register H | PODRH | 8 | H′FE2A | PPG | 8 | 2 |
| Output data register L | PODRL | 8 | H′FE2B | PPG | 8 | 2 |
| Next data register H | NDRH | 8 | H′FE2C | PPG | 8 | 2 |
| Next data register L | NDRL | 8 | H′FE2D | PPG | 8 | 2 |
| Next data register H | NDRH | 8 | H′FE2E | PPG | 8 | 2 |
| Next data register L | NDRL | 8 | H′FE2F | PPG | 8 | 2 |
| Port 1 data direction register | P1DDR | 8 | H′FE30 | PORT | 8 | 2 |
| Port 3 data direction register | P3DDR | 8 | H′FE32 | PORT | 8 | 2 |
| Port 7 data direction register | P7DDR | 8 | H′FE36 | PORT | 8 | 2 |
| Port A data direction register | PADDR | 8 | H′FE39 | PORT | 8 | 2 |
| Port B data direction register | PBDDR | 8 | H′FE3A | PORT | 8 | 2 |
| Port C data direction register | PCDDR | 8 | H′FE3B | PORT | 8 | 2 |
| Port D data direction register | PDDDR | 8 | H′FE3C | PORT | 8 | 2 |
| Port F data direction register | PFDDR | 8 | H′FE3E | PORT | 8 | 2 |
| Port A pull-up MOS control register | PAPCR | 8 | H′FE40 | PORT | 8 | 2 |
| Port B pull-up MOS control register | PBPCR | 8 | H′FE41 | PORT | 8 | 2 |
| Port C pull-up MOS control register | PCPCR | 8 | H′FE42 | PORT | 8 | 2 |
| Port D pull-up MOS control register | PDPCR | 8 | H′FE43 | PORT | 8 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Port 3 open drain control register | P3ODR | 8 | H′FE46 | PORT | 8 | 2 |
| Port A open drain control register | PAODR | 8 | H′FE47 | PORT | 8 | 2 |
| Port B open drain control register | PBODR | 8 | H′FE48 | PORT | 8 | 2 |
| Port C open drain control register | PCODR | 8 | H′FE49 | PORT | 8 | 2 |
| Timer control register_3 | TCR_3 | 8 | H′FE80 | TPU_3 | 16 | 2 |
| Timer mode register_3 | TMDR_3 | 8 | H′FE81 | TPU_3 | 16 | 2 |
| Timer I/O control register H_3 | TIORH_3 | 8 | H′FE82 | TPU_3 | 16 | 2 |
| Timer I/O control register L_3 | TIORL_3 | 8 | H′FE83 | TPU_3 | 16 | 2 |
| Timer interrupt enable register_3 | TIER_3 | 8 | H′FE84 | TPU_3 | 16 | 2 |
| Timer status register_3 | TSR_3 | 8 | H′FE85 | TPU_3 | 16 | 2 |
| Timer counter H_3 | TCNTH_3 | 8 | H′FE86 | TPU_3 | 16 | 2 |
| Timer counter L_3 | TCNTL_3 | 8 | H′FE87 | TPU_3 | 16 | 2 |
| Timer general register AH_3 | TGRAH_3 | 8 | H′FE88 | TPU_3 | 16 | 2 |
| Timer general register AL_3 | TGRAL_3 | 8 | H′FE89 | TPU_3 | 16 | 2 |
| Timer general register BH_3 | TGRBH_3 | 8 | H′FE8A | TPU_3 | 16 | 2 |
| Timer general register BL_3 | TGRBL_3 | 8 | H′FE8B | TPU_3 | 16 | 2 |
| Timer general register CH_3 | TGRCH_3 | 8 | H′FE8C | TPU_3 | 16 | 2 |
| Timer general register CL_3 | TGRCL_3 | 8 | H′FE8D | TPU_3 | 16 | 2 |
| Timer general register DH_3 | TGRDH_3 | 8 | H′FE8E | TPU_3 | 16 | 2 |
| Timer general register DL_3 | TGRDL_3 | 8 | H′FE8F | TPU_3 | 16 | 2 |
| Timer control register_4 | TCR_4 | 8 | H′FE90 | TPU_4 | 16 | 2 |
| Timer mode register_4 | TMDR_4 | 8 | H′FE91 | TPU_4 | 16 | 2 |
| Timer I/O control register_4 | TIOR_4 | 8 | H′FE92 | TPU_4 | 16 | 2 |
| Timer interrupt enable register_4 | TIER_4 | 8 | H′FE94 | TPU_4 | 16 | 2 |
| Timer status register_4 | TSR_4 | 8 | H′FE95 | TPU_4 | 16 | 2 |
| Timer counter H_4 | TCNTH_4 | 8 | H′FE96 | TPU_4 | 16 | 2 |
| Timer counter L_4 | TCNTL_4 | 8 | H′FE97 | TPU_4 | 16 | 2 |
| Timer general register AH_4 | TGRAH_4 | 8 | H′FE98 | TPU_4 | 16 | 2 |
| Timer general register AL_4 | TGRAL_4 | 8 | H′FE99 | TPU_4 | 16 | 2 |
| Timer general register BH_4 | TGRBH_4 | 8 | H′FE9A | TPU_4 | 16 | 2 |
| Timer general register BL_4 | TGRBL_4 | 8 | H′FE9B | TPU_4 | 16 | 2 |
| Timer control register_5 | TCR_5 | 8 | H′FEA0 | TPU_5 | 16 | 2 |
| Timer mode register_5 | TMDR_5 | 8 | H′FEA1 | TPU_5 | 16 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Timer I/O control register_5 | TIOR_5 | 8 | H′FEA2 | TPU_5 | 16 | 2 |
| Timer interrupt enable register_5 | TIER_5 | 8 | H′FEA4 | TPU_5 | 16 | 2 |
| Timer status register_5 | TSR_5 | 8 | H′FEA5 | TPU_5 | 16 | 2 |
| Timer counter H_5 | TCNTH_5 | 8 | H′FEA6 | TPU_5 | 16 | 2 |
| Timer counter L_5 | TCNTL_5 | 8 | H′FEA7 | TPU_5 | 16 | 2 |
| Timer general register AH_5 | TGRAH_5 | 8 | H′FEA8 | TPU_5 | 16 | 2 |
| Timer general register AL_5 | TGRAL_5 | 8 | H′FEA9 | TPU_5 | 16 | 2 |
| Timer general register BH_5 | TGRBH_5 | 8 | H′FEAA | TPU_5 | 16 | 2 |
| Timer general register BL_5 | TGRBL_5 | 8 | H′FEAB | TPU_5 | 16 | 2 |
| Timer start register | TSTR | 8 | H′FEB0 | TPU common | 16 | 2 |
| Timer synchro register | TSYR | 8 | H′FEB1 | TPU common | 16 | 2 |
| Interrupt priority register A | IPRA | 8 | H′FEC0 | INT | 8 | 2 |
| Interrupt priority register B | IPRB | 8 | H′FEC1 | INT | 8 | 2 |
| Interrupt priority register C | IPRC | 8 | H′FEC2 | INT | 8 | 2 |
| Interrupt priority register D | IPRD | 8 | H′FEC3 | INT | 8 | 2 |
| Interrupt priority register E | IPRE | 8 | H′FEC4 | INT | 8 | 2 |
| Interrupt priority register F | IPRF | 8 | H′FEC5 | INT | 8 | 2 |
| Interrupt priority register G | IPRG | 8 | H′FEC6 | INT | 8 | 2 |
| Interrupt priority register H | IPRH | 8 | H′FEC7 | INT | 8 | 2 |
| Interrupt priority register J | IPRJ | 8 | H′FEC9 | INT | 8 | 2 |
| Interrupt priority register K | IPRK | 8 | H′FECA | INT | 8 | 2 |
| Interrupt priority register M | IPRM | 8 | H′FECC | INT | 8 | 2 |
| RAM emulation register | RAMER | 8 | H′FEDB | ROM | 8 | 2 |
| Port 1 data register | P1DR | 8 | H′FF00 | PORT | 8 | 2 |
| Port 3 data register | P3DR | 8 | H′FF02 | PORT | 8 | 2 |
| Port 7 data register | P7DR | 8 | H′FF06 | PORT | 8 | 2 |
| Port A data register | PADR | 8 | H′FF09 | PORT | 8 | 2 |
| Port B data register | PBDR | 8 | H′FF0A | PORT | 8 | 2 |
| Port C data register | PCDR | 8 | H′FF0B | PORT | 8 | 2 |
| Port D data register | PDDR | 8 | H′FF0C | PORT | 8 | 2 |
| Port F data register | PFDR | 8 | H′FF0E | PORT | 8 | 2 |
| Timer control register_0 | TCR_0 | 8 | H′FF10 | TPU_0 | 16 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Timer mode register_0 | TMDR_0 | 8 | H′FF11 | TPU_0 | 16 | 2 |
| Timer I/O control register H_0 | TIORH_0 | 8 | H′FF12 | TPU_0 | 16 | 2 |
| Timer I/O control register L_0 | TIORL_0 | 8 | H′FF13 | TPU_0 | 16 | 2 |
| Timer interrupt enable register_0 | TIER_0 | 8 | H′FF14 | TPU_0 | 16 | 2 |
| Timer status register_0 | TSR_0 | 8 | H′FF15 | TPU_0 | 16 | 2 |
| Timer counter H_0 | TCNTH_0 | 8 | H′FF16 | TPU_0 | 16 | 2 |
| Timer counter L_0 | TCNTL_0 | 8 | H′FF17 | TPU_0 | 16 | 2 |
| Timer general register AH_0 | TGRAH_0 | 8 | H′FF18 | TPU_0 | 16 | 2 |
| Timer general register AL_0 | TGRAL_0 | 8 | H′FF19 | TPU_0 | 16 | 2 |
| Timer general register BH_0 | TGRBH_0 | 8 | H′FF1A | TPU_0 | 16 | 2 |
| Timer general register BL_0 | TGRBL_0 | 8 | H′FF1B | TPU_0 | 16 | 2 |
| Timer general register CH_0 | TGRCH_0 | 8 | H′FF1C | TPU_0 | 16 | 2 |
| Timer general register CL_0 | TGRCL_0 | 8 | H′FF1D | TPU_0 | 16 | 2 |
| Timer general register DH_0 | TGRDH_0 | 8 | H′FF1E | TPU_0 | 16 | 2 |
| Timer general register DL_0 | TGRDL_0 | 8 | H′FF1F | TPU_0 | 16 | 2 |
| Timer control register_1 | TCR_1 | 8 | H′FF20 | TPU_1 | 16 | 2 |
| Timer mode register_1 | TMDR_1 | 8 | H′FF21 | TPU_1 | 16 | 2 |
| Timer I/O control register_1 | TIOR_1 | 8 | H′FF22 | TPU_1 | 16 | 2 |
| Timer interrupt enable register_1 | TIER_1 | 8 | H′FF24 | TPU_1 | 16 | 2 |
| Timer status register_1 | TSR_1 | 8 | H′FF25 | TPU_1 | 16 | 2 |
| Timer counter H_1 | TCNTH_1 | 8 | H′FF26 | TPU_1 | 16 | 2 |
| Timer counter L_1 | TCNTL_1 | 8 | H′FF27 | TPU_1 | 16 | 2 |
| Timer general register AH_1 | TGRAH_1 | 8 | H′FF28 | TPU_1 | 16 | 2 |
| Timer general register AL_1 | TGRAL_1 | 8 | H′FF29 | TPU_1 | 16 | 2 |
| Timer general register BH_1 | TGRBH_1 | 8 | H′FF2A | TPU_1 | 16 | 2 |
| Timer general register BL_1 | TGRBL_1 | 8 | H′FF2B | TPU_1 | 16 | 2 |
| Timer control register_2 | TCR_2 | 8 | H′FF30 | TPU_2 | 16 | 2 |
| Timer mode register_2 | TMDR_2 | 8 | H′FF31 | TPU_2 | 16 | 2 |
| Timer I/O control register_2 | TIOR_2 | 8 | H′FF32 | TPU_2 | 16 | 2 |
| Timer interrupt enable register_2 | TIER_2 | 8 | H′FF34 | TPU_2 | 16 | 2 |
| Timer status register_2 | TSR_2 | 8 | H′FF35 | TPU_2 | 16 | 2 |
| Timer counterH_2 | TCNTH_2 | 8 | H′FF36 | TPU_2 | 16 | 2 |
| Timer counter L_2 | TCNTL_2 | 8 | H′FF37 | TPU_2 | 16 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| Timer general register AH_2 | TGRAH_2 | 8 | H′FF38 | TPU_2 | 16 | 2 |
| Timer general register AL_2 | TGRAL_2 | 8 | H′FF39 | TPU_2 | 16 | 2 |
| Timer general register BH_2 | TGRBH_2 | 8 | H′FF3A | TPU_2 | 16 | 2 |
| Timer general register BL_2 | TGRBL_2 | 8 | H′FF3B | TPU_2 | 16 | 2 |
| Timer control register_0 | TCR_0 | 8 | H′FF68 | TMR_0 | 8 | 2 |
| Timer control register_1 | TCR_1 | 8 | H′FF69 | TMR_1 | 8 | 2 |
| Timer control/status register_0 | TCSR_0 | 8 | H′FF6A | TMR_0 | 8 | 2 |
| Timer control/status register_1 | TCSR_1 | 8 | H′FF6B | TMR_1 | 8 | 2 |
| Time constant register A_0 | TCORA_0 | 8 | H′FF6C | TMR_0 | 8 | 2 |
| Time constant register A_1 | TCORA_1 | 8 | H′FF6D | TMR_1 | 8 | 2 |
| Time constant register B_0 | TCORB_0 | 8 | H′FF6E | TMR_0 | 8 | 2 |
| Time constant register B_1 | TCORB_1 | 8 | H′FF6F | TMR_1 | 8 | 2 |
| Timer counter_0 | TCNT_0 | 8 | H′FF70 | TMR_0 | 8 | 2 |
| Timer counter_1 | TCNT_1 | 8 | H′FF71 | TMR_1 | 8 | 2 |
| Timer control/status register_0 | TCSR_0 | 8 | H′FF74 | WDT_0 | 16 | 2 |
| Timer counter_0 | TCNT_0 | 8 | H′FF75 | WDT_0 | 16 | 2 |
| Reset control/status register | RSTCSR | 8 | H′FF77 | WDT | 16 | 2 |
| Serial mode register_0 | SMR_0 | 8 | H′FF78 | SCI_0 | 8 | 2 |
| Bit rate register_0 | BRR_0 | 8 | H′FF79 | SCI_0 | 8 | 2 |
| Serial control register_0 | SCR_0 | 8 | H′FF7A | SCI_0 | 8 | 2 |
| Transmit data register_0 | TDR_0 | 8 | H′FF7B | SCI_0 | 8 | 2 |
| Serial status register_0 | SSR_0 | 8 | H′FF7C | SCI_0 | 8 | 2 |
| Receive data register_0 | RDR_0 | 8 | H′FF7D | SCI_0 | 8 | 2 |
| Smart card mode register_0 | SCMR_0 | 8 | H′FF7E | SCI_0 | 8 | 2 |
| Serial mode register_2 | SMR_2 | 8 | H′FF88 | SCI_2 | 8 | 2 |
| Bit rate register_2 | BRR_2 | 8 | H′FF89 | SCI_2 | 8 | 2 |
| Serial control register_2 | SCR_2 | 8 | H′FF8A | SCI_2 | 8 | 2 |
| Transmit data register_2 | TDR_2 | 8 | H′FF8B | SCI_2 | 8 | 2 |
| Serial status register_2 | SSR_2 | 8 | H′FF8C | SCI_2 | 8 | 2 |
| Receive data register_2 | RDR_2 | 8 | H′FF8D | SCI_2 | 8 | 2 |
| Smart card mode register_2 | SCMR_2 | 8 | H′FF8E | SCI_2 | 8 | 2 |
| A/D data register AH | ADDRAH | 8 | H′FF90 | A/D | 8 | 2 |
| A/D data register AL | ADDRAL | 8 | H′FF91 | A/D | 8 | 2 |

| Register Name | Abbreviation | Number of Bits | Address* | Module | Data Width | Access State |
|---|---|---|---|---|---|---|
| A/D data register BH | ADDRBH | 8 | H′FF92 | A/D | 8 | 2 |
| A/D data register BL | ADDRBL | 8 | H′FF93 | A/D | 8 | 2 |
| A/D data register CH | ADDRCH | 8 | H′FF94 | A/D | 8 | 2 |
| A/D data register CL | ADDRCL | 8 | H′FF95 | A/D | 8 | 2 |
| A/D data register DH | ADDRDH | 8 | H′FF96 | A/D | 8 | 2 |
| A/D data register DL | ADDRDL | 8 | H′FF97 | A/D | 8 | 2 |
| A/D control/status register | ADCSR | 8 | H′FF98 | A/D | 8 | 2 |
| A/D control register | ADCR | 8 | H′FF99 | A/D | 8 | 2 |
| Flash memory control register 1 | FLMCR1 | 8 | H′FFA8 | ROM | 8 | 2 |
| Flash memory control register 2 | FLMCR2 | 8 | H′FFA9 | ROM | 8 | 2 |
| Erase block register 1 | EBR1 | 8 | H′FFAA | ROM | 8 | 2 |
| Erase block register 2 | EBR2 | 8 | H′FFAB | ROM | 8 | 2 |
| Port 1 register | PORT1 | 8 | H′FFB0 | PORT | 8 | 2 |
| Port 3 register | PORT3 | 8 | H′FFB2 | PORT | 8 | 2 |
| Port 4 register | PORT4 | 8 | H′FFB3 | PORT | 8 | 2 |
| Port 7 register | PORT7 | 8 | H′FFB6 | PORT | 8 | 2 |
| Port 9 register | PORT9 | 8 | H′FFB8 | PORT | 8 | 2 |
| Port A register | PORTA | 8 | H′FFB9 | PORT | 8 | 2 |
| Port B register | PORTB | 8 | H′FFBA | PORT | 8 | 2 |
| Port C register | PORTC | 8 | H′FFBB | PORT | 8 | 2 |
| Port D register | PORTD | 8 | H′FFBC | PORT | 8 | 2 |
| Port F register | PORTF | 8 | H′FFBE | PORT | 8 | 2 |

Note: Lower 16 bits of the address.

## 22.2 Register Bits

The addresses and bit names of the registers in the on-chip peripheral modules are listed below. The 16-bit register is indicated in two rows, 8 bits for each row.

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MCR | MCR7 | — | MCR5 | — | — | MCR2 | MCR1 | MCR0 | HCAN |
| GSR | — | — | — | — | GSR3 | GSR2 | GSR1 | GSR0 | |
| BCR | BCR7 | BCR6 | BCR5 | BCR4 | BCR3 | BCR2 | BCR1 | BCR0 | |
| | BCR15 | BCR14 | BCR13 | BCR12 | BCR11 | BCR10 | BCR9 | BCR8 | |
| MBCR | MBCR7 | MBCR6 | MBCR5 | MBCR4 | MBCR3 | MBCR2 | MBCR1 | — | |
| | MBCR15 | MBCR14 | MBCR13 | MBCR12 | MBCR11 | MBCR10 | MBCR9 | MBCR8 | |
| TXPR | TXPR7 | TXPR6 | TXPR5 | TXPR4 | TXPR3 | TXPR2 | TXPR1 | — | |
| | TXPR15 | TXPR14 | TXPR13 | TXPR12 | TXPR11 | TXPR10 | TXPR9 | TXPR8 | |
| TXCR | TXCR7 | TXCR6 | TXCR5 | TXCR4 | TXCR3 | TXCR2 | TXCR1 | — | |
| | TXCR15 | TXCR14 | TXCR13 | TXCR12 | TXCR11 | TXCR10 | TXCR9 | TXCR8 | |
| TXACK | TXACK7 | TXACK6 | TXACK5 | TXACK4 | TXACK3 | TXACK2 | TXACK1 | — | |
| | TXACK15 | TXACK14 | TXACK13 | TXACK12 | TXACK11 | TXACK10 | TXACK9 | TXACK8 | |
| ABACK | ABACK7 | ABACK6 | ABACK5 | ABACK4 | ABACK3 | ABACK2 | ABACK1 | — | |
| | ABACK15 | ABACK14 | ABACK13 | ABACK12 | ABACK11 | ABACK10 | ABACK9 | ABACK8 | |
| RXPR | RXPR7 | RXPR6 | RXPR5 | RXPR4 | RXPR3 | RXPR2 | RXPR1 | RXPR0 | |
| | RXPR15 | RXPR14 | RXPR13 | RXPR12 | RXPR11 | RXPR10 | RXPR9 | RXPR8 | |
| RFPR | RFPR7 | RFPR6 | RFPR5 | RFPR4 | RFPR3 | RFPR2 | RFPR1 | RFPR0 | |
| | RFPR15 | RFPR14 | RFPR13 | RFPR12 | RFPR11 | RFPR10 | RFPR9 | RFPR8 | |
| IRR | IRR7 | IRR6 | IRR5 | IRR4 | IRR3 | IRR2 | IRR1 | IRR0 | |
| | — | — | — | IRR12 | — | — | IRR9 | IRR8 | |
| MBIMR | MBIMR7 | MBIMR6 | MBIMR5 | MBIMR4 | MBIMR3 | MBIMR2 | MBIMR1 | MBIMR0 | |
| | MBIMR15 | MBIMR14 | MBIMR13 | MBIMR12 | MBIMR11 | MBIMR10 | MBIMR9 | MBIMR8 | |
| IMR | IMR7 | IMR6 | IMR5 | IMR4 | IMR3 | IMR2 | IMR1 | — | |
| | — | — | — | IMR12 | — | — | IMR9 | IMR8 | |
| REC | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TEC | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| UMSR | UMSR7 | UMSR6 | UMSR5 | UMSR4 | UMSR3 | UMSR2 | UMSR1 | UMSR0 | |
| | UMSR15 | UMSR14 | UMSR13 | UMSR12 | UMSR11 | UMSR10 | UMSR9 | UMSR8 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| LAFML | LAFML7 | LAFML6 | LAFML5 | LAFML4 | LAFML3 | LAFML2 | LAFML1 | LAFML0 | HCAN |
| | LAFML15 | LAFML14 | LAFML13 | LAFML12 | LAFML11 | LAFML10 | LAFML9 | LAFML8 | |
| LAFMH | LAFMH7 | LAFMH6 | LAFMH5 | — | — | — | LAFMH1 | LAFMH0 | |
| | LAFMH15 | LAFMH14 | LAFMH13 | LAFMH12 | LAFMH11 | LAFMH10 | LAFMH9 | LAFMH8 | |
| MC0[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC0[2] | — | — | — | — | — | — | — | — | |
| MC0[3] | — | — | — | — | — | — | — | — | |
| MC0[4] | — | — | — | — | — | — | — | — | |
| MC0[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC0[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC0[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC0[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC1[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC1[2] | — | — | — | — | — | — | — | — | |
| MC1[3] | — | — | — | — | — | — | — | — | |
| MC1[4] | — | — | — | — | — | — | — | — | |
| MC1[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | – | ID–17 | ID–16 | |
| MC1[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC1[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC1[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC2[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC2[2] | — | — | — | — | — | — | — | — | |
| MC2[3] | — | — | — | — | — | — | — | — | |
| MC2[4] | — | — | — | — | — | — | — | — | |
| MC2[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC2[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC2[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC2[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC3[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC3[2] | — | — | — | — | — | — | — | — | |
| MC3[3] | — | — | — | — | — | — | — | — | |
| MC3[4] | — | — | — | — | — | — | — | — | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MC3[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | HCAN |
| MC3[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC3[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC3[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC4[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC4[2] | — | — | — | — | — | — | — | — | |
| MC4[3] | — | — | — | — | — | — | — | — | |
| MC4[4] | — | — | — | — | — | — | — | — | |
| MC4[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC4[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC4[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC4[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC5[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC5[2] | — | — | — | — | — | — | — | — | |
| MC5[3] | — | — | — | — | — | — | — | — | |
| MC5[4] | — | — | — | — | — | — | — | — | |
| MC5[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC5[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC5[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC5[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC6[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC6[2] | — | — | — | — | — | — | — | — | |
| MC6[3] | — | — | — | — | — | — | — | — | |
| MC6[4] | — | — | — | — | — | — | — | — | |
| MC6[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC6[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC6[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC6[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC7[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC7[2] | — | — | — | — | — | — | — | — | |
| MC7[3] | — | — | — | — | — | — | — | — | |
| MC7[4] | — | — | — | — | — | — | — | — | |
| MC7[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MC7[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | HCAN |
| MC7[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC7[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC8[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC8[2] | — | — | — | — | — | — | — | — | |
| MC8[3] | — | — | — | — | — | — | — | — | |
| MC8[4] | — | — | — | — | — | — | — | — | |
| MC8[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC8[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC8[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC8[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC9[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC9[2] | — | — | — | — | — | — | — | — | |
| MC9[3] | — | — | — | — | — | — | — | — | |
| MC9[4] | — | — | — | — | — | — | — | — | |
| MC9[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC9[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC9[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC9[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC10[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC10[2] | — | — | — | — | — | — | — | — | |
| MC10[3] | — | — | — | — | — | — | — | — | |
| MC10[4] | — | — | — | — | — | — | — | — | |
| MC10[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC10[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC10[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC10[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC11[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC11[2] | — | — | — | — | — | — | — | — | |
| MC11[3] | — | — | — | — | — | — | — | — | |
| MC11[4] | — | — | — | — | — | — | — | — | |
| MC11[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | – | ID–17 | ID–16 | |
| MC11[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MC11[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | HCAN |
| MC11[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC12[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC12[2] | — | — | — | — | — | — | — | — | |
| MC12[3] | — | — | — | — | — | — | — | — | |
| MC12[4] | — | — | — | — | — | — | — | — | |
| MC12[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC12[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC12[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC12[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC13[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC13[2] | — | — | — | — | — | — | — | — | |
| MC13[3] | — | — | — | — | — | — | — | — | |
| MC13[4] | — | — | — | — | — | — | — | — | |
| MC13[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC13[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC13[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC13[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC14[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC14[2] | — | — | — | — | — | — | — | — | |
| MC14[3] | — | — | — | — | — | — | — | — | |
| MC14[4] | — | — | — | — | — | — | — | — | |
| MC14[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC14[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |
| MC14[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | |
| MC14[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MC15[1] | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 | |
| MC15[2] | — | — | — | — | — | — | — | — | |
| MC15[3] | — | — | — | — | — | — | — | — | |
| MC15[4] | — | — | — | — | — | — | — | — | |
| MC15[5] | ID–20 | ID–19 | ID–18 | RTR | IDE | — | ID–17 | ID–16 | |
| MC15[6] | ID–28 | ID–27 | ID–26 | ID–25 | ID–24 | ID–23 | ID–22 | ID–21 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MC15[7] | ID–7 | ID–6 | ID–5 | ID–4 | ID–3 | ID–2 | ID–1 | ID–0 | HCAN |
| MC15[8] | ID–15 | ID–14 | ID–13 | ID–12 | ID–11 | ID–10 | ID–9 | ID–8 | |
| MD0[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD0[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD1[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD2[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD3[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MD3[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | HCAN |
| MD4[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD4[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD5[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD6[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD7[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MD8[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | HCAN |
| MD8[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD8[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD9[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD10[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD11[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| MD12[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | HCAN |
| MD12[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD12[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD13[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD14[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[2] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[3] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[4] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[5] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[6] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[7] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| MD15[8] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| HCAN MON | — | — | — | — | — | — | TxD | RxD | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| SSCRH_0 | MSS | BIDE | — | SOL | SOLP | SCKS | CSSI | CSSO | SSU_0 |
| SSCRL_0 | — | — | SRES | — | — | — | DATSI | DATSO | |
| SSMR_0 | MLS | CPOS | CPHS | — | — | CKS2 | CKS1 | CKS0 | |
| SSER_0 | TE | RE | — | — | TEIE | TIE | RIE | CEIE | |
| SSSR_0 | — | ORER | — | — | TEND | TDRE | RDRF | CE | |
| SSTDR0_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSTDR1_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSTDR2_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSTDR3_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR0_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR1_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR2_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR3_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSCRH_1 | MSS | BIDE | — | SOL | SOLP | SCKS | CSSI | CSSO | SSU_1 |
| SSCRL_1 | — | — | SRES | — | — | — | DATSI | DATSO | |
| SSMR_1 | MLS | CPOS | CPHS | — | — | CKS2 | CKS1 | CKS0 | |
| SSER_1 | TE | RE | — | — | TEIE | TIE | RIE | CEIE | |
| SSSR_1 | — | ORER | — | — | TEND | TDRE | RDRF | CE | |
| SSTDR0_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSTDR1_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| SSTDR2_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | SSU_1 |
| SSTDR3_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR0_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR1_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR2_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSRDR3_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| PDRTIDR | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | PORT |
| TCR_2 | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR_2 |
| TCR_3 | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR_3 |
| TCSR_2 | CMFB | CMFA | OVF | ADTE | OS3 | OS2 | OS1 | OS0 | |
| TCSR_3 | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 | |
| TCORA_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORA_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORB_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORB_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCNT_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCNT_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SBYCR | SSBY | STS2 | STS1 | STS0 | — | — | — | — | SYSTEM |
| SYSCR | MACS | — | INTM1 | INTM0 | NMIEG | — | — | RAME | |
| SCKCR | PSTOP | — | — | — | STCS | SCK2 | SCK1 | SCK0 | |
| MDCR | — | — | — | — | — | MDS2 | MDS1 | MDS0 | |
| MSTPCRA | MSTPA7 | MSTPA6 | MSTPA5 | MSTPA4 | MSTPA3 | MSTPA2 | MSTPA1 | MSTPA0 | |
| MSTPCRB | MSTPB7 | MSTPB6 | MSTPB5 | MSTPB4 | MSTPB3 | MSTPB2 | MSTPB1 | MSTPB0 | |
| MSTPCRC | MSTPC7 | MSTPC6 | MSTPC5 | MSTPC4 | MSTPC3 | MSTPC2 | MSTPC1 | MSTPC0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| LPWRCR | — | — | — | — | — | — | STC1 | STC0 | SYSTEM |
| BARA | — | — | — | — | — | — | — | — | PBC |
| | BAA23 | BAA22 | BAA21 | BAA20 | BAA19 | BAA18 | BAA17 | BAA16 | |
| | BAA15 | BAA14 | BAA13 | BAA12 | BAA11 | BAA10 | BAA9 | BAA8 | |
| | BAA7 | BAA6 | BAA5 | BAA4 | BAA3 | BAA2 | BAA1 | BAA0 | |
| BARB | — | — | — | — | — | — | — | — | |
| | BAB23 | BAB22 | BAB21 | BAB20 | BAB19 | BAB18 | BAB17 | BAB16 | |
| | BAB15 | BAB14 | BAB13 | BAB12 | BAB11 | BAB10 | BAB9 | BAB8 | |
| | BAB7 | BAB6 | BAB5 | BAB4 | BAB3 | BAB2 | BAB1 | BAB0 | |
| BCRA | CMFA | CDA | BAMRA2 | BAMRA1 | BAMRA0 | CSELA1 | CSELA0 | BIEA | |
| BCRB | CMFB | CDB | BAMRB2 | BAMRB1 | BAMRB0 | CSELB1 | CSELB0 | BIEB | |
| ISCRH | — | — | — | — | IRQ5SCB | IRQ5SCA | IRQ4SCB | IRQ4SCA | INT |
| ISCRL | IRQ3SCB | IRQ3SCA | IRQ2SCB | IRQ2SCA | IRQ1SCB | IRQ1SCA | IRQ0SCB | IRQ0SCA | |
| IER | — | — | IRQ5E | IRQ4E | IRQ3E | IRQ2E | IRQ1E | IRQ0E | |
| ISR | — | — | IRQ5F | IRQ4F | IRQ3F | IRQ2F | IRQ1F | IRQ0F | |
| DTCERA | DTCEA7 | DTCEA6 | DTCEA5 | DTCEA4 | DTCEA3 | DTCEA2 | DTCEA1 | DTCEA0 | DTC |
| DTCERB | DTCEB7 | DTCEB6 | DTCEB5 | DTCEB4 | DTCEB3 | DTCEB2 | DTCEB1 | DTCEB0 | |
| DTCERC | DTCEC7 | DTCEC6 | DTCEC5 | DTCEC4 | DTCEC3 | DTCEC2 | DTCEC1 | DTCEC0 | |
| DTCERD | DTCED7 | DTCED6 | DTCED5 | DTCED4 | DTCED3 | DTCED2 | DTCED1 | DTCED0 | |
| DTCERE | DTCEE7 | DTCEE6 | DTCEE5 | DTCEE4 | DTCEE3 | DTCEE2 | DTCEE1 | DTCEE0 | |
| DTCERF | DTCEF7 | DTCEF6 | DTCEF5 | DTCEF4 | DTCEF3 | DTCEF2 | DTCEF1 | DTCEF0 | |
| DTCERG | DTCEG7 | DTCEG6 | DTCEG5 | DTCEG4 | DTCEG3 | DTCEG2 | DTCEG1 | DTCEG0 | |
| DTVECR | SWDTE | DTVEC6 | DTVEC5 | DTVEC4 | DTVEC3 | DTVEC2 | DTVEC1 | DTVEC0 | |
| PCR | G3CMS1 | G3CMS0 | G2CMS1 | G2CMS0 | G1CMS1 | G1CMS0 | G0CMS1 | G0CMS0 | PPG |
| PMR | G3INV | G2INV | — | — | G3NOV | G2NOV | — | — | |
| NDERH | NDER15 | NDER14 | NDER13 | NDER12 | NDER11 | NDER10 | NDER9 | NDER8 | |
| NDERL | NDER7 | NDER6 | NDER5 | NDER4 | NDER3 | NDER2 | NDER1 | NDER0 | |
| PODRH | POD15 | POD14 | POD13 | POD12 | POD11 | POD10 | POD9 | POD8 | |
| PODRL | POD7 | POD6 | POD5 | POD4 | POD3 | POD2 | POD1 | POD0 | |
| NDRH | NDR15 | NDR14 | NDR13 | NDR12 | NDR11 | NDR10 | NDR9 | NDR8 | |
| NDRL | NDR7 | NDR6 | NDR5 | NDR4 | NDR3 | NDR2 | NDR1 | NDR0 | |
| NDRH | — | — | — | — | NDR11 | NDR10 | NDR9 | NDR8 | |
| NDRL | — | — | — | — | NDR3 | NDR2 | NDR1 | NDR0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| P1DDR | P17DDR | P16DDR | P15DDR | P14DDR | P13DDR | P12DDR | P11DDR | P10DDR | PORT |
| P3DDR | P37DDR | P36DDR | P35DDR | P34DDR | P33DDR | P32DDR | P31DDR | P30DDR | |
| P7DDR | P77DDR | P76DDR | P75DDR | P74DDR | P73DDR | P72DDR | P71DDR | P70DDR | |
| PADDR | — | — | — | — | PA3DDR | PA2DDR | PA1DDR | PA0DDR | |
| PBDDR | PB7DDR | PB6DDR | PB5DDR | PB4DDR | PB3DDR | PB2DDR | PB1DDR | PB0DDR | |
| PCDDR | PC7DDR | PC6DDR | PC5DDR | PC4DDR | PC3DDR | PC2DDR | PC1DDR | PC0DDR | |
| PDDDR | PD7DDR | PD6DDR | PD5DDR | PD4DDR | PD3DDR | PD2DDR | PD1DDR | PD0DDR | |
| PFDDR | PF7DDR | PF6DDR | PF5DDR | PF4DDR | PF3DDR | PF2DDR | PF1DDR | PF0DDR | |
| PAPCR | — | — | — | — | PA3PCR | PA2PCR | PA1PCR | PA0PCR | |
| PBPCR | PB7PCR | PB6PCR | PB5PCR | PB4PCR | PB3PCR | PB2PCR | PB1PCR | PB0PCR | |
| PCPCR | PC7PCR | PC6PCR | PC5PCR | PC4PCR | PC3PCR | PC2PCR | PC1PCR | PC0PCR | |
| PDPCR | PD7PCR | PD6PCR | PD5PCR | PD4PCR | PD3PCR | PD2PCR | PD1PCR | PD0PCR | |
| P3ODR | P37ODR | P36ODR | P35ODR | P34ODR | P33ODR | P32ODR | P31ODR | P30ODR | |
| PAODR | — | — | — | — | PA3ODR | PA2ODR | PA1ODR | PA0ODR | |
| PBODR | PB7ODR | PB6ODR | PB5ODR | PB4ODR | PB3ODR | PB2ODR | PB1ODR | PB0ODR | |
| PCODR | PC7ODR | PC6ODR | PC5ODR | PC4ODR | PC3ODR | PC2ODR | PC1ODR | PC0ODR | |
| TCR_3 | CCLR2 | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_3 |
| TMDR_3 | — | — | BFB | BFA | MD3 | MD2 | MD1 | MD0 | |
| TIORH_3 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIORL_3 | IOD3 | IOD2 | IOD1 | IOD0 | IOC3 | IOC2 | IOC1 | IOC0 | |
| TIER_3 | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA | |
| TSR_3 | — | — | — | TCFV | TGFD | TGFC | TGFB | TGFA | |
| TCNTH_3 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TCNTL_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRAH_3 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_3 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRCH_3 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRCL_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRDH_3 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRDL_3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| TCR_4 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_4 |
| TMDR_4 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| TIOR_4 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIER_4 | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| TSR_4 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| TCNTH_4 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TCNTL_4 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRAH_4 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_4 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_4 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_4 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCR_5 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_5 |
| TMDR_5 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| TIOR_5 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIER_5 | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| TSR_5 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| TCNTH_5 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TCNTL_5 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRAH_5 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_5 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_5 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_5 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TSTR | — | — | CST5 | CST4 | CST3 | CST2 | CST1 | CST0 | TPU |
| TSYR | — | — | SYNC5 | SYNC4 | SYNC3 | SYNC2 | SYNC1 | SYNC0 | common |
| IPRA | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | INT |
| IPRB | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRC | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRD | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRE | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRF | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRG | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRH | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| IPRJ | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| IPRK | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | INT |
| IPRM | — | IPR6 | IPR5 | IPR4 | — | IPR2 | IPR1 | IPR0 | |
| RAMER | — | — | — | — | RAMS | RAM2 | RAM1 | RAM0 | ROM |
| P1DR | P17DR | P16DR | P15DR | P14DR | P13DR | P12DR | P11DR | P10DR | PORT |
| P3DR | P37DR | P36DR | P35DR | P34DR | P33DR | P32DR | P31DR | P30DR | |
| P7DR | P77DR | P76DR | P75DR | P74DR | P73DR | P72DR | P71DR | P70DR | |
| PADR | — | — | — | — | PA3DR | PA2DR | PA1DR | PA0DR | |
| PBDR | PB7DR | PB6DR | PB5DR | PB4DR | PB3DR | PB2DR | PB1DR | PB0DR | |
| PCDR | PC7DR | PC6DR | PC5DR | PC4DR | PC3DR | PC2DR | PC1DR | PC0DR | |
| PDDR | PD7DR | PD6DR | PD5DR | PD4DR | PD3DR | PD2DR | PD1DR | PD0DR | |
| PFDR | PF7DR | PF6DR | PF5DR | PF4DR | PF3DR | PF2DR | PF1DR | PF0DR | |
| TCR_0 | CCLR2 | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_0 |
| TMDR_0 | — | — | BFB | BFA | MD3 | MD2 | MD1 | MD0 | |
| TIORH_0 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIORL_0 | IOD3 | IOD2 | IOD1 | IOD0 | IOC3 | IOC2 | IOC1 | IOC0 | |
| TIER_0 | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA | |
| TSR_0 | — | — | — | TCFV | TGFD | TGFC | TGFB | TGFA | |
| TCNTH_0 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TCNTL_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRAH_0 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_0 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRCH_0 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRCL_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRDH_0 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | TPU_0 |
| TGRDL_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCR_1 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_1 |
| TMDR_1 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| TIOR_1 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIER_1 | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| TSR_1 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| TCNTH_1 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| TCNTL_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | TPU_1 |
| TGRAH_1 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_1 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCR_2 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU_2 |
| TMDR_2 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| TIOR_2 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| TIER_2 | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| TSR_2 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| TCNTH_2 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TCNTL_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRAH_2 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRAL_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TGRBH_2 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | |
| TGRBL_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCR_0 | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR_0 |
| TCR_1 | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR_1 |
| TCSR_0 | CMFB | CMFA | OVF | ADTE | OS3 | OS2 | OS1 | OS0 | |
| TCSR_1 | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 | |
| TCORA_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORA_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORB_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCORB_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCNT_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCNT_1 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| TCSR_0 | OVF | WT/IT | TME | — | — | CKS2 | CKS1 | CKS0 | WDT_0 |
| TCNT_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| RSTCSR | WOVF | RSTE | RSTS | — | — | — | — | — | |
| SMR_0*1 | C/A | CHR | PE | O/E | STOP | MP | CKS1 | CKS0 | SCI_0 |
| (SMR_0*2) | (GM) | (BLK) | (PE) | (O/E) | (BCP1) | (BCP0) | (CKS1) | (CKS0) | |
| BRR_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SCR_0 | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| TDR_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | SCI_0 |
| SSR_0*1 | TDRE | RDRF | ORER | FER | PER | TEND | MPB | MPBT | |
| (SSR_0*2) | (TDRE) | (RDRF) | (ORER) | (ERS) | (PER) | (TEND) | (MPB) | (MPBT) | |
| RDR_0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SCMR_0 | — | — | — | — | SDIR | SINV | — | SMIF | |
| SMR_2*1 | C/A | CHR | PE | O/E | STOP | MP | CKS1 | CKS0 | SCI_2 |
| (SMR_2*2) | (GM) | (BLK) | (PE) | (O/E) | (BCP1) | (BCP0) | (CKS1) | (CKS0) | |
| BRR_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SCR_2 | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 | |
| TDR_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SSR_2*1 | TDRE | RDRF | ORER | FER | PER | TEND | MPB | MPBT | |
| (SSR_2*2) | (TDRE) | (RDRF) | (ORER) | (ERS) | (PER) | (TEND) | (MPB) | (MPBT) | |
| RDR_2 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| SCMR_2 | — | — | — | — | SDIR | SINV | — | SMIF | |
| ADDRAH | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | A/D |
| ADDRAL | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRBH | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| ADDRBL | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRCH | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| ADDRCL | AD1 | AD0 | — | — | — | — | — | — | |
| ADDRDH | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | |
| ADDRDL | AD1 | AD0 | — | — | — | — | — | — | |
| ADCSR | ADF | ADIE | ADST | SCAN | CH3 | CH2 | CH1 | CH0 | |
| ADCR | TRGS1 | TRGS0 | — | — | CKS1 | CKS0 | — | — | |
| FLMCR1 | FWE | SWE | ESU1 | PSU1 | EV1 | PV1 | E1 | P1 | ROM |
| FLMCR2 | FLER | — | — | — | — | — | — | — | |
| EBR1 | EB7 | EB6 | EB5 | EB4 | EB3 | EB2 | EB1 | EB0 | |
| EBR2 | — | — | — | — | — | — | EB9 | EB8 | |
| PORT1 | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 | PORT |
| PORT3 | P37 | P36 | P35 | P34 | P33 | P32 | P31 | P30 | |
| PORT4 | P47 | P46 | P45 | P44 | P43 | P42 | P41 | P40 | |
| PORT7 | P77 | P76 | P75 | P74 | P73 | P72 | P71 | P70 | |
| PORT9 | P97 | P96 | P95 | P94 | P93 | P92 | P91 | P90 | |

| Abbreviation | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Module |
|---|---|---|---|---|---|---|---|---|---|
| PORTA | — | — | — | — | PA3 | PA2 | PA1 | PA0 | PORT |
| PORTB | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 | |
| PORTC | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 | |
| PORTD | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 | |
| PORTF | PF7 | PF6 | PF5 | PF4 | PF3 | PF2 | PF1 | PF0 | |

Notes: 1. For buffer operation.
2. For free operation.
3. Normal serial communication interface mode.
4. Smart Card interface mode.

Some bit functions of SMR differ in normal serial communication interface mode and Smart Card interface mode.

## 22.3 Register States in Each Operating Mode

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MCR | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| GSR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| BCR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MBCR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TXPR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TXCR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TXACK | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ABACK | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| RXPR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| RFPR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| IRR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MBIMR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| IMR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| REC | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TEC | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| UMSR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| LAFML | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| LAFMH | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC0[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MC1[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MC1[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC1[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC2[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC3[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC4[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC5[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MC5[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MC5[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC6[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC7[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC8[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC9[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MC9[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MC10[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC10[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC11[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC12[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC13[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MC14[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MC14[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC14[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MC15[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD0[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD1[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MD2[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MD2[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD2[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD3[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD4[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD5[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MD6[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MD6[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD6[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD7[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD8[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD9[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MD10[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | HCAN |
| MD10[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD10[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD11[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD12[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD13[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| MD14[5] | Initialized | | | | Initialized | Initialized | Initialized | HCAN |
| MD14[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD14[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[1] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[2] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[3] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[4] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[5] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[6] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[7] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| MD15[8] | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| HCANMON | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSCRH_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | SSU_0 |
| SSCRL_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSMR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSER_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSSR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR0_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR1_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR2_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR3_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR0_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR1_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR2_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR3_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSCRH_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | SSU_1 |
| SSCRL_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSMR_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSER_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSSR_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR0_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSTDR1_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| SSTDR2_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | SSU_1 |
| SSTDR3_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR0_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR1_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR2_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSRDR3_1 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| PDRTIDR | Initialized | — | — | — | — | — | Initialized | PORT |
| TCR_2 | Initialized | — | — | — | — | — | Initialized | TMR_2 |
| TCR_3 | Initialized | — | — | — | — | — | Initialized | TMR_3 |
| TCSR_2 | Initialized | — | — | — | — | — | Initialized | |
| TCSR_3 | Initialized | — | — | — | — | — | Initialized | |
| TCORA_2 | Initialized | — | — | — | — | — | Initialized | |
| TCORA_3 | Initialized | — | — | — | — | — | Initialized | |
| TCORB_2 | Initialized | — | — | — | — | — | Initialized | |
| TCORB_3 | Initialized | — | — | — | — | — | Initialized | |
| TCNT_2 | Initialized | — | — | — | — | — | Initialized | |
| TCNT_3 | Initialized | — | — | — | — | — | Initialized | |
| SBYCR | Initialized | — | — | — | — | — | — | SYSTEM |
| SYSCR | Initialized | — | — | — | — | — | — | |
| SCKCR | Initialized | — | — | — | — | — | — | |
| MDCR | Initialized | — | — | — | — | — | — | |
| MSTPCRA | Initialized | — | — | — | — | — | — | |
| MSTPCRB | Initialized | — | — | — | — | — | — | |
| MSTPCRC | Initialized | — | — | — | — | — | — | |
| LPWRCR | Initialized | — | — | — | — | — | — | |
| BARA | Initialized | — | — | — | — | — | Initialized | PBC |
| BARB | Initialized | — | — | — | — | — | Initialized | |
| BCRA | Initialized | — | — | — | — | — | Initialized | |
| BCRB | Initialized | — | — | — | — | — | Initialized | |
| ISCRH | Initialized | — | — | — | — | — | Initialized | INT |
| ISCRL | Initialized | — | — | — | — | — | Initialized | |
| IER | Initialized | — | — | — | — | — | Initialized | |
| ISR | Initialized | — | — | — | — | — | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| DTCERA | Initialized | — | — | — | — | — | Initialized | DTC |
| DTCERB | Initialized | — | — | — | — | — | Initialized | |
| DTCERC | Initialized | — | — | — | — | — | Initialized | |
| DTCERD | Initialized | — | — | — | — | — | Initialized | |
| DTCERE | Initialized | — | — | — | — | — | Initialized | |
| DTCERF | Initialized | — | — | — | — | — | Initialized | |
| DTCERG | Initialized | — | — | — | — | — | Initialized | |
| DTVECR | Initialized | — | — | — | — | — | Initialized | |
| PCR | Initialized | — | — | — | — | — | Initialized | PPG |
| PMR | Initialized | — | — | — | — | — | Initialized | |
| NDERH | Initialized | — | — | — | — | — | Initialized | |
| NDERL | Initialized | — | — | — | — | — | Initialized | |
| PODRH | Initialized | — | — | — | — | — | Initialized | |
| PODRL | Initialized | — | — | — | — | — | Initialized | |
| NDRH | Initialized | — | — | — | — | — | Initialized | |
| NDRL | Initialized | — | — | — | — | — | Initialized | |
| NDRH | Initialized | — | — | — | — | — | Initialized | |
| NDRL | Initialized | — | — | — | — | — | Initialized | |
| P1DDR | Initialized | — | — | — | — | — | — | PORT |
| P3DDR | Initialized | — | — | — | — | — | — | |
| P7DDR | Initialized | — | — | — | — | — | — | |
| PADDR | Initialized | — | — | — | — | — | — | |
| PBDDR | Initialized | — | — | — | — | — | — | |
| PCDDR | Initialized | — | — | — | — | — | — | |
| PDDDR | Initialized | — | — | — | — | — | — | |
| PFDDR | Initialized | — | — | — | — | — | — | |
| PAPCR | Initialized | — | — | — | — | — | — | |
| PBPCR | Initialized | — | — | — | — | — | — | |
| PCPCR | Initialized | — | — | — | — | — | — | |
| PDPCR | Initialized | — | — | — | — | — | — | |
| P3ODR | Initialized | — | — | — | — | — | — | |
| PAODR | Initialized | — | — | — | — | — | — | |
| PBODR | Initialized | — | — | — | — | — | — | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| PCODR | Initialized | — | — | — | — | — | — | PORT |
| TCR_3 | Initialized | — | — | — | — | — | Initialized | TPU_3 |
| TMDR_3 | Initialized | — | — | — | — | — | Initialized | |
| TIORH_3 | Initialized | — | — | — | — | — | Initialized | |
| TIORL_3 | Initialized | — | — | — | — | — | Initialized | |
| TIER_3 | Initialized | — | — | — | — | — | Initialized | |
| TSR_3 | Initialized | — | — | — | — | — | Initialized | |
| TCNTH_3 | Initialized | — | — | — | — | — | Initialized | |
| TCNTL_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRCH_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRCL_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRDH_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRDL_3 | Initialized | — | — | — | — | — | Initialized | |
| TCR_4 | Initialized | — | — | — | — | — | Initialized | TPU_4 |
| TMDR_4 | Initialized | — | — | — | — | — | Initialized | |
| TIOR_4 | Initialized | — | — | — | — | — | Initialized | |
| TIER_4 | Initialized | — | — | — | — | — | Initialized | |
| TSR_4 | Initialized | — | — | — | — | — | Initialized | |
| TCNTH_4 | Initialized | — | — | — | — | — | Initialized | |
| TCNTL_4 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_4 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_4 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_3 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_4 | Initialized | — | — | — | — | — | Initialized | |
| TCR_5 | Initialized | — | — | — | — | — | Initialized | TPU_5 |
| TMDR_5 | Initialized | — | — | — | — | — | Initialized | |
| TIOR_5 | Initialized | — | — | — | — | — | Initialized | |
| TIER_5 | Initialized | — | — | — | — | — | Initialized | |
| TSR_5 | Initialized | — | — | — | — | — | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| TCNTH_5 | Initialized | — | — | — | — | — | Initialized | TPU_5 |
| TCNTL_5 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_5 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_5 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_5 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_5 | Initialized | — | — | — | — | — | Initialized | |
| TSTR | Initialized | — | — | — | — | — | Initialized | TPU |
| TSYR | Initialized | — | — | — | — | — | Initialized | common |
| IPRA | Initialized | — | — | — | — | — | Initialized | INT |
| IPRB | Initialized | — | — | — | — | — | Initialized | |
| IPRC | Initialized | — | — | — | — | — | Initialized | |
| IPRD | Initialized | — | — | — | — | — | Initialized | |
| IPRE | Initialized | — | — | — | — | — | Initialized | |
| IPRF | Initialized | — | — | — | — | — | Initialized | |
| IPRG | Initialized | — | — | — | — | — | Initialized | |
| IPRH | Initialized | — | — | — | — | — | Initialized | |
| IPRJ | Initialized | — | — | — | — | — | Initialized | |
| IPRK | Initialized | — | — | — | — | — | Initialized | |
| IPRM | Initialized | — | — | — | — | — | Initialized | |
| RAMER | Initialized | — | — | — | — | — | Initialized | ROM |
| P1DR | Initialized | — | — | — | — | — | — | PORT |
| P3DR | Initialized | — | — | — | — | — | — | |
| P7DR | Initialized | — | — | — | — | — | — | |
| PADR | Initialized | — | — | — | — | — | — | |
| PBDR | Initialized | — | — | — | — | — | — | |
| PCDR | Initialized | — | — | — | — | — | — | |
| PDDR | Initialized | — | — | — | — | — | — | |
| PFDR | Initialized | — | — | — | — | — | — | |
| TCR_0 | Initialized | — | — | — | — | — | Initialized | TPU_0 |
| TMDR_0 | Initialized | — | — | — | — | — | Initialized | |
| TIORH_0 | Initialized | — | — | — | — | — | Initialized | |
| TIORL_0 | Initialized | — | — | — | — | — | Initialized | |
| TIER_0 | Initialized | — | — | — | — | — | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| TSR_0 | Initialized | — | — | — | — | — | Initialized | TPU_0 |
| TCNTH_0 | Initialized | — | — | — | — | — | Initialized | |
| TCNTL_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRCH_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRCL_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRDH_0 | Initialized | — | — | — | — | — | Initialized | |
| TGRDL_0 | Initialized | — | — | — | — | — | Initialized | |
| TCR_1 | Initialized | — | — | — | — | — | Initialized | TPU_1 |
| TMDR_1 | Initialized | — | — | — | — | — | Initialized | |
| TIOR_1 | Initialized | — | — | — | — | — | Initialized | |
| TIER_1 | Initialized | — | — | — | — | — | Initialized | |
| TSR_1 | Initialized | — | — | — | — | — | Initialized | |
| TCNTH_1 | Initialized | — | — | — | — | — | Initialized | |
| TCNTL_1 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_1 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_1 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_1 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_1 | Initialized | — | — | — | — | — | Initialized | |
| TCR_2 | Initialized | — | — | — | — | — | Initialized | TPU_2 |
| TMDR_2 | Initialized | — | — | — | — | — | Initialized | |
| TIOR_2 | Initialized | — | — | — | — | — | Initialized | |
| TIER_2 | Initialized | — | — | — | — | — | Initialized | |
| TSR_2 | Initialized | — | — | — | — | — | Initialized | |
| TCNTH_2 | Initialized | — | — | — | — | — | Initialized | |
| TCNTL_2 | Initialized | — | — | — | — | — | Initialized | |
| TGRAH_2 | Initialized | — | — | — | — | — | Initialized | |
| TGRAL_2 | Initialized | — | — | — | — | — | Initialized | |
| TGRBH_2 | Initialized | — | — | — | — | — | Initialized | |
| TGRBL_2 | Initialized | — | — | — | — | — | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| TCR_0 | Initialized | — | — | — | — | — | Initialized | TMR_0 |
| TCR_1 | Initialized | — | — | — | — | — | Initialized | TMR_1 |
| TCSR_0 | Initialized | — | — | — | — | — | Initialized | |
| TCSR_1 | Initialized | — | — | — | — | — | Initialized | |
| TCORA_0 | Initialized | — | — | — | — | — | Initialized | TMR_0 |
| TCORA_1 | Initialized | — | — | — | — | — | Initialized | TMR_1 |
| TCORB_0 | Initialized | — | — | — | — | — | Initialized | |
| TCORB_1 | Initialized | — | — | — | — | — | Initialized | |
| TCNT_0 | Initialized | — | — | — | — | — | Initialized | |
| TCNT_1 | Initialized | — | — | — | — | — | Initialized | |
| TCSR_0 | Initialized | — | — | — | — | — | Initialized | WDT_0 |
| TCNT_0 | Initialized | — | — | — | — | — | Initialized | |
| RSTCSR | Initialized | — | — | — | — | — | Initialized | |
| SMR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | SCI_0 |
| BRR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SCR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TDR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| RDR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SCMR_0 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SMR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | SCI_2 |
| BRR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SCR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| TDR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SSR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| RDR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| SCMR_2 | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADDRAH | Initialized | — | — | — | Initialized | Initialized | Initialized | A/D |
| ADDRAL | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADDRBH | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADDRBL | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADDRCH | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADDRCL | Initialized | — | — | — | Initialized | Initialized | Initialized | |

| Register Abbreviation | Reset | High Speed | Medium Speed | Sleep | Module Stop | Software Standby | Hardware Standby | Module |
|---|---|---|---|---|---|---|---|---|
| ADDRDH | Initialized | — | — | — | Initialized | Initialized | Initialized | A/D |
| ADDRDL | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADCSR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| ADCR | Initialized | — | — | — | Initialized | Initialized | Initialized | |
| FLMCR1 | Initialized | — | — | — | — | — | Initialized | ROM |
| FLMCR2 | Initialized | — | — | — | — | — | Initialized | |
| EBR1 | Initialized | — | — | — | — | — | Initialized | |
| EBR2 | Initialized | — | — | — | — | — | Initialized | |
| PORT1 | Initialized | — | — | — | — | — | — | PORT |
| PORT3 | Initialized | — | — | — | — | — | — | |
| PORT4 | Initialized | — | — | — | — | — | — | |
| PORT7 | Initialized | — | — | — | — | — | — | |
| PORT9 | Initialized | — | — | — | — | — | — | |
| PORTA | Initialized | — | — | — | — | — | — | |
| PORTB | Initialized | — | — | — | — | — | — | |
| PORTC | Initialized | — | — | — | — | — | — | |
| PORTD | Initialized | — | — | — | — | — | — | |
| PORTF | Initialized | — | — | — | — | — | — | |

Note: — is not initialized.

## 23.1 Absolute Maximum Ratings

Table 23.1 lists the absolute maximum ratings.

**Table 23.1 Absolute Maximum Ratings**

| Item | Symbol | Value | Unit |
|---|---|---|---|
| Power supply voltage | $V_{CC}$ | –0.3 to +7.0 | V |
| Input voltage (XTAL, EXTAL) | $V_{in}$ | –0.3 to $V_{CC}$ +0.3 | V |
| Input voltage (port 4 and 9) | $V_{in}$ | –0.3 to $AV_{CC}$ +0.3 | V |
| Input voltage (except XTAL, EXTAL, port 4 and 9) | $V_{in}$ | –0.3 to $V_{CC}$ +0.3 | V |
| Analog power supply voltage | $AV_{CC}$ | –0.3 to +7.0 | V |
| Analog input voltage | $V_{AN}$ | –0.3 to $AV_{CC}$ +0.3 | V |
| Operating temperature | $T_{opr}$ | Regular specifications: –20 to +75 | °C |
| | | Wide-range specifications: –40 to +85 | °C |
| Storage temperature | $T_{stg}$ | –55 to +125 | °C |

Caution: Permanent damage to the chip may result if absolute maximum rating are exceeded.

Preliminary

## 23.2 DC Characteristics

Table 23.2 lists the DC characteristics. Table 23.3 lists the permissible output currents.

**Table 23.2 DC Characteristics**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)*[1]

| Item | | Symbol | Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Schmitt trigger input voltage | $\overline{IRQ0}$ to $\overline{IRQ5}$ | $V_T^-$ | $V_{CC} \times 0.2$ | — | — | V | |
| | | $V_T^+$ | — | — | $V_{CC} \times 0.7$ | V | |
| | | $V_T^+ - V_T^-$ | $V_{CC} \times 0.05$ | — | — | V | |
| Input high voltage | $\overline{RES}$, $\overline{STBY}$, NMI, MD2 to MD0, FWE | $V_{IH}$ | $V_{CC} \times 0.9$ | — | $V_{CC}$ + 0.3 | V | |
| | EXTAL | | $V_{CC} \times 0.7$ | — | $V_{CC}$ + 0.3 | V | |
| | Ports 1, 3, 7, A to D, F, HRxD | | $V_{CC} \times 0.7$ | — | $V_{CC}$ + 0.3 | V | |
| | Ports 4 and 9 | | $AV_{CC} \times 0.7$ | — | $AV_{CC}$ + 0.3 | V | |
| Input low voltage | $\overline{RES}$, $\overline{STBY}$, NMI, MD2 to MD0, FWE | $V_{IL}$ | –0.3 | — | $V_{CC} \times 0.1$ | V | |
| | EXTAL | | –0.3 | — | $V_{CC} \times 0.2$ | V | |
| | Ports 1, 3, 7, A to D, F, HRxD | | –0.3 | — | $V_{CC} \times 0.2$ | V | |
| | Ports 4, 9 | | –0.3 | — | $AV_{CC} \times 0.2$ | V | |
| Output high voltage | All output pins | $V_{OH}$ | $V_{CC}$ – 0.5 | — | — | V | $I_{OH}$ = –200 µA |
| | | | $V_{CC}$ – 1.0 | — | — | V | $I_{OH}$ = –1 mA |
| Output low voltage | All output pins | $V_{OL}$ | — | — | 0.4 | V | $I_{OL}$ = 1.6 mA |
| Input leakage current | $\overline{RES}$ | $\lvert I_{in} \rvert$ | — | — | 1.0 | µA | $V_{in}$ = 0.5 to $V_{CC}$ – 0.5 V |
| | $\overline{STBY}$, NMI, MD2 to MD0, FWE, HRxD | | — | — | 1.0 | µA | |
| | Ports 4, 9 | | — | — | 1.0 | µA | $V_{in}$ = 0.5 to $AV_{CC}$ – 0.5 V |

Preliminary

| Item | | Symbol | Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Input pull-up MOS current | Ports A to D | $-I_P$ | 30 | — | 300 | µA | $V_{in}$ = 0 V |
| Input capacitance | $\overline{RES}$ | $C_{in}$ | — | — | 30 | pF | $V_{in}$ = 0 V |
| | NMI | | — | — | 30 | pF | f = 1 MHz |
| | All input pins except $\overline{RES}$ and NMI | | — | — | 15 | pF | $T_a$ = 25°C |
| Current consumption*[2] | Normal operation | $I_{CC}$*[3] | — | TBD<br>$V_{CC}$ = 5.0 V | TBD<br>$V_{CC}$ = 5.5 V | mA | f = 24 MHz |
| | Sleep mode | | — | TBD<br>$V_{CC}$ = 5.0 V | TBD<br>$V_{CC}$ = 5.5 V | mA | f = 24 MHz |
| | All modules stopped | | — | TBD | — | mA | f = 24 MHz, $V_{CC}$ = 5.0 V (reference values) |
| | Medium-speed mode ($\phi$/32) | | — | TBD | — | mA | f = 24 MHz, $V_{CC}$ = 5.0 V (reference values) |
| | Standby mode | | — | TBD | TBD | µA | $T_a \leq 50°C$ |
| | | | — | — | TBD | µA | $50°C < T_a$ |
| Analog power supply current | During A/D conversion | $AI_{CC}$ | — | 2.5 | 4.0 | mA | $AV_{CC}$ = 5.0 V |
| | Idle | | — | — | 5.0 | µA | |
| RAM standby voltage | | $V_{RAM}$ | 2.0 | — | — | V | |

Notes: 1. If the A/D converter is not used, do not leave the $AV_{CC}$, and $AV_{SS}$ pins open. Apply a voltage between 4.5 V and 5.5 V to the $AV_{CC}$ pin by connecting them to $V_{CC}$, for instance.

2. Current consumption values are for $V_{IH} = V_{CC}$ (EXTAL), $AV_{CC}$ (ports 4 and 9), or $V_{CC}$ (other), and $V_{IL}$ = 0 V, with all output pins unloaded and the on-chip pull-up MOS transistors in the off state.

3. $I_{CC}$ depends on $V_{CC}$ and f as follows:
   $I_{CC}$ max = $27+0.435 \times V_{CC} \times f$ (normal operation)
   $I_{CC}$ max = $27+0.3 \times V_{CC} \times f$ (sleep mode)

Preliminary

**Table 23.3 Permissible Output Currents**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V,
$T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)*

| Item | | | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Permissible output low current (per pin) | All output pins | $V_{CC}$ = 4.5 to 5.5 V | $I_{OL}$ | — | — | 10 | mA |
| Permissible output low current (total) | Total of all output pins | $V_{CC}$ = 4.5 to 5.5 V | $\Sigma I_{OL}$ | — | — | 100 | mA |
| Permissible output high current (per pin) | All output pins | $V_{CC}$ = 4.5 to 5.5 V | $-I_{OH}$ | — | — | 2.0 | mA |
| Permissible output high current (total) | Total of all output pins | $V_{CC}$ = 4.5 to 5.5 V | $\Sigma -I_{OH}$ | — | — | 30 | mA |

Note: * To protect chip reliability, do not exceed the output current values in table 23.3.

## 23.3 AC Characteristics

Figure 23.1 shows the test conditions for the AC characteristics.

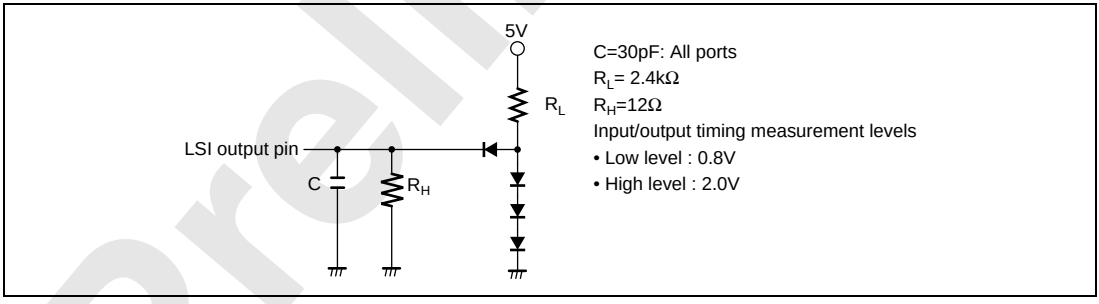



**Figure 23.1 Output Load Circuit**

Preliminary

### 23.3.1 Clock Timing

Table 23.4 lists the clock timing

**Table 23.4 Clock Timing**

Conditions : $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V, φ =4MHz to 24MHz, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)

| Item | Symbol | Min | Max | Unit | Test Conditions |
|---|---|---|---|---|---|
| Clock cycle time | $t_{cyc}$ | 41.6 | 250 | ns | Figure 23.2 |
| Clock high pulse width | $t_{CH}$ | TBD | — | ns | |
| Clock low pulse width | $t_{CL}$ | TBD | — | ns | |
| Clock rise time | $t_{Cr}$ | — | TBD | ns | |
| Clock fall time | $t_{Cf}$ | — | TBD | ns | |
| Oscillation settling time at reset (crystal) | $t_{OSC1}$ | 20 | — | ms | Figure 23.3 |
| Oscillation settling time in software standby (crystal) | $t_{OSC2}$ | 8 | — | ms | Figure 21.3 |
| External clock output settling delay time | $t_{DEXT}$ | 2 | — | ms | Figure 23.3 |

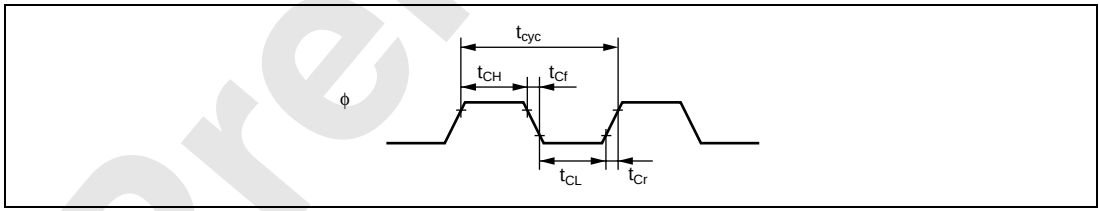



**Figure 23.2 System Clock Timing**

Preliminary

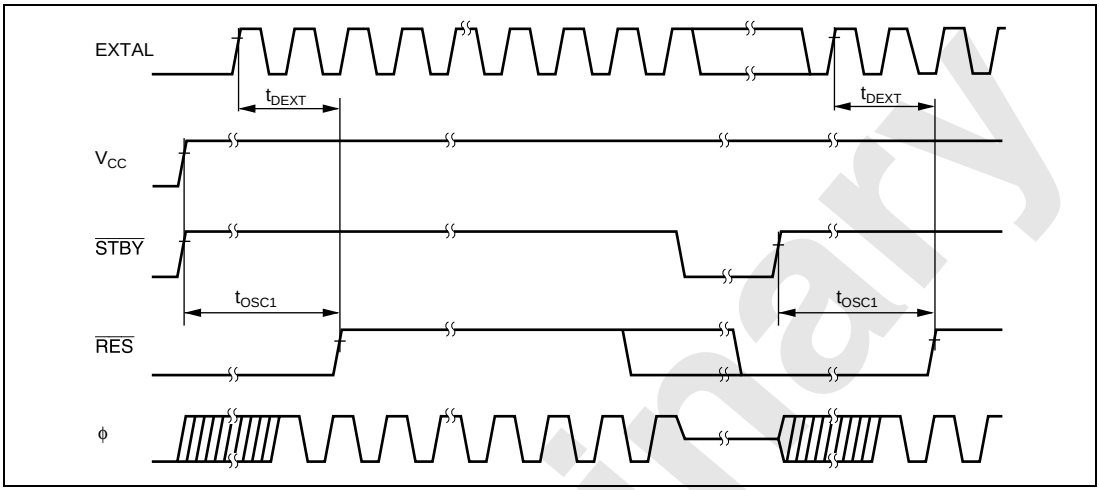



**Figure 23.3 Oscillation Settling Timing**

### 23.3.2 Control Signal Timing

Table 23.5 lists the control signal timing.

**Table 23.5 Control Signal Timing**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V, $\phi$ =4MHz to 24MHz, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)

| Item | Symbol | Min | Max | Unit | Test Conditions |
|---|---|---|---|---|---|
| $\overline{RES}$ setup time | $t_{RESS}$ | 200 | — | ns | Figure 23.4 |
| $\overline{RES}$ pulse width | $t_{RESW}$ | 20 | — | $t_{cyc}$ | |
| NMI setup time | $t_{NMIS}$ | 150 | — | ns | Figure 23.5 |
| NMI hold time | $t_{NMIH}$ | 10 | — | ns | |
| NMI pulse width (exiting software standby mode) | $t_{NMIW}$ | 200 | — | ns | |
| $\overline{IRQ}$ setup time | $t_{IRQS}$ | 150 | — | ns | |
| $\overline{IRQ}$ hold time | $t_{IRQH}$ | 10 | — | ns | |
| $\overline{IRQ}$ pulse width (exiting software standby mode) | $t_{IRQW}$ | 200 | — | ns | |

Preliminary

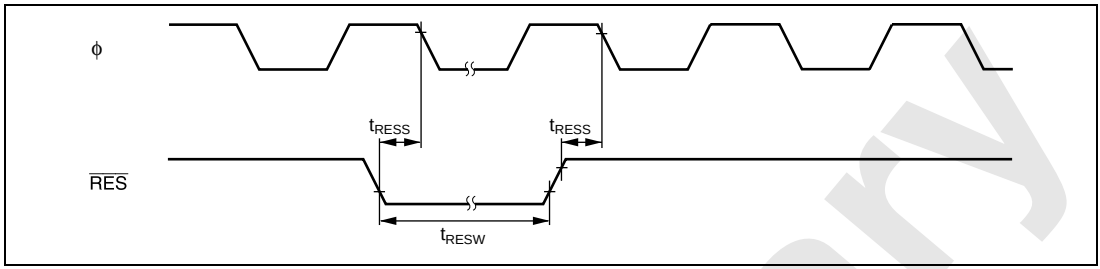



**Figure 23.4 Reset Input Timing**

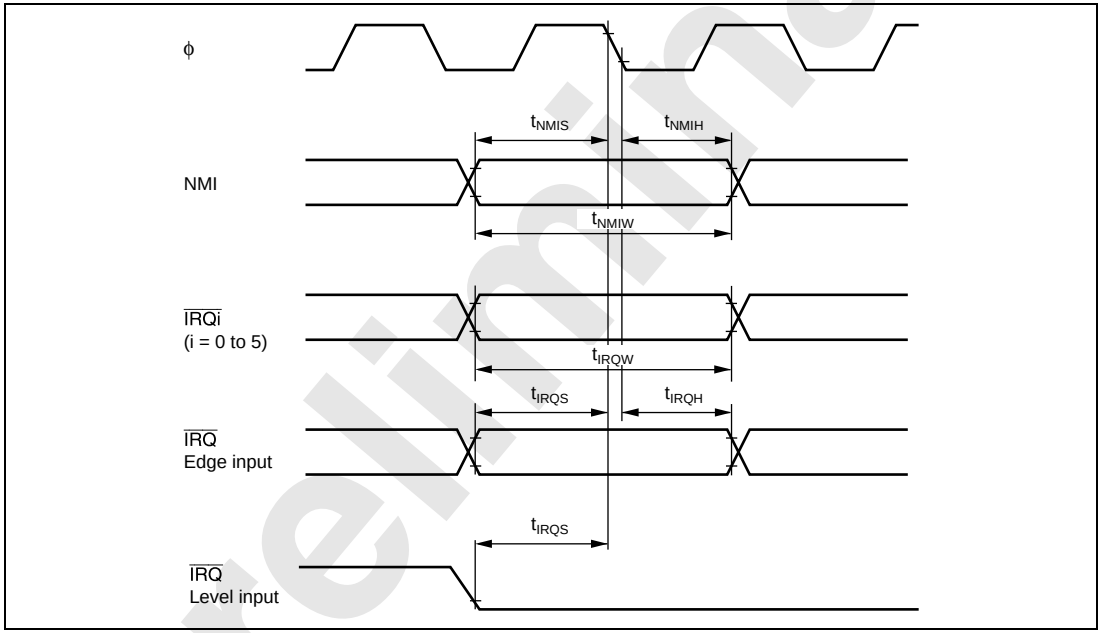



**Figure 23.5 Interrupt Input Timing**

Preliminary

### 23.3.3 Timing of On-Chip Peripheral Modules

Table 23.6 lists the timing of on-chip peripheral modules.

**Table 23.6 Timing of On-Chip Peripheral Modules**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 , $\phi$ =4MHz to 24MHz, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)

| Item | | | Symbol | Min | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| I/O port | Output data delay time | | $t_{PWD}$ | — | 40 | ns | Figure 23.6 |
| | Input data setup time | | $t_{PRS}$ | 25 | — | | |
| | Input data hold time | | $t_{PRH}$ | 25 | — | | |
| | Realtime input port data hold time | | $t_{RTIPH}$ | 4 | — | $t_{cyc}$ | Figure 23.7 |
| TPU | Timer output delay time | | $t_{TOCD}$ | — | 40 | ns | Figure 23.8 |
| | Timer input setup time | | $t_{TICS}$ | 25 | — | | |
| | Timer clock input setup time | | $t_{TCKS}$ | 25 | — | ns | Figure 23.9 |
| | Timer clock pulse width | Single edge | $t_{TCKWH}$ | 1.5 | — | $t_{cyc}$ | |
| | | Both edges | $t_{TCKWL}$ | 2.5 | — | | |
| SCI | Input clock cycle | Asynchro-nous | $t_{Scyc}$ | 4 | — | $t_{cyc}$ | Figure 23.10 |
| | | Synchro-nous | | 6 | — | | |
| | Input clock pulse width | | $t_{SCKW}$ | 0.4 | 0.6 | $t_{Scyc}$ | |
| | Input clock rise time | | $t_{SCKr}$ | — | 1.5 | $t_{cyc}$ | |
| | Input clock fall time | | $t_{SCKf}$ | — | 1.5 | | |
| | Transmit data delay time | | $t_{TXD}$ | — | 40 | ns | Figure 23.11 |
| | Receive data setup time (synchronous) | | $t_{RXS}$ | 40 | — | | |
| | Receive data hold time (synchronous) | | $t_{RXH}$ | 40 | — | | |

Preliminary

| Item | | Symbol | Min | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| A/D converter | Trigger input setup time | $t_{TRGS}$ | 30 | — | ns | Figure 23.12 |
| HCAN* | Transmit data delay time | $t_{HTXD}$ | — | 80 | ns | Figure 23.13 |
| | Transmit data setup time | $t_{HRXS}$ | 80 | — | | |
| | Transmit data hold time | $t_{HRXH}$ | 80 | — | | |
| PPG | Pulse output delay time | $t_{POD}$ | — | 40 | ns | Figure 23.14 |

Note:* The HCAN input signal is asynchronous. However, its state is judged to have changed at the rising-edge (two clock cycles) of the $\phi$ clock signal shown in figure 23.12. The HCAN output signal is also asynchronous. Its state changes based on the rising-edge (two clock cycles) of the $\phi$ clock signal shown in figure 23.12.

**Table 23.7 Timing of SSU**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 , $\phi$ =4MHz to 24MHz, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)

Preliminary

| Item | | | Symbol | Min | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| SSU | Clock cycle | Master<br>Slave | $t_{SUCYC}$ | 2<br>4 | 256<br>256 | $t_{CYC}$ | Figure 23.15<br>Figure 23.16<br>Figure 23.17<br>Figure 23.18 |
| | Clock high level pulse width | Master<br>Slave | $t_{HI}$ | 25<br>TBD | —<br>— | ns | |
| | Clock low level pulse width | Master<br>Slave | $t_{LO}$ | 25<br>TBD | —<br>— | ns | |
| | Clock rise time | Master<br>Slave | $t_{RISE}$ | —<br>— | 15<br>TBD | ns | |
| | Clock fall time | Master<br>Slave | $t_{FALL}$ | —<br>— | 25<br>TBD | ns | |
| | Data input setup time | Master<br>Slave | $t_{SU}$ | 30<br>TBD | —<br>— | ns | |
| | Data input hold time | Master<br>Slave | $t_{H}$ | 10<br>TBD | —<br>— | ns | |
| | SCS setup time | Master<br>Slave | $t_{LEAD}$ | 1<br>TBD | —<br>— | $t_{CYC}$ | |
| | SCS hold time | Master<br>Slave | $t_{LAG}$ | 1<br>TBD | —<br>— | $t_{CYC}$ | |
| | Data output delay time | Master<br>Slave | $t_{OD}$ | —<br>— | 40<br>TBD | ns | |
| | Data output hold time | Master<br>Slave | $t_{OH}$ | 0<br>TBD | —<br>— | ns | |
| | Continuous transfer delay time | Master<br>Slave | $t_{TD}$ | —<br>TBD | —<br>— | $t_{CYC}$ | |
| | Slave access time | | $t_{SA}$ | — | TBD | ns | |
| | Slave out release time | | $t_{REL}$ | — | TBD | ns | |

Preliminary

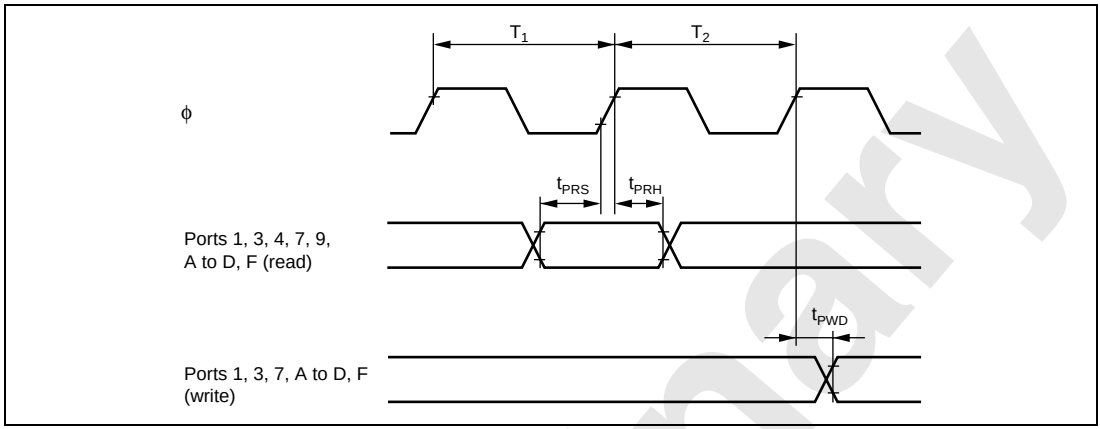



**Figure 23.6 I/O Port Input/Output Timing**

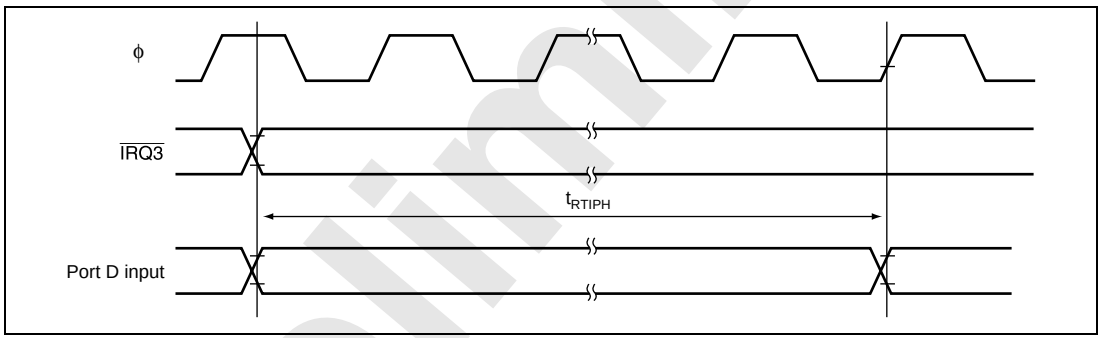



**Figure 23.7 Realtime Input Port Data Input Timing**

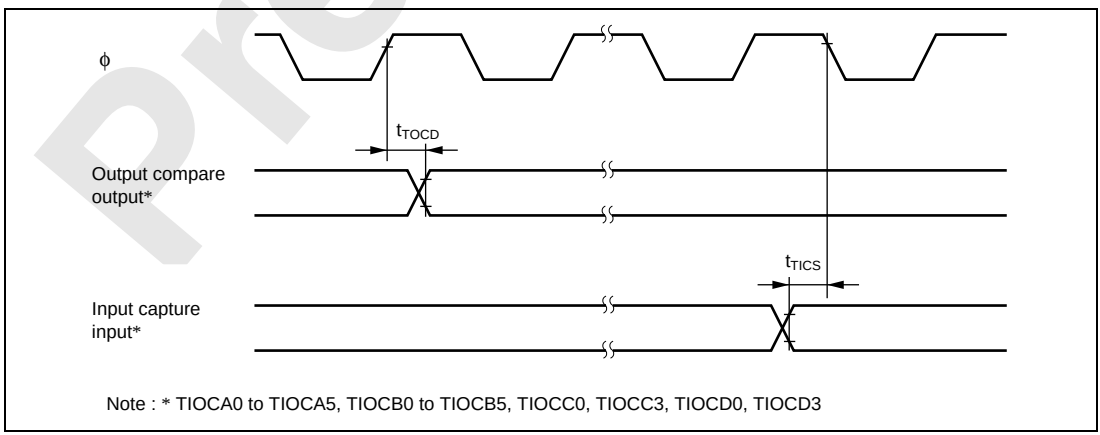



**Figure 23.8 TPU Input/Output Timing**

Preliminary

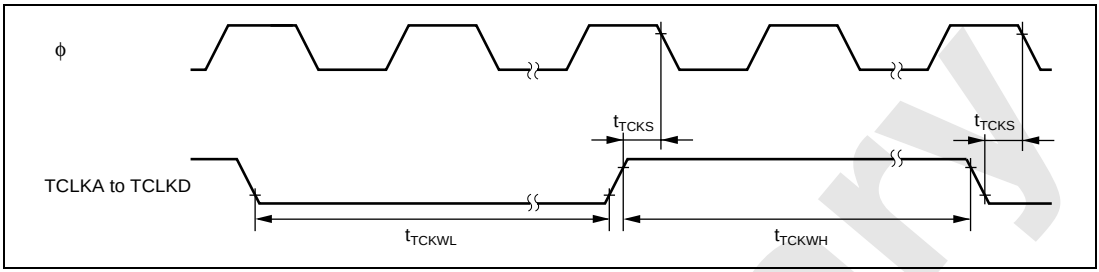



**Figure 23.9 TPU Clock Input Timing**

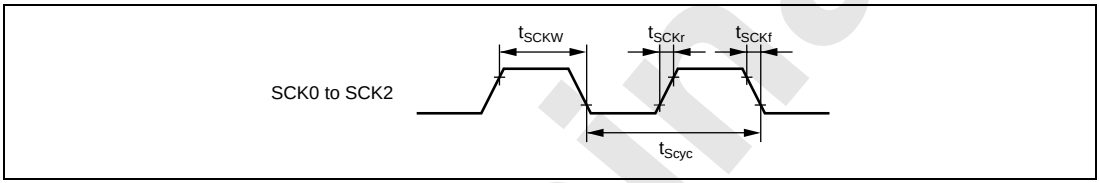



**Figure 23.10 SCK Clock Input Timing**

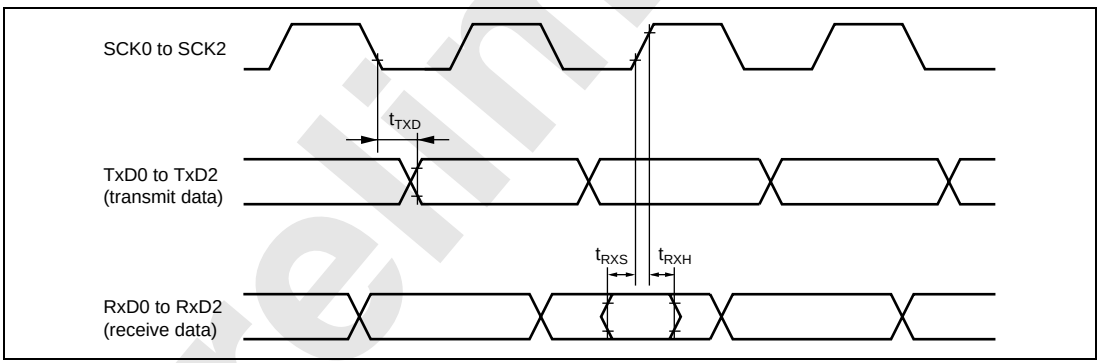



**Figure 23.11 SCI Input/Output Timing (Clocked Synchronous Mode)**

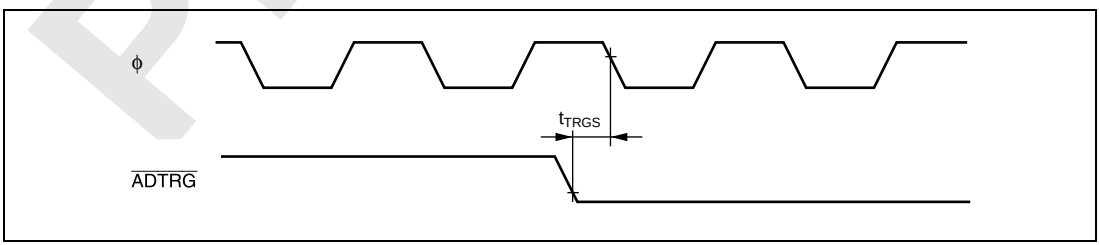



**Figure 23.12 A/D Converter External Trigger Input Timing**

Preliminary

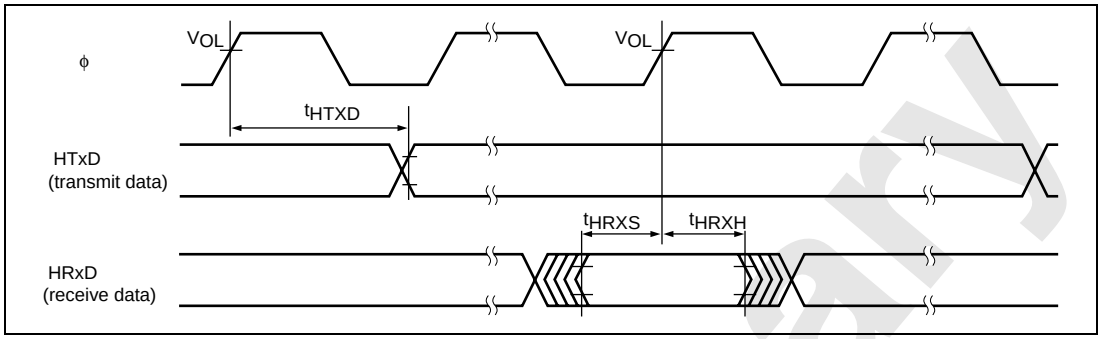



**Figure 23.13 HCAN Input/Output Timing**

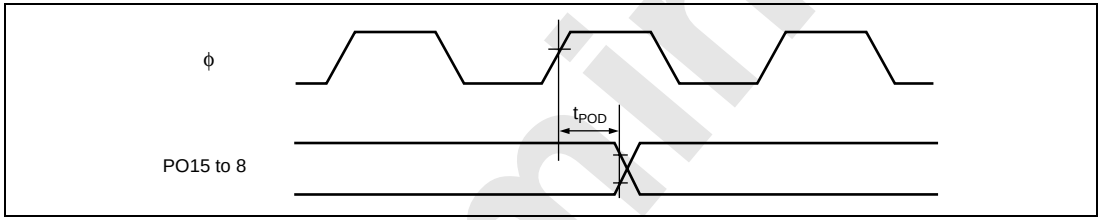



**Figure 23.14 PPG Output Timing**

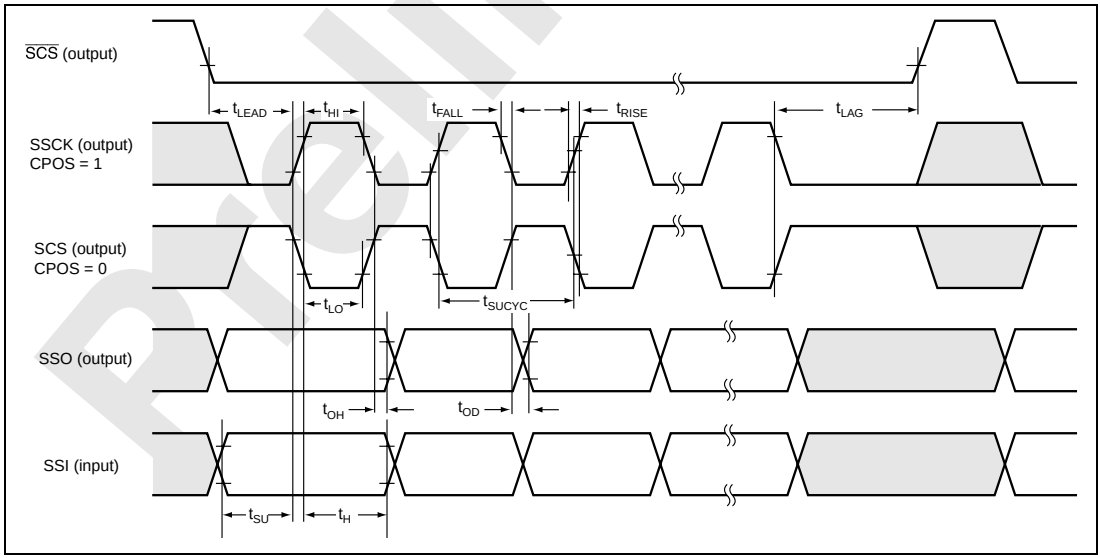



**Figure 23.15 SSU Timing (Master, CPHS = 1)**

Preliminary

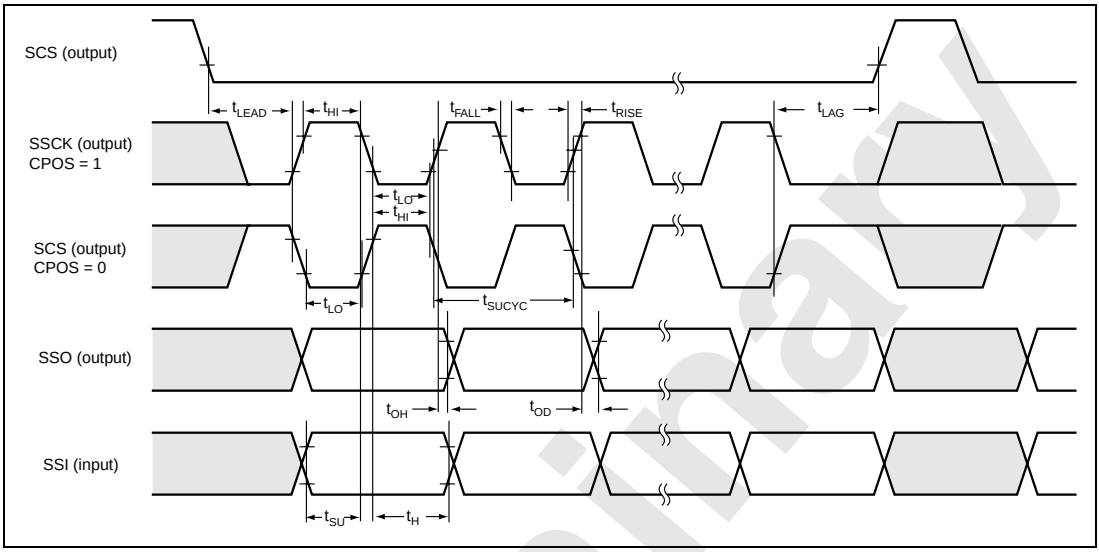



**Figure 23.16 SSU Timing (Master, CPHS = 0)**

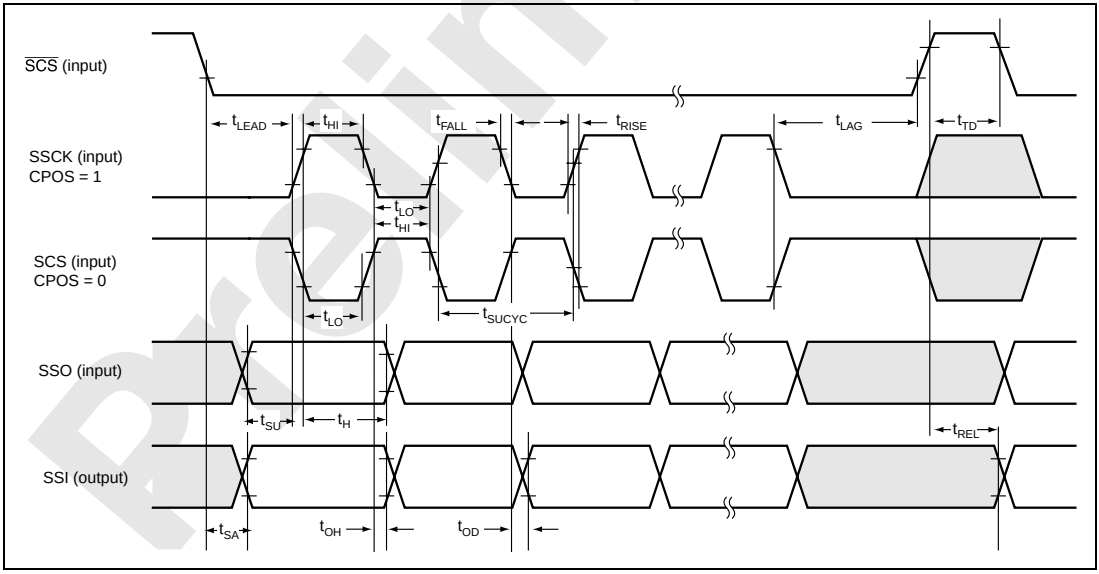



**Figure 23.17 SSU Timing (Slave, CPHS = 1)**

Preliminary

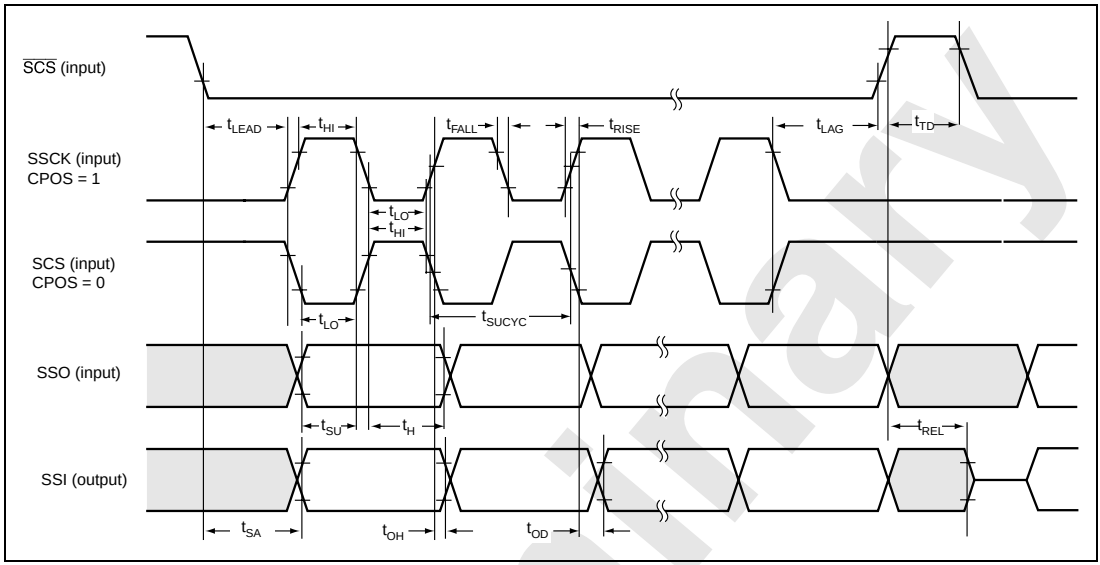



**Figure 23.18 SSU Timing (Slave, CPHS = 0)**

Preliminary

## 23.4 A/D Conversion Characteristics

Table 23.8 lists the A/D conversion characteristics.

**Table 23.8 A/D Conversion Characteristics**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0V, $\phi$ =4MHz to 24MHz, $T_a$ = –20°C to +75°C (regular specifications), $T_a$ = –40°C to +85°C (wide-range specifications)

| Item | Min | Typ | Max | Unit |
|---|---|---|---|---|
| Resolution | 10 | 10 | 10 | bits |
| Conversion time | 10 | — | 200 | µs |
| Analog input capacitance | — | — | 20 | pF |
| Permissible signal-source impedance | — | — | 5 | kΩ |
| Nonlinearity error | — | — | ±3.5 | LSB |
| Offset error | — | — | ±3.5 | LSB |
| Full-scale error | — | — | ±3.5 | LSB |
| Quantization | — | ±0.5 | — | LSB |
| Absolute accuracy | — | — | ±4.0 | LSB |

Preliminary

## 23.5 Flash Memory Characteristics

Table 23.9 lists the flash memory characteristics.

**Table 23.9 Flash Memory Characteristics**

Conditions: $V_{CC}$ = 4.5 V to 5.5 V, $AV_{CC}$ = 4.5 V to 5.5 V,
$V_{SS}$ = $PLLV_{SS}$ = $AV_{SS}$ = 0 V,
$T_a$ = 0 to +75°C (Programming / erasing operating temperature range)

| Item | | Symbol | Min | Typ | Max | Unit | Test Condition |
|---|---|---|---|---|---|---|---|
| Programming time*1, *2, *4 | | $t_P$ | — | 10 | 200 | ms/128 bytes | |
| Erase time*1, *3, *5 | | $t_E$ | — | 100 | 1200 | ms/block | |
| Reprogramming count | | $N_{WEC}$ | — | — | 100 | Times | |
| Programming | Wait time after SWE bit setting*1 | $t_{sswe}$ | 1 | 1 | — | µs | |
| | Wait time after PSU1 bit setting*1 | $t_{spsu}$ | 50 | 50 | — | µs | |
| | Wait time after P1 bit setting*1, *4 | $t_{sp30}$ | 28 | 30 | 32 | µs | Programming time wait |
| | | $t_{sp200}$ | 198 | 200 | 202 | µs | Programming time wait |
| | | $t_{sp10}$ | 8 | 10 | 12 | µs | Additional-programming time wait |
| | Wait time after P1 bit clear*1 | $t_{cp}$ | 5 | 5 | — | µs | |
| | Wait time after PSU1 bit clear*1 | $t_{cpsu}$ | 5 | 5 | — | µs | |
| | Wait time after PV1 bit setting*1 | $t_{spv}$ | 4 | 4 | — | µs | |
| | Wait time after H′FF dummy write*1 | $t_{spvr}$ | 2 | 2 | — | µs | |
| | Wait time after PV1 bit clear*1 | $t_{cpv}$ | 2 | 2 | — | µs | |
| | Wait time after SWE bit clear*1 | $t_{cswe}$ | 100 | 100 | — | µs | |
| | Maximum programming count*1, *4 | N | — | — | 1000 | Times | |
| Erase | Wait time after SWE bit setting*1 | $t_{sswe}$ | 1 | 1 | — | µs | |
| | Wait time after ESU1 bit setting*1 | $t_{sesu}$ | 100 | 100 | — | µs | |
| | Wait time after E1 bit setting*1, *5 | $t_{se}$ | 10 | 10 | 100 | ms | Erase time wait |
| | Wait time after E1 bit clear*1 | $t_{ce}$ | 10 | 10 | — | µs | |
| | Wait time after ESU1 bit clear*1 | $t_{cesu}$ | 10 | 10 | — | µs | |
| | Wait time after EV1 bit setting*1 | $t_{sev}$ | 20 | 20 | — | µs | |
| | Wait time after H′FF dummy write*1 | $t_{sevr}$ | 2 | 2 | — | µs | |
| | Wait time after EV1 bit clear*1 | $t_{cev}$ | 4 | 4 | — | µs | |
| | Wait time after SWE bit clear*1 | $t_{cswe}$ | 100 | 100 | — | µs | |
| | Maximum erase count*1, *5 | N | 12 | — | 120 | Times | |

Preliminary

Notes: 1. Make each time setting in accordance with the program/program-verify flowchart or erase/erase-verify flowchart.

2. Programming time per 128 bytes (shows the total period for which the P1 bit in the flash memory control register (FLMCR1) is set. It does not include the programming verification time.)

3. Block erase time (shows the total period for which the E1-bit FLMCR1 is set. It does not include the erase verification time.)

4. To specify the maximum programming time value (tp (max)) in the 128-bytes programming algorithm, set the max. value (1000) for the maximum programming count (n).
   The wait time after P1 bit setting should be changed as follows according to the value of the programming counter (n).
   Programming counter (n) = 1 to 6: $t_{sp30}$ = 30 µs
   Programming counter (n) = 7 to 1000: $t_{sp200}$ = 200 µs
   [In additional programming]
   Programming counter (n) = 1 to 6: $t_{sp10}$ = 10 µs

5. For the maximum erase time ($t_E$ (max)), the following relationship applies between the wait time after E1 bit setting ($t_{se}$) and the maximum erase count (N):
   $t_E$ (max) = Wait time after E1 bit setting ($t_{se}$) x maximum erase count (N)
   To set the maximum erase time, the values of ($t_{se}$) and (N) should be set so as to satisfy the above formula.
   Examples: When $t_{se}$ = 100 ms, N = 12 times
   When $t_{se}$ = 10 ms, N = 120 times

Preliminary

## A. I/O Port States in Each Pin State

| Port Name | MCU Operating Mode | Reset | Hardware Standby Mode | Software Standby Mode | Program Execution State Sleep Mode |
|---|---|---|---|---|---|
| Port 1 | 7 | T | T | Keep | I/O port |
| Port 3 | 7 | T | T | Keep | I/O port |
| Port 4 | 7 | T | T | T | Input port |
| Port 7 | 7 | T | T | Keep | I/O port |
| Port 9 | 7 | T | T | T | Input port |
| Port A | 7 | T | T | Keep | I/O port |
| Port B | 7 | T | T | Keep | I/O port |
| Port C | 7 | T | T | Keep | I/O port |
| Port D | 7 | T | T | Keep | I/O port |
| PF7 | 7 | T | T | [DDR = 0]<br>T<br>[DDR = 1]<br>H | [DDR = 0]<br>T<br>[DDR = 1]<br>Clock output |
| PF6<br>PF5<br>PF4<br>PF3<br>PF2<br>PF1<br>PF0 | 7 | T | T | Keep | I/O port |
| HTxD | 7 | H | T | H | Output |
| HRxD | 7 | Input | T | T | Input |

**Legend:**

H: High level

T: High impedance

Keep: Input port becomes high-impedance, output port retains state

## B. Product Code Lineup

| Product Type | | | Package (Hitachi Package Code) QFP-100 (FP-100M) |
|---|---|---|---|
| H8S/2628 | F-ZTAT version | Standard product | HD64F2628 |
| | Masked ROM version | Standard product | HD6432628 |
| H8S/2627 | Masked ROM version | Standard product | HD6432627 |

## C. Package Dimensions

The package dimension that is shown in the Hitachi Semiconductor Package Data Book has priority.

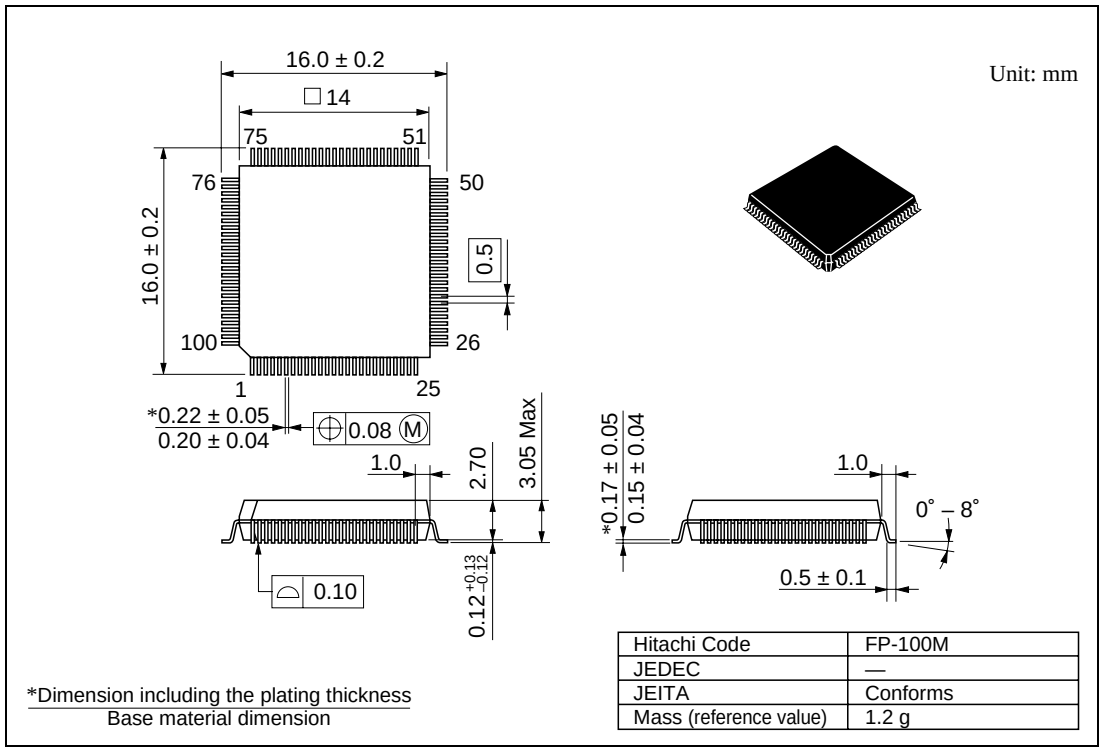



**Figure C.1 FP-100M Package Dimensions**

# Index


16-Bit Timer Pulse Unit (TPU) ..... 159
Buffer Operation ..... 204
Cascaded Operation ..... 208
Free-running count operation ..... 198
Input Capture ..... 201
periodic count operation ..... 198
Phase Counting Mode ..... 214
PWM Modes ..... 209
Synchronous Operation ..... 203
toggle output ..... 199
Waveform Output by Compare Match ..... 199
8-Bit Timers ..... 241
16-Bit Count Mode ..... 255
Cascaded Connection ..... 255
Compare-Match Count Mode ..... 256
Pulse Output ..... 251
TCNT Incrementation Timing ..... 252
Toggle output ..... 260
A/D Converter ..... 427
A/D Converter Activation ..... 223
A/D trigger input ..... 157
Conversion Time ..... 435
External Trigger ..... 437
Scan Mode ..... 434
Single Mode ..... 434
Address Map ..... 51
Address Space ..... 16
Addressing Modes ..... 38
Absolute Address ..... 39
Immediate ..... 40
Memory Indirect ..... 40
Program-Counter Relative ..... 40
Register Direct ..... 38
Register Indirect ..... 38
Register Indirect with Displacement .. 38
Register Indirect with Post-Increment 39
Register Indirect with Pre-Decrement 39
Bcc ..... 25, 34
Bit Rate ..... 386
break address ..... 85, 88
break conditions ..... 88
Bus Arbitration ..... 95
bus cycle ..... 93
Bus Masters ..... 95
Clock Pulse Generator ..... 467
Condition Field ..... 37
Condition-Code Register (CCR) ..... 20
CPU Operating Modes ..... 12
Advanced Mode ..... 13
Normal Mode ..... 12
data direction register ..... 121
data register ..... 121
Data Transfer Controller ..... 97
Activation by Software ..... 116
Block Transfer Mode ..... 111
Chain Transfer ..... 112, 117
DTC Vector Table ..... 104
Normal Mode ..... 109, 116
Register Information ..... 104
Repeat Mode ..... 110
software activation ..... 113
Software Activation ..... 118
vector number for the software activation interrupt ..... 103
Effective Address ..... 38, 41
Effective Address Extension ..... 37
Exception Handling ..... 53
Interrupts ..... 58
Reset Exception Handling ..... 55
Stack Status ..... 60
Traces ..... 58
Trap Instruction ..... 59
Extended Control Register (EXR) ..... 19
Flash memory ..... 445
Flash Memory
Boot Mode ..... 456
Emulation ..... 459
Erase/Erase-Verify ..... 463
erasing units ..... 450

Program/Program-Verify ..... 461
programming units ..... 450
Programming/Erasing in User Program Mode ..... 458
General Registers ..... 18
HCAN ..... 94, 355
11 consecutive recessive bits ..... 384
Arbitration field ..... 391, 394
buffer segment ..... 387
Configuration mode ..... 384
Control field ..... 391
Data field ..... 391
Data frame ..... 394
DTC Interface ..... 401
HCAN Halt Mode ..... 399
HCAN Sleep Mode ..... 396
mailbox ..... 380
Message Control (MC0 to MC15) ..... 380
Message Data (MD0 to MD15) ..... 382
Message transmission cancellation ..... 391
Message Transmission Method ..... 389
Remote frame ..... 395
remote transmission request bit ..... 395
Unread message overwrite ..... 395
input pull-up MOS ..... 121
Instruction Set ..... 25
Arithmetic Operations Instructions ..... 28
Bit Manipulation Instructions ..... 32
Block Data Transfer Instructions ..... 36
Branch Instructions ..... 34
Data Transfer Instructions ..... 27
Logic Operations Instructions ..... 30
Shift Instructions ..... 31
System Control Instructions ..... 35
Interrupt Control Modes ..... 76
Interrupt Controller ..... 63
Interrupt Exception Handling Vector Table ..... 72
Interrupt Mask Bit ..... 20
interrupt mask level ..... 19
interrupt priority register (IPR) ..... 63
Interrupts
ADI ..... 437
CMIA ..... 257
CMIB ..... 257
ERS0/OVR0 ..... 400
NMI ..... 71
OVI ..... 257
RM0 ..... 400
RM1 ..... 400
SLE0 ..... 400
SWDTEND ..... 113
TCIU_1 ..... 222
TCIU_2 ..... 222
TCIU_4 ..... 222
TCIU_5 ..... 222
TCIV_0 ..... 222
TCIV_1 ..... 222
TCIV_2 ..... 222
TCIV_3 ..... 222
TCIV_4 ..... 222
TCIV_5 ..... 222
TGIA_0 ..... 222
TGIA_1 ..... 222
TGIA_2 ..... 222
TGIA_3 ..... 222
TGIA_4 ..... 222
TGIA_5 ..... 222
TGIB_0 ..... 222
TGIB_1 ..... 222
TGIB_2 ..... 222
TGIB_3 ..... 222
TGIB_4 ..... 222
TGIB_5 ..... 222
TGIC_0 ..... 222
TGIC_3 ..... 222
TGID_0 ..... 222
TGID_3 ..... 222
WOVI ..... 288
MAC instruction ..... 49
memory cycle ..... 93
Multiply-Accumulate Register (MAC) ..... 21
On-Board Programming ..... 455
open-drain control register ..... 121

Operating Mode Selection ..... 47
Operation Field ..... 37
PC Break Controller ..... 85
Pin Arrangement ..... 3
PLL Circuit ..... 473
port register ..... 121
Program Counter (PC) ..... 19
Program/Erase Protection ..... 465
Programmable Pulse Generator ..... 263
Non-Overlapping Pulse Output ..... 276
output trigger ..... 270
Programmer Mode ..... 466
Register Field ..... 37
Registers
ABACK ..... 367, 494, 510, 528
ADCR ..... 433, 509, 526, 543
ADCSR ..... 431, 509, 526, 543
ADDR ..... 430, 508, 526, 542
BARA ..... 86, 503, 521, 537
BARB ..... 87, 503, 521, 537
BCR ..... 361, 494, 510, 528
BCRA ..... 87, 503, 521, 537
BCRB ..... 88, 503, 521, 537
BRR ..... 308, 508, 525, 542
CRA ..... 101
CRB ..... 102
DAR ..... 101
DTCER ..... 102, 504, 521, 538
DTVECR ..... 103, 504, 521, 538
EBR1 ..... 453, 509, 526, 543
EBR2 ..... 454, 509, 526, 543
FLMCR1 ..... 452, 509, 526, 543
FLMCR2 ..... 453, 509, 526, 543
GSR ..... 359, 494, 510, 528
HCANMON ..... 382, 502, 518, 536
IER ..... 67, 504, 521, 537
IMR ..... 375, 494, 510, 528
IPR ..... 66, 506, 523, 540
IRR ..... 370, 494, 510, 528
ISCR ..... 68, 503, 521, 537
ISR ..... 70, 504, 521, 537
LAFMH ..... 378, 494, 511, 528
LAFML ..... 378, 494, 511, 528
LPWRCR ..... 469, 503, 521, 537
MBCR ..... 363, 494, 510, 528
MBIMR ..... 374, 494, 510, 528
MC ..... 380, 494, 511, 528
MCR ..... 358, 494, 510, 528
MD ..... 382, 498, 515, 532
MDCR ..... 48, 503, 520, 537
MRA ..... 100
MRB ..... 101
MSTPCR ..... 482, 503, 520, 537
NDER ..... 266, 504, 521, 538
NDR ..... 268, 504, 521, 538
P1DDR ..... 125, 504, 522, 538
P1DR ..... 126, 506, 524, 540
PADDR ..... 138, 504, 522, 538
PADR ..... 139, 506, 524, 540
PAODR ..... 140, 505, 522, 538
PAPCR ..... 140, 504, 522, 538
PBDDR ..... 142, 504, 522, 538
PBDR ..... 143, 506, 524, 540
PBODR ..... 144, 505, 522, 538
PBPCR ..... 144, 504, 522, 538
PCDDR ..... 147, 504, 522, 538
PCDR ..... 147, 506, 524, 540
PCODR ..... 148, 505, 522, 539
PCPCR ..... 148, 504, 522, 538
PCR ..... 270, 504, 521, 538
PDDDR ..... 152, 504, 522, 538
PDDR ..... 153, 506, 524, 540
PDPCR ..... 153, 504, 522, 538
PFDDR ..... 155, 504, 522, 538
PFDR ..... 156, 506, 524, 540
PMR ..... 271, 504, 521, 538
PODR ..... 267, 504, 521, 538
PORT1 ..... 126, 509, 526, 543
PORT4 ..... 133, 509, 526, 543
PORT9 ..... 136, 509, 526, 543
PORTA ..... 139, 509, 527, 543
PORTB ..... 143, 509, 527, 543
PORTC ..... 148, 509, 527, 543
PORTD ..... 153, 509, 527, 543
PORTF ..... 156, 509, 527, 543
RAMER ..... 454, 506, 524, 540

RDR ..... 294, 508, 526, 542
REC ..... 376, 494, 510, 528
RFPR ..... 369, 494, 510, 528
RSR ..... 294
RSTCSR ..... 286, 508, 525, 542
RXPR..... 368, 494, 510, 528
SAR ..... 101
SBYCR ..... 480, 503, 520, 537
SCKCR ..... 468, 503, 520, 537
SCMR ..... 307, 508, 526, 542
SCR ..... 299, 508, 525, 542
SMR..... 295, 508, 525, 542
SSR .....c9yuy302, 508, 526, 542
SYSCR ..... 49, 503, 520, 537
TCNT..... 195, 244, 284, 507, 508, 522, 525, 542
TCOR ..... 244, 503, 520, 537
TCR ..... 244, 503, 520, 537
TCSR ..... 247, 503, 520, 537
TDR ..... 294, 508, 526, 542
TEC..... 376, 494, 510, 528
TGR ..... 205, 505, 522, 539
TIER ..... 190, 505, 522, 539
TIOR..... 173, 505, 522, 539
TMDR..... 171, 505, 522, 539
TSR ..... 294, 505, 522, 539
TSTR..... 195, 506, 523, 540
TSYR ..... 196, 506, 523, 540
TXACK..... 366, 494, 510, 528
TXCR..... 365, 494, 510, 528
TXPR ..... 364, 494, 510, 528
UMSR ..... 377, 494, 510, 528
Registres
TCNTH ..... 524
Reset..... 55
Serial Communication Interface ..... 291
Asynchronous Mode ..... 315
bit rate ..... 308
Break..... 353
framing error ..... 322
Mark State..... 353
overrun error ..... 322
parity error ..... 322
stack pointer (SP)..... 18
Time Quanta (TQ)..... 388
Trace Bit..... 19
TRAPA instruction ..... 40, 59
Watchdog Timer ..... 283
Interval Timer Mode ..... 287

**H8S/2628 Series Hardware Manual**

Publication Date: 1st. Edition, September 2002
Published by: Business Operation Division
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.
Copyright © Hitachi, Ltd., 2002. All rights reserved. Printed in Japan.