Hitachi SuperH™ RISC engine

# SH7612

## HD6417612

Hardware Manual

HITACHI

ADE-602-153
Rev. 1.0
9/17/99
Hitachi, Ltd.

Hitachi semiconductor

## 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.

# Preface

The SH7612 implements high-performance operations by using a CPU which employs the Reduced Instruction Set Computer (RISC) system. This is a new-generation RISC microcomputer which realizes low power consumption, an essential feature of microcomputer devices, as well as integrating peripheral features necessary for system configuration.

The CPU has a set of RISC-type instructions; basic instructions operate at one state per instruction, that is, in one system clock cycle, dramatically increasing execution speeds. The chip incorporates a 32-bit multiplier which performs high-speed sum-of-product (multiply-and-accumulate) operations. Instructions are upwardly compatible with the SH7000 Series, allowing easy migration from the SH7000 Series.

Moreover, the chip incorporates on-chip peripheral modules such as an interrupt controller (INTC), direct memory access controller (DMAC), division unit (DIVU), timers (FRT, WDT), and serial communication interface (SCI), so that a user system can be configured using the minimum number of parts.

On-chip cache memory enhances the CPU throughput. A bus control feature, which supports external memory access, improves external memory access efficiency, allowing direct connection to synchronous DRAM, DRAM, and pseudo-SRAM without the help of glue logic.

This hardware manual explains the hardware features of the chip. For details of instructions, see the programming manual.

Related Documents:
*SH-1/SH-2/SH-DSP Programming Manual*

For the development environment system, call your nearest Hitachi sales office.

# Contents


Section 1 Overview ..... 1
1.1 Features ..... 1
1.2 Architecture ..... 3
1.3 Pin Arrangement ..... 6
1.4 Pins ..... 8
1.5 CPU ..... 15
1.5.1 Fetch and Decode ..... 15
1.5.2 Integer Unit ..... 15
1.5.3 DSP Unit ..... 16
1.6 Peripheral Module Units ..... 17
1.6.1 System Controller (SYSC) ..... 17
1.6.2 Interrupt Controller (INTC) ..... 18
1.6.3 User Break Controller (UBC) ..... 18
1.6.4 Bus State Controller (BSC) ..... 19
1.6.5 Cache ..... 20
1.6.6 Direct Memory Access Controller (DMAC) ..... 20
1.6.7 Division Unit (DIVU) ..... 21
1.6.8 16-Bit Free-Running Timer (FRT) ..... 22
1.6.9 Watchdog Timer (WDT) ..... 22
1.6.10 Serial Communication Interface (SCI) ..... 23
1.6.11 Smart Card Interface ..... 24
1.6.12 Serial Communication Interface with FIFO (SCIF) ..... 24
1.6.13 IrDA ..... 25
1.6.14 Serial I/O (SIO) ..... 26
1.6.15 16-Bit Timer Pulse Unit (TPU) ..... 26
1.6.16 Hitachi User Debug Interface (H-UDI) ..... 27
1.6.17 Pin Function Controller (PFC) ..... 28
1.6.18 I/O Ports ..... 28
1.6.19 Processing States ..... 29

Section 2 CPU ..... 33
2.1 Register Configuration ..... 33
2.1.1 General Registers ..... 33
2.1.2 Control Registers ..... 35
2.1.3 System Registers ..... 38
2.1.4 DSP Registers ..... 39
2.1.5 Notes on Guard Bits and Overflow Treatment ..... 42
2.1.6 Initial Values of Registers ..... 43
2.2 Data Formats ..... 44

2.2.1 Data Format in Registers ... 44
2.2.2 Data Formats in Memory ... 44
2.2.3 Immediate Data Format ... 45
2.2.4 DSP Type Data Formats ... 45
2.2.5 DSP Type Instructions and Data Formats ... 47
2.3 CPU Core Instruction Features ... 51
2.4 Instruction Formats ... 55
2.4.1 CPU Instruction Addressing Modes ... 55
2.4.2 DSP Data Addressing ... 59
2.4.3 Instruction Formats for CPU Instructions ... 65
2.4.4 Instruction Formats for DSP Instructions ... 69
2.5 Instruction Set ... 75
2.5.1 CPU Instruction Set ... 76
2.5.2 DSP Data Transfer Instruction Set ... 92
2.5.3 DSP Operation Instruction Set ... 96
2.5.4 Various Operation Instructions ... 99

Section 3 Oscillator Circuits and Operating Modes ... 109
3.1 Overview ... 109
3.2 On-Chip Clock Pulse Generator and Operating Modes ... 109
3.2.1 Clock Pulse Generator ... 109
3.2.2 Clock Operating Mode Settings ... 111
3.2.3 Connecting a Crystal Resonator ... 114
3.2.4 External Clock Input ... 115
3.2.5 Operating Frequency Selection by Register ... 116
3.2.6 Clock Modes and Frequency Ranges ... 128
3.2.7 Notes on Board Design ... 129
3.3 Bus Width of the CS0 Area ... 130

Section 4 Exception Handling ... 131
4.1 Overview ... 131
4.1.1 Types of Exception Handling and Priority Order ... 131
4.1.2 Exception Handling Operations ... 133
4.1.3 Exception Vector Table ... 134
4.2 Resets ... 136
4.2.1 Types of Resets ... 136
4.2.2 Power-On Reset ... 137
4.2.3 Manual Reset ... 137
4.3 Address Errors ... 137
4.3.1 Sources of Address Errors ... 137
4.3.2 Address Error Exception Handling ... 139
4.4 Interrupts ... 139
4.4.1 Interrupt Sources ... 139

4.4.2 Interrupt Priority Levels ..... 140
4.4.3 Interrupt Exception Handling ..... 141
4.5 Exceptions Triggered by Instructions ..... 141
4.5.1 Instruction-Triggered Exception Types ..... 141
4.5.2 Trap Instructions ..... 142
4.5.3 Illegal Slot Instructions ..... 142
4.5.4 General Illegal Instructions ..... 142
4.6 When Exception Sources Are Not Accepted ..... 143
4.6.1 Immediately after a Delayed Branch Instruction ..... 143
4.6.2 Immediately after an Interrupt-Disabled Instruction ..... 143
4.6.3 Instructions in Repeat Loops ..... 144
4.7 Stack Status after Exception Handling ..... 145
4.8 Usage Notes ..... 145
4.8.1 Value of Stack Pointer (SP) ..... 145
4.8.2 Value of Vector Base Register (VBR) ..... 145
4.8.3 Address Errors Caused by Stacking of Address Error Exception Handling ..... 146
4.8.4 Manual Reset during Register Access ..... 146

Section 5 Interrupt Controller (INTC) ..... 147
5.1 Overview ..... 147
5.1.1 Features ..... 147
5.1.2 Block Diagram ..... 147
5.1.3 Pin Configuration ..... 149
5.1.4 Register Configuration ..... 149
5.2 Interrupt Sources ..... 150
5.2.1 NMI Interrupt ..... 151
5.2.2 User Break Interrupt ..... 151
5.2.3 H-UDI Interrupt ..... 151
5.2.4 IRL Interrupts ..... 151
5.2.5 IRQ Interrupts ..... 152
5.2.6 On-chip Peripheral Module Interrupts ..... 156
5.2.7 Interrupt Exception Vectors and Priority Order ..... 156
5.3 Register Descriptions ..... 163
5.3.1 Interrupt Priority Level Setting Register A (IPRA) ..... 163
5.3.2 Interrupt Priority Level Setting Register B (IPRB) ..... 164
5.3.3 Interrupt Priority Level Setting Register C (IPRC) ..... 165
5.3.4 Interrupt Priority Level Setting Register D (IPRD) ..... 166
5.3.5 Interrupt Priority Level Setting Register E (IPRE) ..... 167
5.3.6 Vector Number Setting Register WDT (VCRWDT) ..... 168
5.3.7 Vector Number Setting Register A (VCRA) ..... 169
5.3.8 Vector Number Setting Register B (VCRB) ..... 170
5.3.9 Vector Number Setting Register C (VCRC) ..... 171
5.3.10 Vector Number Setting Register D (VCRD) ..... 172

5.3.11 Vector Number Setting Register E (VCRE) ..... 173
5.3.12 Vector Number Setting Register F (VCRF) ..... 174
5.3.13 Vector Number Setting Register G (VCRG) ..... 175
5.3.14 Vector Number Setting Register H (VCRH) ..... 176
5.3.15 Vector Number Setting Register I (VCRI) ..... 177
5.3.16 Vector Number Setting Register J (VCRJ) ..... 178
5.3.17 Vector Number Setting Register K (VCRK) ..... 179
5.3.18 Vector Number Setting Register L (VCRL) ..... 180
5.3.19 Vector Number Setting Register M (VCRM) ..... 181
5.3.20 Vector Number Setting Register N (VCRN) ..... 182
5.3.21 Vector Number Setting Register O (VCRO) ..... 183
5.3.22 Vector Number Setting Register P (VCRP) ..... 184
5.3.23 Vector Number Setting Register Q (VCRQ) ..... 185
5.3.24 Vector Number Setting Register R (VCRR) ..... 186
5.3.25 Vector Number Setting Register S (VCRS) ..... 187
5.3.26 Vector Number Setting Register T (VCRT) ..... 188
5.3.27 Vector Number Setting Register U (VCRU) ..... 189
5.3.28 Interrupt Control Register (ICR) ..... 192
5.3.29 IRQ Control/Status Register (IRQCSR) ..... 193
5.4 Interrupt Operation ..... 195
5.4.1 Interrupt Sequence ..... 195
5.4.2 Stack State after Interrupt Exception Handling ..... 197
5.5 Interrupt Response Time ..... 198
5.6 Sampling of Pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ ..... 199
5.7 Usage Notes ..... 200

Section 6 User Break Controller (UBC) ..... 205
6.1 Overview ..... 205
6.1.1 Features ..... 205
6.1.2 Block Diagram ..... 206
6.1.3 Register Configuration ..... 207
6.2 Register Descriptions ..... 208
6.2.1 Break Address Register A (BARA) ..... 208
6.2.2 Break Address Mask Register A (BAMRA) ..... 209
6.2.3 Break Bus Cycle Register A (BBRA) ..... 210
6.2.4 Break Address Register B (BARB) ..... 212
6.2.5 Break Address Mask Register B (BAMRB) ..... 213
6.2.6 Break Data Register B (BDRB) ..... 215
6.2.7 Break Data Mask Register B (BDMRB) ..... 216
6.2.8 Break Bus Cycle Register B (BBRB) ..... 218
6.2.9 Break Control Register (BRCR) ..... 219
6.2.10 Execution Times Break Register (BETR) ..... 222
6.2.11 Branch Source Register (BRSR) ..... 223

6.2.12 Branch Destination Register (BRDR) ..... 224
6.2.13 Branch Flag Register (BRFR) ..... 225
6.3 Operation ..... 226
6.3.1 Flow of the User Break Operation ..... 226
6.3.2 Break on Instruction Fetch Cycle ..... 227
6.3.3 Break on Data Access Cycle ..... 227
6.3.4 Program Counter (PC) Values Saved ..... 228
6.3.5 X Memory or Y Memory Bus Cycle Breaks ..... 229
6.3.6 Sequential Breaks ..... 229
6.3.7 PC Trace ..... 230
6.3.8 Example of Use ..... 232
6.4 Usage Notes ..... 235

Section 7 Bus State Controller (BSC) ..... 237
7.1 Overview ..... 237
7.1.1 Features ..... 237
7.1.2 Block Diagram ..... 239
7.1.3 Pin Configuration ..... 240
7.1.4 Register Configuration ..... 241
7.1.5 Address Map ..... 242
7.2 Register Descriptions ..... 244
7.2.1 Bus Control Register 1 (BCR1) ..... 244
7.2.2 Bus Control Register 2 (BCR2) ..... 247
7.2.3 Bus Control Register 3 (BCR3) ..... 248
7.2.4 Wait Control Register 1 (WCR1) ..... 250
7.2.5 Wait Control Register 2 (WCR2) ..... 251
7.2.6 Wait Control Register 3 (WCR3) ..... 253
7.2.7 Individual Memory Control Register (MCR) ..... 254
7.2.8 Refresh Timer Control/Status Register (RTCSR) ..... 259
7.2.9 Refresh Timer Counter (RTCNT) ..... 261
7.2.10 Refresh Time Constant Register (RTCOR) ..... 261
7.3 Access Size and Data Alignment ..... 262
7.3.1 Connection to Ordinary Devices ..... 262
7.3.2 Connection to Little-Endian Devices ..... 263
7.4 Accessing Ordinary Space ..... 266
7.4.1 Basic Timing ..... 266
7.4.2 Wait State Control ..... 271
7.4.3 $\overline{CS}$ Assertion Period Extension ..... 275
7.5 Synchronous DRAM Interface ..... 276
7.5.1 Synchronous DRAM Direct Connection ..... 276
7.5.2 Address Multiplexing ..... 278
7.5.3 Burst Reads ..... 280
7.5.4 Single Reads ..... 283

7.5.5 Single Writes ... 285
7.5.6 Burst Write Mode ... 287
7.5.7 Bank Active Function ... 289
7.5.8 Refreshes ... 296
7.5.9 Power-On Sequence ... 299
7.6 DRAM Interface ... 301
7.6.1 DRAM Direct Connection ... 301
7.6.2 Address Multiplexing ... 302
7.6.3 Basic Timing ... 303
7.6.4 Wait State Control ... 304
7.6.5 Burst Access ... 306
7.6.6 EDO Mode ... 309
7.6.7 DRAM Single Transfer ... 313
7.6.8 Refreshing ... 314
7.6.9 Power-On Sequence ... 316
7.7 Burst ROM Interface ... 316
7.8 Idles between Cycles ... 319
7.9 Bus Arbitration ... 320
7.9.1 Master Mode ... 321
7.10 Additional Items ... 323
7.10.1 Resets ... 323
7.10.2 Access as Viewed from CPU or DMAC ... 323
7.10.3 On-Chip Peripheral Module Access ... 325
7.10.4 Notes on Memory Access when Using the DMAC ... 325
7.10.5 Additional Note ... 331

Section 8 Cache ... 333
8.1 Introduction ... 333
8.1.1 Register Configuration ... 334
8.2 Register Description ... 334
8.2.1 Cache Control Register (CCR) ... 334
8.3 Address Space and the Cache ... 336
8.4 Cache Operation ... 337
8.4.1 Cache Reads ... 337
8.4.2 Write Access ... 339
8.4.3 Cache-Through Access ... 340
8.4.4 The TAS Instruction ... 340
8.4.5 Pseudo-LRU and Cache Replacement ... 340
8.4.6 Cache Initialization ... 342
8.4.7 Associative Purges ... 342
8.4.8 Data Array Access ... 343
8.4.9 Address Array Access ... 344
8.5 Cache Use ... 345

8.5.1 Initialization ... 345
8.5.2 Purge of Specific Lines ... 345
8.5.3 Cache Data Coherency ... 346
8.5.4 Two-Way Cache Mode ... 347
8.6 Usage Notes ... 348
8.6.1 Standby ... 348
8.6.2 Cache Control Register ... 348
8.6.3 Cache Initialization ... 348

Section 9 Direct Memory Access Controller (DMAC) ... 349
9.1 Overview ... 349
9.1.1 Features ... 349
9.1.2 Block Diagram ... 351
9.1.3 Pin Configuration ... 352
9.1.4 Register Configuration ... 352
9.2 Register Descriptions ... 353
9.2.1 DMA Source Address Registers 0 and 1 (SAR0, SAR1) ... 353
9.2.2 DMA Destination Address Registers 0 and 1 (DAR0, DAR1) ... 354
9.2.3 DMA Transfer Count Registers 0 and 1 (TCR0, TCR1) ... 354
9.2.4 DMA Channel Control Registers 0 and 1 (CHCR0, CHCR1) ... 355
9.2.5 DMA Vector Number Registers 0 and 1 (VCRDMA0, VCRDMA1) ... 359
9.2.6 DMA Request/Response Selection Control Registers 0 and 1 (DRCR0, DRCR1) ... 360
9.2.7 DMA Operation Register (DMAOR) ... 361
9.3 Operation ... 363
9.3.1 DMA Transfer Flow ... 363
9.3.2 DMA Transfer Requests ... 365
9.3.3 Channel Priorities ... 367
9.3.4 DMA Transfer Types ... 369
9.3.5 Number of Bus Cycles ... 376
9.3.6 DMA Transfer Request Acknowledge Signal Output Timing ... 376
9.3.7 DREQ Pin Input Detection Timing ... 385
9.3.8 DMA Transfer End ... 394
9.4 Usage Notes ... 395
9.4.1 DMAC-Related Notes ... 395
9.4.2 Notes on Memory Access when Using the DMAC ... 398

Section 10 Division Unit (DIVU) ... 405
10.1 Overview ... 405
10.1.1 Features ... 405
10.1.2 Block Diagram ... 406
10.1.3 Register Configuration ... 407
10.2 Register Descriptions ... 407

10.2.1 Divisor Register (DVSR) ..... 407
10.2.2 Dividend Register L for 32-Bit Division (DVDNT) ..... 407
10.2.3 Division Control Register (DVCR) ..... 408
10.2.4 Vector Number Setting Register DIV (VCRDIV) ..... 409
10.2.5 Dividend Register H (DVDNTH) ..... 409
10.2.6 Dividend Register L (DVDNTL) ..... 410
10.3 Operation ..... 410
10.3.1 64-Bit ÷ 32-Bit Operations ..... 410
10.3.2 32-Bit ÷ 32-Bit Operations ..... 410
10.3.3 Handling of Overflows ..... 411
10.4 Usage Notes ..... 411
10.4.1 Access ..... 411
10.4.2 Overflow Flag ..... 412

Section 11 16-Bit Free-Running Timer (FRT) ..... 413
11.1 Overview ..... 413
11.1.1 Features ..... 413
11.1.2 Block Diagram ..... 414
11.1.3 Pin Configuration ..... 415
11.1.4 Register Configuration ..... 415
11.2 Register Descriptions ..... 416
11.2.1 Free-Running Counter (FRC) ..... 416
11.2.2 Output Compare Registers A and B (OCRA and OCRB) ..... 416
11.2.3 Input Capture Register (FICR) ..... 417
11.2.4 Timer Interrupt Enable Register (TIER) ..... 417
11.2.5 Free-Running Timer Control/Status Register (FTCSR) ..... 418
11.2.6 Timer Control Register (TCR) ..... 420
11.2.7 Timer Output Compare Control Register (TOCR) ..... 421
11.3 CPU Interface ..... 422
11.4 Operation ..... 425
11.4.1 FRC Count Timing ..... 425
11.4.2 Output Timing for Output Compare ..... 426
11.4.3 FRC Clear Timing ..... 426
11.4.4 Input Capture Input Timing ..... 427
11.4.5 Input Capture Flag (ICF) Setting Timing ..... 428
11.4.6 Output Compare Flag (OCFA, OCFB) Setting Timing ..... 428
11.4.7 Timer Overflow Flag (OVF) Setting Timing ..... 429
11.5 Interrupt Sources ..... 430
11.6 Example of FRT Use ..... 430
11.7 Usage Notes ..... 431
11.7.1 Contention between FRC Write and Clear ..... 431
11.7.2 Contention between FRC Write and Increment ..... 431
11.7.3 Contention between OCR Write and Compare Match ..... 432

11.7.4 Internal Clock Switching and Counter Operation ..... 433
11.7.5 Timer Output (FTOA, FTOB) ..... 435

Section 12 Watchdog Timer (WDT) ..... 437
12.1 Overview ..... 437
12.1.1 Features ..... 437
12.1.2 Block Diagram ..... 438
12.1.3 Pin Configuration ..... 438
12.1.4 Register Configuration ..... 439
12.2 Register Descriptions ..... 439
12.2.1 Watchdog Timer Counter (WTCNT) ..... 439
12.2.2 Watchdog Timer Control/Status Register (WTCSR) ..... 440
12.2.3 Reset Control/Status Register (RSTCSR) ..... 441
12.2.4 Notes on Register Access ..... 442
12.3 Operation ..... 444
12.3.1 Operation in Watchdog Timer Mode ..... 444
12.3.2 Operation in Interval Timer Mode ..... 446
12.3.3 Operation when Standby Mode is Cleared ..... 446
12.3.4 Timing of Overflow Flag (OVF) Setting ..... 447
12.3.5 Timing of Watchdog Timer Overflow Flag (WOVF) Setting ..... 447
12.4 Usage Notes ..... 448
12.4.1 Contention between WTCNT Write and Increment ..... 448
12.4.2 Changing CKS2 to CKS0 Bit Values ..... 448
12.4.3 Switching between Watchdog Timer Mode and Interval Timer Mode ..... 448
12.4.4 System Reset with $\overline{\text{WDTOVF}}$ ..... 449
12.4.5 Internal Reset in Watchdog Timer Mode ..... 449

Section 13 Serial Communication Interface (SCI) ..... 451
13.1 Overview ..... 451
13.1.1 Features ..... 451
13.1.2 Block Diagram ..... 452
13.1.3 Pin Configuration ..... 453
13.1.4 Register Configuration ..... 453
13.2 Register Descriptions ..... 454
13.2.1 Receive Shift Register (RSR) ..... 454
13.2.2 Receive Data Register (RDR) ..... 454
13.2.3 Transmit Shift Register (TSR) ..... 454
13.2.4 Transmit Data Register (TDR) ..... 455
13.2.5 Serial Mode Register (SMR) ..... 455
13.2.6 Serial Control Register (SCR) ..... 458
13.2.7 Serial Status Register (SSR) ..... 461
13.2.8 Bit Rate Register (BRR) ..... 465
13.3 Operation ..... 472

13.3.1 Overview ..... 472
13.3.2 Operation in Asynchronous Mode ..... 474
13.3.3 Multiprocessor Communication ..... 483
13.3.4 Clocked Synchronous Operation ..... 490
13.4 SCI Interrupt Sources and the DMAC ..... 500
13.5 Usage Notes ..... 500
13.5.1 TDR Write and TDRE Flag ..... 500
13.5.2 Simultaneous Multiple Receive Errors ..... 501
13.5.3 Break Detection and Processing ..... 501
13.5.4 Receive Error Flags and Transmitter Operation (Clocked Synchronous Mode Only) ..... 501
13.5.5 Receive Data Sampling Timing and Receive Margin in Asynchronous Mode ..... 501
13.5.6 Cautions for Clocked Synchronous External Clock Mode ..... 503
13.5.7 Caution for Clocked Synchronous Internal Clock Mode ..... 503

Section 14 Smart Card Interface ..... 505
14.1 Overview ..... 505
14.1.1 Features ..... 505
14.1.2 Block Diagram ..... 506
14.1.3 Pin Configuration ..... 507
14.1.4 Register Configuration ..... 507
14.2 Register Descriptions ..... 508
14.2.1 Smart Card Mode Register (SCMR) ..... 508
14.2.2 Serial Mode Register (SMR) ..... 509
14.2.3 Serial Control Register (SCR) ..... 510
14.2.4 Serial Status Register (SSR) ..... 510
14.3 Operation ..... 512
14.3.1 Overview ..... 512
14.3.2 Pin Connections ..... 513
14.3.3 Data Format ..... 514
14.3.4 Register Settings ..... 515
14.3.5 Clock ..... 517
14.3.6 Data Transfer Operations ..... 520
14.4 Usage Notes ..... 526
14.4.1 Receive Data Timing and Receive Margin in Asynchronous Mode ..... 526
14.4.2 Retransfer (Receive and Transmit Modes) ..... 528

Section 15 Serial Communication Interface with FIFO (SCIF) ..... 531
15.1 Overview ..... 531
15.1.1 Features ..... 531
15.1.2 Block Diagram ..... 532
15.1.3 Pin Configuration ..... 533
15.1.4 Register Configuration ..... 533

15.2 Register Descriptions ... 534
15.2.1 Receive Shift Register (SCRSR) ... 534
15.2.2 Receive FIFO Data Register (SCFRDR) ... 535
15.2.3 Transmit Shift Register (SCTSR) ... 535
15.2.4 Transmit FIFO Data Register (SCFTDR) ... 535
15.2.5 Serial Mode Register (SCSMR) ... 536
15.2.6 Serial Control Register (SCSCR) ... 538
15.2.7 Serial Status Register (SCSSR) ... 540
15.2.8 Bit Rate Register (SCBRR) ... 544
15.2.9 FIFO Control Register (SCFCR) ... 552
15.2.10 FIFO Data Count Register (SCFDR) ... 554
15.3 Operation ... 554
15.3.1 Overview ... 554
15.3.2 Serial Operation ... 556
15.4 SCIF Interrupts ... 566
15.5 Usage Notes ... 566
15.5.1 SCFTDR Writing and the TDFE Flag ... 566
15.5.2 SCFRDR Reading and the RDF Flag ... 567
15.5.3 Break Detection and Processing ... 567
15.5.4 Sending a Break Signal ... 567
15.5.5 TEND Flag and TE Bit Processing ... 568
15.5.6 Receive Data Sampling Timing and Receive Margin ... 568

Section 16 IrDA ... 571
16.1 Overview ... 571
16.1.1 Features ... 571
16.1.2 Block Diagram ... 572
16.1.3 Pin Configuration ... 572
16.1.4 Register Configuration ... 573
16.2 Register Description ... 574
16.2.1 Serial Mode Register (SCSMR) ... 574
16.3 Operation ... 576
16.3.1 Overview ... 576
16.3.2 Transmission ... 576
16.3.3 Reception ... 576

Section 17 Serial I/O (SIO) ... 579
17.1 Overview ... 579
17.1.1 Features ... 579
17.2 Register Configuration ... 582
17.2.1 Receive Shift Register (SIRSR) ... 583
17.2.2 Receive Data Register (SIRDR) ... 583
17.2.3 Transmit Shift Register (SITSR) ... 584

17.2.4 Transmit Data Register (SITDR) ... 584
17.2.5 Serial Control Register (SICTR) ... 585
17.2.6 Serial Status Register (SISTR) ... 587
17.3 Operation ... 589
17.3.1 Input ... 589
17.3.2 Output ... 590
17.4 SIO Interrupt Sources and DMAC ... 592

Section 18 16-Bit Timer Pulse Unit (TPU) ... 595
18.1 Overview ... 595
18.1.1 Features ... 595
18.1.2 Block Diagram ... 598
18.1.3 Pin Configuration ... 599
18.1.4 Register Configuration ... 600
18.2 Register Descriptions ... 601
18.2.1 Timer Control Register (TCR) ... 601
18.2.2 Timer Mode Register (TMDR) ... 605
18.2.3 Timer I/O Control Register (TIOR) ... 607
18.2.4 Timer Interrupt Enable Register (TIER) ... 614
18.2.5 Timer Status Register (TSR) ... 616
18.2.6 Timer Counter (TCNT) ... 619
18.2.7 Timer General Register (TGR) ... 620
18.2.8 Timer Start Register (TSTR) ... 620
18.2.9 Timer Synchro Register (TSYR) ... 621
18.3 Interface to Bus Master ... 622
18.3.1 16-Bit Registers ... 622
18.3.2 8-Bit Registers ... 622
18.4 Operation ... 624
18.4.1 Overview ... 624
18.4.2 Basic Functions ... 625
18.4.3 Synchronous Operation ... 631
18.4.4 Buffer Operation ... 633
18.4.5 PWM Modes ... 637
18.4.6 Phase Counting Mode ... 642
18.5 Interrupts ... 647
18.5.1 Interrupt Sources and Priorities ... 647
18.5.2 DMAC Activation ... 648
18.6 Operation Timing ... 649
18.6.1 Input/Output Timing ... 649
18.6.2 Interrupt Signal Timing ... 653
18.7 Usage Notes ... 656
18.7.1 Input Clock Restrictions ... 656
18.7.2 Caution on Period Setting ... 657

18.7.3 Contention between TCNT Write and Clear Operations ..... 658
18.7.4 Contention between TCNT Write and Increment Operations ..... 659
18.7.5 Contention between TGR Write and Compare Match ..... 660
18.7.6 Contention between Buffer Register Write and Compare Match ..... 661
18.7.7 Contention between TGR Read and Input Capture ..... 662
18.7.8 Contention between TGR Write and Input Capture ..... 663
18.7.9 Contention between Buffer Register Write and Input Capture ..... 664
18.7.10 Contention between Overflow/Underflow and Counter Clearing ..... 665
18.7.11 Contention between TCNT Write and Overflow/Underflow ..... 666
18.7.12 Multiplexing of I/O Pins ..... 666
18.7.13 Interrupts and Module Stop Mode ..... 666

Section 19 Hitachi User Debug Interface (H-UDI) ..... 667
19.1 Overview ..... 667
19.1.1 Features ..... 667
19.1.2 H-UDI Block Diagram ..... 668
19.1.3 Pin Configuration ..... 669
19.1.4 Register Configuration ..... 669
19.2 External Signals ..... 670
19.2.1 Test Clock (TCK) ..... 670
19.2.2 Test Mode Select (TMS) ..... 670
19.2.3 Test Data Input (TDI) ..... 670
19.2.4 Test Data Output (TDO) ..... 670
19.2.5 Test Reset ($\overline{\text{TRST}}$) ..... 670
19.3 Register Descriptions ..... 671
19.3.1 Instruction Register (SDIR) ..... 671
19.3.2 Status Register (SDSR) ..... 673
19.3.3 Data Register (SDDR) ..... 674
19.3.4 Bypass Register (SDBPR) ..... 674
19.4 Operation ..... 675
19.4.1 H-UDI Interrupt and Serial Transfer ..... 675
19.4.2 Bypass Mode ..... 678
19.4.3 H-UDI Reset ..... 678
19.5 Usage Notes ..... 678

Section 20 Pin Function Controller (PFC) ..... 681
20.1 Overview ..... 681
20.2 Register Configuration ..... 683
20.3 Register Descriptions ..... 683
20.3.1 Port A Control Register (PACR) ..... 683
20.3.2 Port A I/O Register (PAIOR) ..... 686
20.3.3 Port B Control Registers (PBCR, PBCR2) ..... 687
20.3.4 Port B I/O Register (PBIOR) ..... 693

Section 21 I/O Ports ..... 695
21.1 Overview ..... 695
21.2 Port A ..... 695
21.2.1 Register Configuration ..... 695
21.2.2 Port A Data Register (PADR) ..... 696
21.3 Port B ..... 697
21.3.1 Register Configuration ..... 697
21.3.2 Port B Data Register (PBDR) ..... 698

Section 22 Power-Down Modes ..... 699
22.1 Overview ..... 699
22.1.1 Power-Down Modes ..... 699
22.1.2 Register ..... 700
22.2 Register Descriptions ..... 701
22.2.1 Standby Control Register 1 (SBYCR1) ..... 701
22.2.2 Standby Control Register 2 (SBYCR2) ..... 703
22.3 Sleep Mode ..... 705
22.3.1 Transition to Sleep Mode ..... 705
22.3.2 Canceling Sleep Mode ..... 705
22.4 Standby Mode ..... 706
22.4.1 Transition to Standby Mode ..... 706
22.4.2 Canceling Standby Mode ..... 708
22.4.3 Standby Mode Cancellation by NMI Interrupt ..... 708
22.4.4 Clock Pause Function ..... 709
22.4.5 Notes on Standby Mode ..... 712
22.5 Module Standby Function ..... 712
22.5.1 Transition to Module Standby Function ..... 712
22.5.2 Clearing the Module Standby Function ..... 712

Section 23 Electrical Characteristics ..... 713
23.1 Absolute Maximum Ratings ..... 713
23.2 DC Characteristics ..... 714
23.3 AC Characteristics ..... 716
23.3.1 Clock Timing ..... 717
23.3.2 Control Signal Timing ..... 721
23.3.3 Bus Timing ..... 723
23.3.4 Direct Memory Access Controller Timing ..... 766
23.3.5 Free-Running Timer Timing ..... 767
23.3.6 Serial Communication Interface Timing ..... 769
23.3.7 Watchdog Timer Timing ..... 772
23.3.8 Serial I/O Timing ..... 773
23.3.9 Hitachi User Debug Interface Timing ..... 775
23.3.10 I/O Port Timing ..... 777

23.4 AC Characteristic Test Conditions ..... 778

Appendix A On-Chip Peripheral Module Registers ..... 779
A.1 Addresses ..... 779

Appendix B Pin States ..... 793
B.1 Pin States in Reset, Power-Down State, and Bus-Released State ..... 793

Appendix C Product Lineup ..... 796
C.1 Product Lineup ..... 796

Appendix D Package Dimensions ..... 797

# Section 1 Overview

## 1.1 Features

The Hitachi SH7612 processor is a single-chip device that combines the functionality of a full-fledged reduced instruction set computer (RISC) processor and a full-fledged digital signal processing (DSP) processor. It is ideally suited for applications that require both microcontroller and DSP type processing. Traditionally, such applications have required the use of two chips: a microcontroller and a DSP chip. This processor offers an economical, one-chip solution for such applications.

The single-chip solution provided by this processor cuts development and system costs. The unified memory and single instruction stream simplify the design, enabling development costs to be reduced. The provision of RISC and DSP functions in a single processor enables development to be carried out with a single emulator, and the use of a single processor also eliminates the need for inter-processor communication support. Power consumption can also be reduced through use of the power-down states.

This processor is upwardly compatible with the Hitachi SH-1 and SH-2 object code, but offers an extension of the basic DSP functions of the SH-1 and SH-2, with full DSP type functions. While the SH-2 processors provide multiply-accumulate functionality, this processor provides advanced DSP operations, including single-cycle 16-bit by 16-bit signed multiplication, barrel shifting, priority encoding, rounding, modulo addressing, and zero-overhead loop control. The processor performs DSP operations in fixed-point arithmetic, providing guard bits to protect against data corruption due to overflows.

The processor uses a RISC architecture like that of the SH-2 family of processors. By using a five-stage pipeline system, it can execute most instructions in a single processor clock cycle.

This processor uses both a standard von Neumann architecture (both instructions and data share a single bus) for the integer unit and an extended Harvard architecture (instructions and data have separate buses) for the DSP unit. The Harvard architecture allows simultaneous access to the program code and to two on-chip memory spaces, called X memory and Y memory, for the efficient implementation of DSP algorithms.

This manual describes all of the hardware features of the processor, with emphasis on interfacing to the processor through the many on-chip peripheral modules.

The features of this processor include:

- Architecture
  - — Original Hitachi architecture
  - — 32/40-bit internal data bus

— von Neumann architecture for integer instructions and extended Harvard architecture for DSP instructions

- Registers
  - — Sixteen 32-bit general registers (R0–R15)
  - — Eight DSP registers: six 32-bits wide (X0, X1, Y0, Y1, M0, M1) and two 40-bits wide (A0, A1)
  - — Four 32-bit system registers (MACH, MACL, PR, PC)
  - — Seven 32-bit control registers: three for 16-bit integer instructions (SR, GBR, VBR) and four for 32-bit DSP instructions (RS, RE, MOD, DSR)
- Address space
  - — 4 Gbytes address space (160 Mbytes of external memory space)
  - — On-chip memory: 16 kbytes of RAM
  - — Modulo addressing
  - — Indirect addressing with pointer update (increment, decrement, indexed)
  - — Dual-addressing capability to support two memory accesses per clock cycle
- Five-stage pipeline
- Instruction set
  - — Single instruction stream for both 16-bit and 32-bit instructions
  - — Parallel instructions: 32-bit length allows up to four operations to be executed simultaneously
  - — Load-store architecture (basic arithmetic and logic operations are performed on register values)
  - — Delay branches to reduce pipeline disruption (delays branching to minimize instances when instructions fetched to the pipeline must be discarded because of a branching instruction)
  - — Conditional DSP instruction execution
  - — Optimized for C programming language implementation
  - — On-chip debugging functions
- Instruction execution time
  - — One instruction/cycle for basic instructions
  - — Zero-overhead loop control
- Arithmetic operations
  - — Two ALUs
  - — Two shifters
  - — Arithmetic and logical barrel shifting
  - — Most-significant bit detection (priority encoding)
  - — Guard bits to protect against overflow
  - — Saturation arithmetic

- On-chip multipliers
  - DSP multiplication operations executed in one cycle (16 bits × 16 bits → 32 bits)
  - Integer multiplication operations execute in one to three cycles (16 bits × 16 bits → 32 bits) or two to four cycles (32 bits × 32 bits → 64 bits), and multiply-accumulate operations execute in two to three cycles (16 bits × 16 bits + 64 bits → 64 bits) or two to four cycles (32 bits × 32 bits + 64 bits → 64 bits)
- On-chip peripheral modules
  - System controller (SYSC)
  - Interrupt controller (INTC)
  - User break controller (UBC)
  - Bus state controller (BSC)
  - Cache memory
  - DMA controller (DMAC) with two channels
  - Division unit (DIVU)
  - 16-bit free-running timer (FRT) with one channel
  - Watchdog timer (WDT)
  - Serial communication interface (SCI) with one channel
  - Single-channel smart card interface
  - Two-channel serial communication interface with FIFO(SCIF)
  - IrDA
  - Three-channel serial I/O (SIO)
  - Three-channel 16-bit timer pulse unit (TPU)
  - Hitachi user debug interface (H-UDI)
- Processing states
  - Reset state (power-on reset and manual reset)
  - Exception processing state
  - Program execution state
  - Power-down states: sleep mode, standby mode, and module standby mode
  - Bus release state

## 1.2 Architecture

Figure 1.1 shows a chip-level block diagram of the processor. The processor consists of a CPU, peripheral modules, and on-chip memory linked by five different buses.

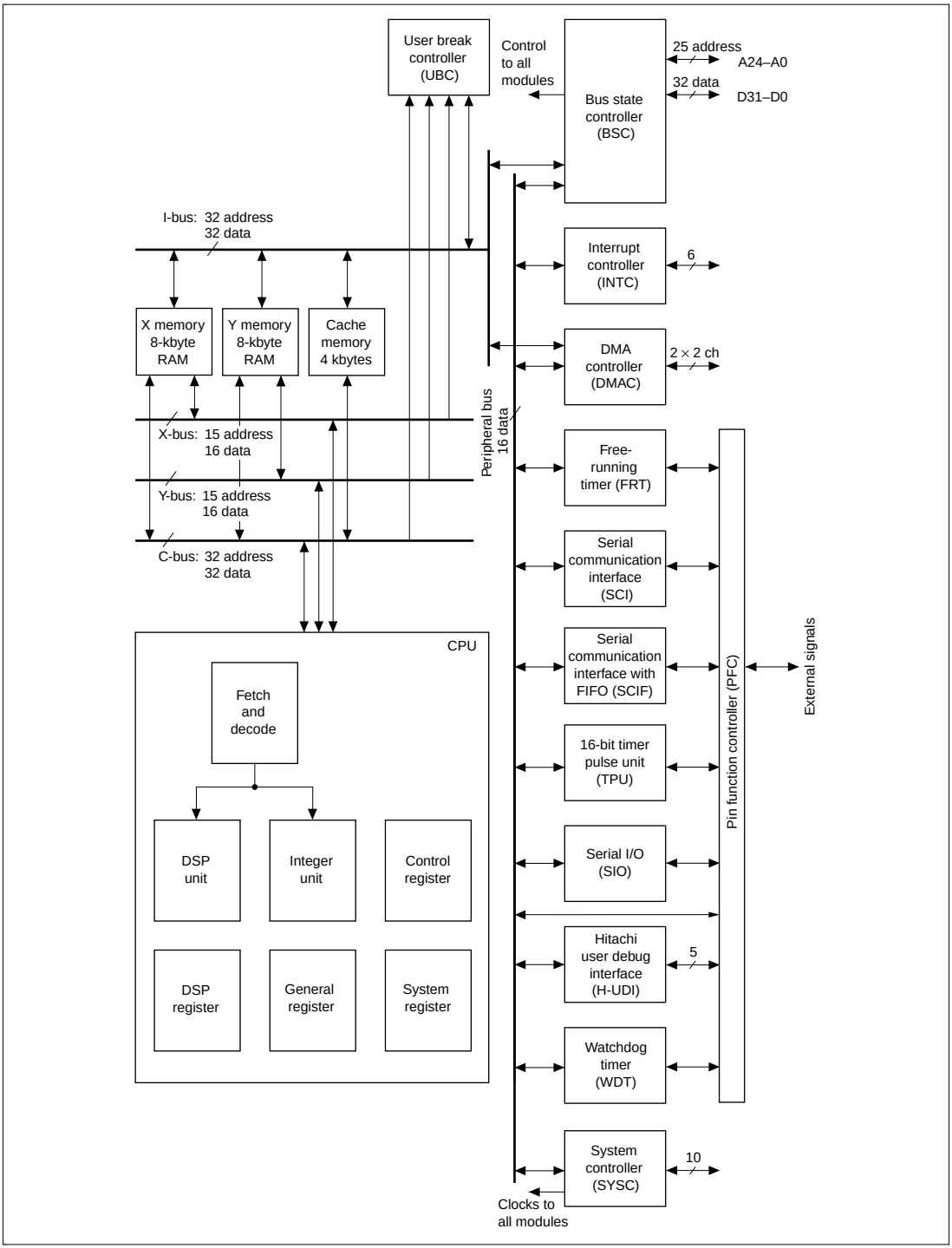



**Figure 1.1 Block Diagram**

The CPU contains a fetch and decode unit, an integer unit, a DSP unit, and associated registers (general, DSP, control, and system). The fetch and decode unit reads instructions and controls both the integer and DSP units. The integer unit has the capabilities of an SH-2 CPU, with the addition of some features to support DSP operations. The DSP unit performs advanced DSP functions and has its own set of registers and separately addressable memory spaces called the X and Y memories. The SH-DSP instruction set is a version of the SH-2 instruction set that has been enhanced to make use of these additional architectural features. The CPU interfaces to external logic through the peripheral modules, as shown on the right side of figure 1.1.

In addition to the external bus, the processor has five internal buses:

- Cache bus or C-bus (CAB, CDB): 32-bit address, 32-bit data
- Internal bus or I-bus (IAB, IDB): 32-bit address, 32-bit data
- X-bus (XAB, XDB): 15-bit address, 16-bit data
- Y-bus (YAB, YDB): 15-bit address, 16-bit data
- Peripheral bus: 16-bit data

The C-bus transfers both instructions and data. The processor can access any region of address space, including external space, through the C-bus. All MOV and MOVS operations use the C-bus. MOVX operations use the X-bus, and MOVY operations use the Y-bus. The X- and Y-data buses, XDB and YDB, are 16 bits wide, and can access only word data.

The SH-DSP integer unit uses the C-bus to transfer instructions and data (von Neumann architecture). The DSP unit also uses the C-bus to transfer instructions and data, but it can also use the two additional internal buses, called the X-bus and Y-bus, to access data from the X memory and Y memory simultaneously (extended-Harvard architecture).

The I-bus can be used to address any memory space, including external memory and on-chip memory. When accessing external memory, the I-bus connects to the external bus through the BSC.

The DSP unit has the X-bus and Y-bus available for data transfer operations. Each of these buses consists of a 15-bit address bus, XAB or YAB, and a 16-bit data bus, XDB or YDB. The address buses are only 15 bits wide because they can access only aligned word-length data, so the least-significant bit of the 16-bit address is always zero. The CPU uses the X-bus and Y-bus to access X memory and Y memory data.

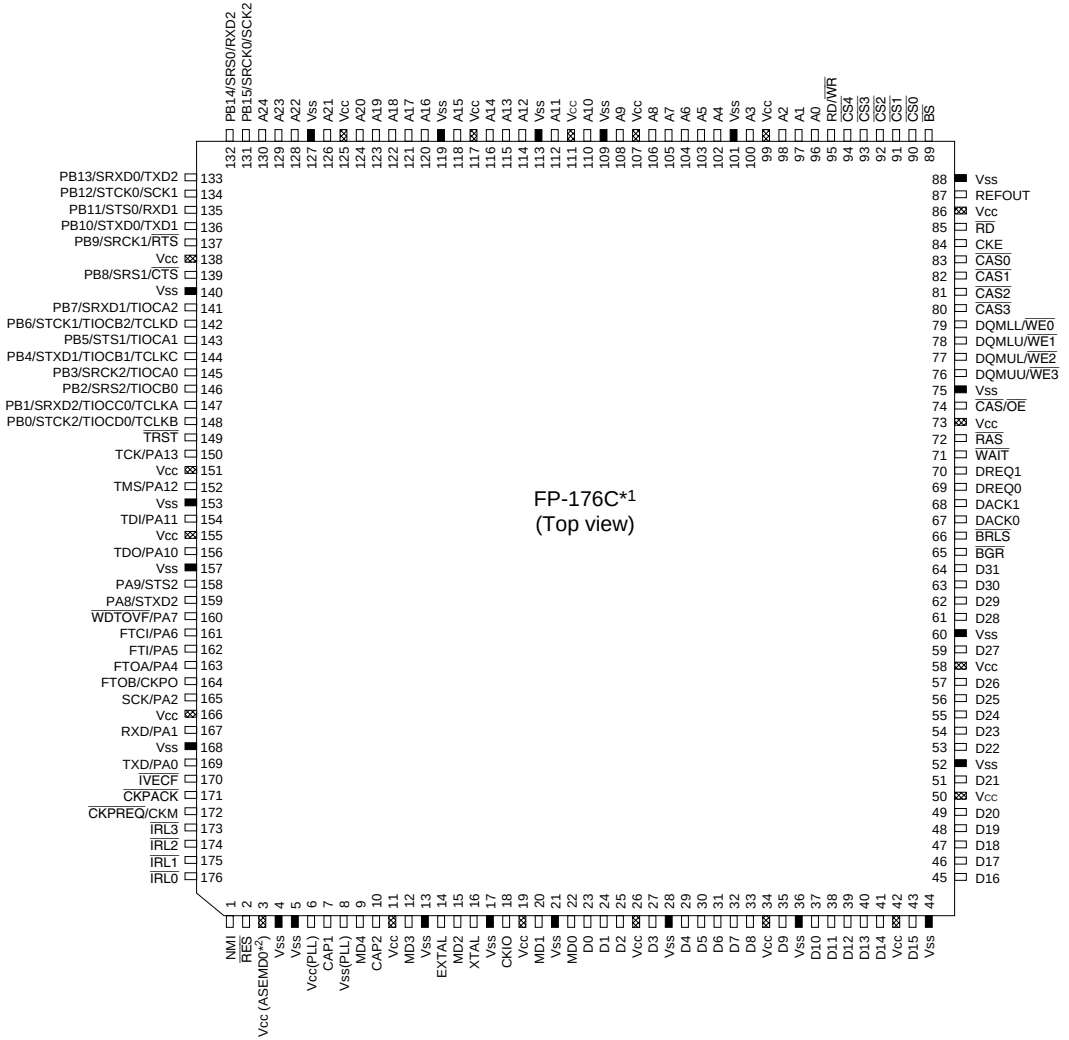
The CPU uses the C-bus to accesses cache memory and on-chip memory. Other spaces are accessed from the C-bus via the I-bus. The DMAC can use the I-bus to access spaces other than cache memory.

A peripheral bus is also provided for accessing peripheral modules through the BSC. This 16-bit data bus allows the CPU and DMAC to access the memory-mapped registers in the peripheral modules.

The UBC monitors the C-bus, I-bus, X-bus, and Y-bus for break conditions. When a break condition is met, an interrupt is requested.

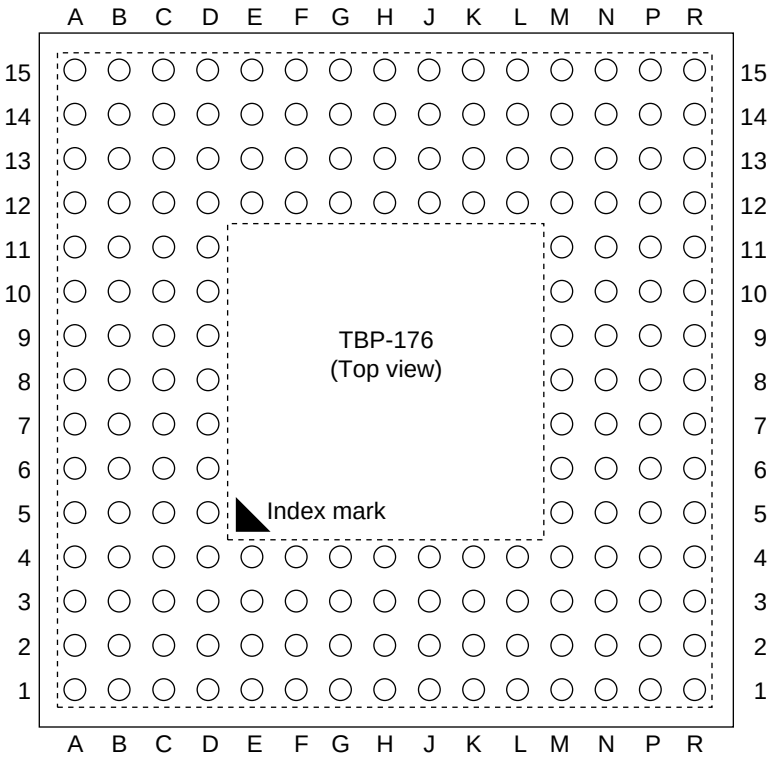
## 1.3 Pin Arrangement

Figures 1.2 and 1.3 show the pin arrangement. Table 1.1 lists the pins.




Notes: 1. Use of the E10 emulator for the SH7612 requires a dedicated emulator chip (debug chip) with the same package as the actual chip. A debug chip can only be used with the FP-176C.
2. Connect to $V_{SS}$ when using the E10 emulator, and to $V_{CC}$ when using the actual chip.

**Figure 1.2 Pin Arrangement (FP-176C)**




Notes: 1. The pin diagram enclosed in dotted lines is a cutaway view.
2. An E10 emulator debug chip is not available in a TBP-176 package.

**Figure 1.3 Pin Arrangement (TBP-176)**

## 1.4 Pins

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 1 | A1 | NMI | I | Nonmaskable interrupt request |
| 2 | C3 | $\overline{\text{RES}}$ | I | Reset |
| 3 | B1 | Vcc (ASEMD0*) | I | Power supply (mode pin in debug chip) |
| 4 | C2 | Vss | I | Ground |
| 5 | D3 | Vss | I | Ground |
| 6 | C1 | Vcc(PLL) | I | On-chip PLL power supply |
| 7 | D2 | CAP1 | O | PLL external capacitance pin |
| 8 | E4 | Vss(PLL) | I | On-chip PLL ground |
| 9 | D1 | MD4 | I | Operating mode pin |
| 10 | E3 | CAP2 | O | PLL external capacitance pin |
| 11 | E2 | Vcc | I | Power supply |
| 12 | E1 | MD3 | I | Operating mode pin |
| 13 | F4 | Vss | I | Ground |
| 14 | F3 | EXTAL | I | Crystal oscillator connection pin |
| 15 | F1 | MD2 | I | Operating mode pin |
| 16 | F2 | XTAL | O | Crystal oscillator connection pin |
| 17 | G4 | Vss | I | Ground |
| 18 | G3 | CKIO | IO | System clock input/output |
| 19 | G1 | Vcc | I | Power supply |
| 20 | G2 | MD1 | I | Operating mode pin |
| 21 | H4 | Vss | I | Ground |
| 22 | H3 | MD0 | I | Operating mode pin |
| 23 | H1 | D0 | IO | Data bus |
| 24 | H2 | D1 | IO | Data bus |
| 25 | J4 | D2 | IO | Data bus |
| 26 | J3 | Vcc | I | Power supply |
| 27 | J1 | D3 | IO | Data bus |
| 28 | J2 | Vss | I | Ground |
| 29 | K4 | D4 | IO | Data bus |
| 30 | K3 | D5 | IO | Data bus |
| 31 | K1 | D6 | IO | Data bus |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 32 | K2 | D7 | IO | Data bus |
| 33 | L3 | D8 | IO | Data bus |
| 34 | L1 | Vcc | I | Power supply |
| 35 | L2 | D9 | IO | Data bus |
| 36 | L4 | Vss | I | Ground |
| 37 | M1 | D10 | IO | Data bus |
| 38 | M2 | D11 | IO | Data bus |
| 39 | M3 | D12 | IO | Data bus |
| 40 | N1 | D13 | IO | Data bus |
| 41 | M4 | D14 | IO | Data bus |
| 42 | N2 | Vcc | I | Power supply |
| 43 | P1 | D15 | IO | Data bus |
| 44 | P2 | Vss | I | Ground |
| 45 | R1 | D16 | IO | Data bus |
| 46 | N3 | D17 | IO | Data bus |
| 47 | R2 | D18 | IO | Data bus |
| 48 | P3 | D19 | IO | Data bus |
| 49 | N4 | D20 | IO | Data bus |
| 50 | R3 | Vcc | I | Power supply |
| 51 | P4 | D21 | IO | Data bus |
| 52 | M5 | Vss | I | Ground |
| 53 | R4 | D22 | IO | Data bus |
| 54 | N5 | D23 | IO | Data bus |
| 55 | P5 | D24 | IO | Data bus |
| 56 | R5 | D25 | IO | Data bus |
| 57 | M6 | D26 | IO | Data bus |
| 58 | N6 | Vcc | I | Power supply |
| 59 | R6 | D27 | IO | Data bus |
| 60 | P6 | Vss | I | Ground |
| 61 | M7 | D28 | IO | Data bus |
| 62 | N7 | D29 | IO | Data bus |
| 63 | R7 | D30 | IO | Data bus |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 64 | P7 | D31 | IO | Data bus |
| 65 | M8 | $\overline{BGR}$ | O | Bus grant |
| 66 | N8 | $\overline{BRLS}$ | I | Bus request |
| 67 | R8 | DACK0 | O | DMAC channel 0 acknowledge |
| 68 | P8 | DACK1 | O | DMAC channel 1 acknowledge |
| 69 | M9 | DREQ0 | I | DMAC channel 0 request |
| 70 | N9 | DREQ1 | I | DMAC channel 1 request |
| 71 | R9 | $\overline{WAIT}$ | I | Hardware wait request |
| 72 | P9 | $\overline{RAS}$ | O | RAS for DRAM and synchronous DRAM |
| 73 | M10 | Vcc | I | Power supply |
| 74 | N10 | $\overline{CAS}$/$\overline{OE}$ | O | CAS for synchronous DRAM/OE for DRAM (EDO mode) |
| 75 | R10 | Vss | I | Ground |
| 76 | P10 | DQMUU/$\overline{WE3}$ | O | SRAM/synchronous DRAM highest byte select signal |
| 77 | N11 | DQMUL/$\overline{WE2}$ | O | SRAM/synchronous DRAM second byte select signal |
| 78 | R11 | DQMLU/$\overline{WE1}$ | O | SRAM/synchronous DRAM third byte select signal |
| 79 | P11 | DQMLL/$\overline{WE0}$ | O | SRAM/synchronous DRAM lowest byte select signal |
| 80 | M11 | $\overline{CAS3}$ | O | DRAM highest byte select signal |
| 81 | R12 | $\overline{CAS2}$ | O | DRAM second byte select signal |
| 82 | P12 | $\overline{CAS1}$ | O | DRAM third byte select signal |
| 83 | N12 | $\overline{CAS0}$ | O | DRAM lowest byte select signal |
| 84 | R13 | CKE | O | Synchronous DRAM clock enable control |
| 85 | M12 | $\overline{RD}$ | O | Read pulse |
| 86 | P13 | Vcc | I | Power supply |
| 87 | R14 | REFOUT | O | External refresh request signal |
| 88 | P14 | Vss | I | Ground |
| 89 | R15 | $\overline{BS}$ | O | Bus cycle start |
| 90 | N13 | $\overline{CS0}$ | O | Chip select 0 |
| 91 | P15 | $\overline{CS1}$ | O | Chip select 1 |
| 92 | N14 | $\overline{CS2}$ | O | Chip select 2 |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 93 | M13 | $\overline{CS3}$ | O | Chip select 3 |
| 94 | N15 | $\overline{CS4}$ | O | Chip select 4 |
| 95 | M14 | RD/$\overline{WR}$ | O | Read/write |
| 96 | L12 | A0 | O | Address bus |
| 97 | M15 | A1 | O | Address bus |
| 98 | L13 | A2 | O | Address bus |
| 99 | L14 | Vcc | I | Power supply |
| 100 | L15 | A3 | O | Address bus |
| 101 | K12 | Vss | I | Ground |
| 102 | K13 | A4 | O | Address bus |
| 103 | K15 | A5 | O | Address bus |
| 104 | K14 | A6 | O | Address bus |
| 105 | J12 | A7 | O | Address bus |
| 106 | J13 | A8 | O | Address bus |
| 107 | J15 | Vcc | I | Power supply |
| 108 | J14 | A9 | O | Address bus |
| 109 | H12 | Vss | I | Ground |
| 110 | H13 | A10 | O | Address bus |
| 111 | H15 | Vcc | I | Power supply |
| 112 | H14 | A11 | O | Address bus |
| 113 | G12 | Vss | I | Ground |
| 114 | G13 | A12 | O | Address bus |
| 115 | G15 | A13 | O | Address bus |
| 116 | G14 | A14 | O | Address bus |
| 117 | F12 | Vcc | I | Power supply |
| 118 | F13 | A15 | O | Address bus |
| 119 | F15 | Vss | I | Ground |
| 120 | F14 | A16 | O | Address bus |
| 121 | E13 | A17 | O | Address bus |
| 122 | E15 | A18 | O | Address bus |
| 123 | E14 | A19 | O | Address bus |
| 124 | E12 | A20 | O | Address bus |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 125 | D15 | Vcc | I | Power supply |
| 126 | D14 | A21 | O | Address bus |
| 127 | D13 | Vss | I | Ground |
| 128 | C15 | A22 | O | Address bus |
| 129 | D12 | A23 | O | Address bus |
| 130 | C14 | A24 | O | Address bus |
| 131 | B15 | PB15/SRCK0/ SCK2 | IO/I/IO | General input/output / SIO channel 0 serial receive clock input / SCIF channel 2 serial clock input/output |
| 132 | B14 | PB14/SRS0/ RXD2 | IO/I/I | General input/output / SIO channel 0 serial receive synchronization input / SCIF channel 2 serial data input |
| 133 | A15 | PB13/SRXD0/ TXD2 | IO/I/O | General input/output / SIO channel 0 serial data input / SCIF channel 2 serial data output |
| 134 | C13 | PB12/STCK0/ SCK1 | IO/I/IO | General input/output / SIO channel 0 serial transmit clock input / SCIF channel 1 serial clock input/output |
| 135 | A14 | PB11/STS0/ RXD1 | IO/IO/I | General input/output / SIO channel 0 serial transmit synchronization input/output / SCIF channel 1 serial data input |
| 136 | B13 | PB10/STXD0/ TXD1 | IO/O/O | General input/output / SIO channel 0 serial data output / SCIF channel 1 serial data output |
| 137 | C12 | PB9/SRCK1/ $\overline{\text{RTS}}$ | IO/I/O | General input/output / SIO channel 1 serial receive clock input / SCIF channel 1 transmit request |
| 138 | A13 | Vcc | I | Power supply |
| 139 | B12 | PB8/SRS1/ $\overline{\text{CTS}}$ | IO/I/I | General input/output / SIO channel 1 serial receive synchronization input / SCIF channel 1 transmit enable |
| 140 | D11 | Vss | I | Ground |
| 141 | A12 | PB7/SRXD1/ TIOCA2 | IO/I/O | General input/output / SIO channel 1 serial data input / TPU channel 2 input capture/out compare match A2 |
| 142 | C11 | PB6/STCK1/ TIOCB2/ TCLKD | IO/I/IO | General input/output / SIO channel 1 serial data transmit clock input / TPU channel 2 input capture/out compare match B2 / TPU clock input D |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 143 | B11 | PB5/STS1/ TIOCA1 | IO/IO/IO | General input/output / SIO channel 1 serial transmit synchronization input/output / TPU channel 1 input capture/out compare match A1 |
| 144 | A11 | PB4/STXD1/ TIOCB1/TCLKC | IO/O/IO | General input/output / SIO channel 1 serial data output / TPU channel 1 input capture/out compare match B1 / TPU clock input C |
| 145 | D10 | PB3/SRCK2/ TIOCA0 | IO/I/IO | General input/output / SIO channel 2 serial receive clock input / TPU channel 0 input capture/out compare match A0 |
| 146 | C10 | PB2/SRS2/ TIOCB0 | IO/I/IO | General input/output / SIO channel 2 serial receive synchronization input / TPU channel 0 input capture/out compare match B0 |
| 147 | A10 | PB1/SRXD2/ TIOCC0/TCLKA | IO/I/IO | General input/output / SIO channel 2 serial data input / TPU channel 0 input capture/out compare match C0 / TPU clock input A |
| 148 | B10 | PB0/STCK2/ TIOCD0/TCLKB | IO/I/IO | General input/output / SIO channel 2 serial data transmit clock input / TPU channel 0 input capture/out compare match D0 / TPU clock input B |
| 149 | D9 | $\overline{\text{TRST}}$ | I | H-UDI test reset input |
| 150 | C9 | TCK/PA13 | I/IO | H-UDI test clock input / general input/output |
| 151 | A9 | Vcc | I | Power supply |
| 152 | B9 | TMS/PA12 | I/IO | H-UDI test mode select / general input/output |
| 153 | D8 | Vss | I | Ground |
| 154 | C8 | TDI/PA11 | I/IO | H-UDI serial data input / general input/output |
| 155 | A8 | Vcc | I | Power supply |
| 156 | B8 | TDO/PA10 | O/IO | H-UDI serial data output / general input/output |
| 157 | D7 | Vss | I | Ground |
| 158 | C7 | PA9/STS2 | IO/IO | General input/output / SIO channel 2 serial transmit synchronization input/output |
| 159 | A7 | PA8/STXD2 | IO/O | General input/output / SIO channel 2 serial data output |
| 160 | B7 | $\overline{\text{WDTOVF}}$/PA7 | O/IO | Watchdog timer output / general input/output |
| 161 | D6 | FTCI/PA6 | I/IO | FRT clock input / general input/output |
| 162 | C6 | FTI/PA5 | I/IO | FRT input capture input / general input/output |
| 163 | A6 | FTOA/PA4 | O/IO | FRT output compare A output / general input/output |

| Pin No. (FP-176C) | Pin No. (TBP-176) | Pin Name | I/O | Function |
|---|---|---|---|---|
| 164 | B6 | FTOB/CKPO | O/O | FRT output compare B output / Pϕ |
| 165 | C5 | SCK/PA2 | IO/IO | SCI serial clock input/output / general input/output |
| 166 | A5 | Vcc | I | Power supply |
| 167 | B5 | RXD/PA1 | I/IO | SCI serial data input / general input/output |
| 168 | D5 | Vss | I | Ground |
| 169 | A4 | TXD/PA0 | O/IO | SCI serial data output / general input/output |
| 170 | B4 | $\overline{\text{IVECF}}$ | O | Interrupt vector fetch cycle |
| 171 | C4 | $\overline{\text{CKPACK}}$ | O | Clock pause acknowledge output |
| 172 | A3 | $\overline{\text{CKPREQ}}$/CKM | I | Clock pause request input |
| 173 | D4 | $\overline{\text{IRL3}}$ | I | External interrupt source input |
| 174 | B3 | $\overline{\text{IRL2}}$ | I | External interrupt source input |
| 175 | A2 | $\overline{\text{IRL1}}$ | I | External interrupt source input |
| 176 | B2 | $\overline{\text{IRL0}}$ | I | External interrupt source input |

Note: When performing debugging with an E10 emulator (series type E10), mode switching is carried out with this pin. Connect this pin to $V_{SS}$ when using the E10 emulator, and to $V_{CC}$ when using the actual chip.

## 1.5 CPU

### 1.5.1 Fetch and Decode

This chip supports a mixed 16-bit/32-bit instruction stream. There are no restrictions on the order or sequence of instructions in the mixed 16-bit/32-bit instruction stream.

### 1.5.2 Integer Unit

The integer unit is a version of the SH-2 CPU core that has been enhanced to support DSP operations. It can execute all SH-1 and SH-2 object code, but it is not upwardly compatible with SH-3 object code. Compared with the SH-2 CPU core, the SH-DSP integer unit offers the following additional features:

**Dual-Addressing Capability:** The chip supports the simultaneous access of data from two on-chip memory locations by using the main integer unit ALU to calculate the X memory address and a separate 16-bit ALU called the Pointer Arithmetic Unit (PAU) to calculate the Y memory address.

**Index Addressing with Pointer Update:** The addressing mechanism supports index addressing with automatic updating of the address pointer. The address pointer is automatically incremented or decremented by 2 or 4 to access sequential words or longwords in memory, or incremented by a specified index amount after each memory access.

**Modulo Addressing:** Modulo addressing is useful for implementing circular buffers. The starting and ending modulo addresses are specified in the MOD control register. When the address register is incremented to the ending address, it is automatically reset to the starting address.

**Zero-Overhead Loop Control:** The chip supports zero-overhead program loops, in which loop counter incrementing and judgment of completion of the loop are performed automatically. These loops are important for high-speed DSP applications. To set up such a loop, special registers are used to specify the repeat count, the starting address, and the ending address of the instruction loop. The processor then executes the loop the specified number of times.

### 1.5.3 DSP Unit

The CPU has a powerful DSP unit that can execute up to two operations in parallel, providing faster DSP performance than conventional multiply-accumulate (MAC) units. The DSP unit offers the following features:

**Parallel Operations:** The DSP unit can execute up to two independent operations simultaneously: one addition or subtraction operation and one multiply operation. Simultaneously when the DSP unit executes these operations, the integer unit can execute up to two load or store operations concerning X memory or Y memory. A single 32-bit instruction specifies these two operations and two transfers.

**32/40 Bit Data Registers:** The DSP unit uses a set of eight data registers: two 40-bit registers and six 32-bit registers. The upper eight bits of the 40-bit registers serve as guard bits to accommodate overflow results in certain operations.

**Fixed-Point Arithmetic Operations:** Most of the 32-bit instructions executed by the DSP use fixed-point data with the binary point fixed between bits 30 and 31. All 16-bit instructions executed by the integer unit use integer data with the binary point fixed to the right of bit 0.

**Single-Cycle Multiply Operations:** The DSP unit executes a 16-bit by 16-bit multiply instruction in a single processor clock cycle. The chip can also execute all of the multiply operations supported by the SH-2 family with the same performance. See the *SH-1/SH-2/SH-DSP Programming Manual* for details.

**Barrel-Shift Operations:** These perform both arithmetic and logical barrel shifting of data in DSP registers, using either the contents of another register or an immediate value to specify the amount and direction of the shift.

**Conditional-Instruction Execution:** The instruction set allows conditional execution of some ALU and shift operations in the DSP unit. The conditions reflect such situations as negative data, data equal to zero, or the occurrence of a carry/borrow.

**Valid Most-Significant Bit Detection (Priority Encoding):** The chip provides an instruction to locate the most-significant bit of the data. The result of this operation can be combined with an arithmetic shift to normalize a value.

**Saturation Arithmetic:** The DSP fixed-point arithmetic and SH-2 multiply-accumulate operations can use saturation arithmetic to prevent overflows and underflows. Any result that exceeds the register width is set to the largest magnitude that the register can accommodate with the appropriate sign.

Section 2 describes the CPU.

## 1.6 Peripheral Module Units

The chip includes peripheral module units that are useful in many DSP applications:

- System controller (SYSC)
- Interrupt controller (INTC)
- User break interface (UBC)
- Bus state controller (BSC)
- Cache memory
- Direct memory access controller (DMAC): two channels
- Division unit (DIVU)
- 16-bit free-running timer (FRT): one channel
- Watchdog timer (WDT)
- Serial communication interface (SCI): one channel
- Smart card interface: one channel
- Serial communication interface with FIFO (SCIF): two channels
- IrDA
- Serial I/O (SIO): three channels
- 16-bit timer-pulse unit (TPU): three channels
- Hitachi user debug interface (H-UDI)
- Pin function controller (PFC)
- I/O ports

### 1.6.1 System Controller (SYSC)

The system controller generates and controls the clock signals for all on-chip modules and the external bus.

The SYSC functions in one of seven clock operating modes or in one of three power-down modes:

- Operating modes: Control the method of clock generation (PLL ON/OFF, etc.) and clock division ratio.
- Power-down modes: Sleep mode halts CPU function; standby mode halts all functions; module standby function selectively halts operation of the FRT, SCIF, SIO, DMAC, UBC, DSP, DIVU, TPU, and SCI.

### 1.6.2 Interrupt Controller (INTC)

The interrupt controller (INTC) determines the priority order of interrupt sources and controls interrupt requests to the CPU. The INTC includes the following features.

- Sixteen interrupt priority levels can be set

  By setting the interrupt priority registers, the priorities of on-chip peripheral module interrupts can be selected at 16 levels for different request sources.
- Vector numbers for on-chip peripheral module interrupt can be set

  By setting the vector number setting registers, the vector numbers of on-chip peripheral module interrupts can be set to values from 0 to 127 for different request sources.
- The IRL interrupt vector number setting can be selected: Either of two modes can be selected by a register setting: auto-vector mode in which vector numbers are determined internally, and external vector mode in which vector numbers are set externally.
- IRQ interrupt settings can be made (low level, rising-, falling-, or both-edge detection)

Section 5 describes the INTC.

### 1.6.3 User Break Controller (UBC)

The user break controller (UBC) provides functions that simplify user program debugging. The UBC includes the following features.

- The following can be set as break conditions:
  - — Number of break channels: Two (channels A and B)
  - — User break interrupts can be generated on independent conditions for channels A and B, or on sequential conditions (sequential break setting: channel A → channel B).
  - — Address: 32-bit maskable, individual address setting possible (cache bus (CPU), internal bus (DMAC), X/Y bus)
  - — Data (channel B only,): 32-bit maskable, individual address setting possible (cache bus (CPU), internal bus (DMAC), X/Y bus)
  - — Bus master: CPU cycle/DMA cycle
  - — Bus cycle: Instruction fetch/data access
  - — Read/write
  - — Operand cycle: Byte/word/longword
- User break interrupt generation on occurrence of break condition

  A user-written user break interrupt handling routine can be executed.
- When an instruction fetch cycle is set as a condition, it is possible to select interrupt generation before instruction execution or interrupt handling generation after instruction execution
- Multiple-execution specification break (channel B only)

  Settable number of executions: Max. $2^{12} - 1$ (4095)

- PC trace functions

  The branch source/branch destination can be traced when a branch instruction is executed (max. 4 pairs, 8 addresses)

Section 6 describes the UBC.

### 1.6.4 Bus State Controller (BSC)

The bus state controller (BSC) manages the address spaces and outputs control signals to allow optimum memory accesses to five spaces. The BSC includes the following features.

- Address space is managed as five spaces
  - — Maximum linear 32 Mbytes for each of spaces CS0 to CS4
  - — Memory type (DRAM, synchronous DRAM, burst ROM, etc.) can be specified for each space.
  - — Bus width (8, 16, or 32 bits) can be selected for each space.
  - — Wait state insertion is controlled for each space.
  - — Control signals are output for each space.
- Cache
  - — Cache area or cache-through area can be selected by access address.
  - — In cache access, in the event of a cache miss 16 bytes are read consecutively in 4-byte units to fill the cache.
  - — In cache-through access, access is performed in accordance with the access size.
- Refresh functions
  - — CAS-before-RAS refresh (auto-refresh) and self-refresh
  - — Refresh interval can be set by refresh counter and clock selection.
  - — Intensive refreshing by means of a refresh count setting (1, 2, 4, 6, or 8)
- Direct DRAM interface
  - — Row address/column address multiplexed output
  - — Burst transfer for reads, fast page mode for continuous access
  - — TP cycle generation to secure RAS precharge time
  - — EDO mode
- Direct synchronous DRAM interface
  - — Row address/column address multiplexed output
  - — Selection of burst read, single write mode or burst read, burst write mode
  - — Bank active mode
- Bus arbitration
  - — All resources are shared with the CPU, and use of the bus is granted on reception of a bus release request from off-chip.

- Refresh counter can be used as interval timer
  — Interrupt request generation on compare match (CMI interrupt request signal)

Section 7 describes the BSC.

### 1.6.5 Cache

The cache uses spatial and temporal localities to provide high-speed memory access. The cache includes the following features.

- 4 kbytes
- 64 entries, 4-way set-associative, 16-byte line length
- LRU replacement algorithm
- Can also be used as 2-kbyte cache and 2-kbyte on-chip RAM

Section 8 describes the cache.

### 1.6.6 Direct Memory Access Controller (DMAC)

The direct memory access controller (DMAC) can be used in place of the CPU to perform high-speed data transfers between external devices equipped with DACK (transfer request acknowledge signal), external memories, on-chip memories, and memory-mapped external devices. The DMAC includes the following features.

- Two channels
- Address space: Architecturally 4 GB
- Choice of data transfer unit: Byte, word (2-byte), longword (4-byte), or 16-byte unit (In 16-byte transfer, four longword reads are followed by four longword writes.)
- Maximum of 16 M (16,777,216) transfers
- In the event of a cache hit, CPU instruction processing and DMA operation can be executed in parallel
- Single address mode transfer: Either the transfer source or transfer destination peripheral device is accessed (selected) by a DACK signal while the other is accessed by address. One data transfer is completed in one bus cycle.

  Possible transfer devices: External devices with DACK and memory-mapped external devices (including external memory)
- Dual address mode transfer: Both the transfer source and transfer destination are accessed by address. Two bus cycles are required for one data transfer.

Possible transfer devices:

— Two external memories
— External memory and memory-mapped external device
— Two memory-mapped external devices
— On-chip memory and memory-mapped external device
— Two on-chip memories
— On-chip memory and external memory

- Transfer requests
  — External request: From the DREQ pin. Edge or level detection, and active-low or active-high mode, can be specified for DREQ.
  — Requests from on-chip peripheral modules: 16-bit timer pulse unit (TPU)
  — Auto-request: The transfer request is generated automatically within the DMAC.
- Choice of bus mode
  — Cycle steal mode
  — Burst mode
- Choice of channel priority order:
  — Fixed mode
  — Round robin mode
- An interrupt request can be sent to the CPU on completion of data transfer

Section 9 describes the DMAC.

### 1.6.7 Division Unit (DIVU)

The division unit (DIVU) performs 64-bit ÷ 32-bit or 32-bit ÷ 32-bit operations. The DIVU includes the following features.

- Signed 64-bit ÷ 32-bit or signed 32-bit ÷ 32-bit operations.
- 32-bit quotient, 32-bit remainder
- Operation execution cycles: 39
- Overflow interrupt enable/disable control
- Instructions that do not access the division unit can be processed in parallel even during execution of division processing.

Section 10 describes the DIVU.

### 1.6.8 16-Bit Free-Running Timer (FRT)

The 16-bit free-running timer (FRT) allows output of two independent waveforms, based on a 16-bit free-running counter, and also permits input pulse width and external clock cycle measurement. The FRT includes the following features.

- Choice of four counter input clocks

  The counter input clock can be selected from three internal clocks (Pϕ/8, Pϕ/32, Pϕ/128) and an external clock (enabling external event counting).
- Two independent comparators

  Two waveform outputs can be generated.
- Input capture

  Choice of rising edge or falling edge
- Counter clear specification

  The counter value can be cleared by compare match A.
- Four interrupt sources

  Two compare match sources, one input capture source, and one overflow source can issue requests independently.

Section 11 describes the FRT.

### 1.6.9 Watchdog Timer (WDT)

The watchdog timer (WDT) is a single-channel timer that enables system monitoring to be carried out. The WDT includes the following features.

- Can be switched between watchdog timer mode and interval timer mode.
- $\overline{\text{WDTOVF}}$ output in watchdog timer mode

  The $\overline{\text{WDTOVF}}$ signal is output externally when the counter overflows, and a simultaneous internal reset of the chip can also be selected (either a power-on reset or manual reset can be specified).
- Interrupt generation in interval timer mode

  An interval timer interrupt is generated when the counter overflows.
- Used when standby mode is cleared or the clock frequency is changed, and in clock pause mode.
- Choice of eight counter input clocks

Section 12 describes the WDT.

### 1.6.10 Serial Communication Interface (SCI)

The serial communication interface has two channels for serial transmission and reception, which may be synchronous or asynchronous. The SCI includes the following features.

- Choice of asynchronous mode or synchronous mode as the serial communication mode

  Asynchronous mode:

  Serial data communication is synchronized by start-stop in character units. The SCI can communicate with a universal asynchronous receiver/transmitter (UART), an asynchronous communication interface adapter (ACIA), or any other chip that employs a standard asynchronous serial communication. There are twelve selectable serial data communication formats.

  — Data length: seven or eight bits
  — Stop-bit length: one or two bits
  — Parity: even, odd, or none
  — Multiprocessor bit: 1 or 0
  — Receive error detection: parity, overrun, and framing errors

  Clocked synchronous mode:

  Serial data communication is synchronized with a clock signal. The SCI can communicate with other chips having a clocked synchronous communication function. There is one serial data communication format.

  — Data length: eight bits
  — Receive error detection: overrun errors
- Full-duplex communication: Transmission and reception are independent, so the SCI can transmit and receive simultaneously. Double buffering permits continuous data transfer in both the transmit and receive directions.
- On-chip baud-rate generator: Selectable bit rates
- Internal or external transmit/receive clock source: Baud-rate generator (internal) or SCK input signal (external)
- Four types of interrupts

  Transmit-data-empty, transmit-end, receive-data-full, and receive-error interrupts are requested independently.

Section 13 describes the SCI.

### 1.6.11 Smart Card Interface

An IC card (smart card) interface conforming to the ISO/IEC 7816-3 (Identification Card) data transmission protocol format (T = 0: asynchronous full-duplex character transmission protocol) is supported as a serial communication interface (SCI) extension function. The smart card interface includes the following features.

- Asynchronous mode
  - Data length: 8 bits
  - Parity bit generation and check
  - 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
- On-chip baud rate generator allows any bit rate to be selected.
- Three interrupt sources

  There are three interrupt sources—transmit-data-empty, receive-data-full, and transmit/receive error—that can issue requests independently.

Section 14 describes the smart card interface.

### 1.6.12 Serial Communication Interface with FIFO (SCIF)

The chip is equipped with a two-channel serial communication interface with FIFO (SCIF). Sixteen-stage FIFO registers are provided for both transmit and receive, enabling fast, efficient, and continuous communication. The SCIF includes the following features.

- Asynchronous serial communication
  - Serial data communication is executed using an asynchronous system in which synchronization is achieved character by character. Serial data communication can be carried out with standard asynchronous communication chips such as a Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA). There is a choice of eight serial data transfer formats.
  - Data length: 7 or 8 bits
  - Stop bit length: 1 or 2 bits
  - Parity: Even/odd/none
  - Receive error detection: Parity and framing errors
  - Break detection: If the receive data following that in which a framing error occurred is also at the space “0” level, and there is a frame error, a break is detected. When a framing error occurs, a break can also be detected by reading the RxD pin level directly from the port data register (PBDR).
- Full-duplex communication

The transmitter and receiver are independent units, enabling transmission and reception to be performed simultaneously. The transmitter and receiver both have a 16-stage FIFO buffer structure, enabling fast and continuous serial data transmission and reception.

- On-chip baud rate generator allows any bit rate to be selected.
- Internal or external transmit/receive clock source

  Choice of internal clock from baud rate generator or external clock from SCK pin
- Four interrupt sources

  There are four interrupt sources—transmit-FIFO-data-empty, break, receive-FIFO-data-full, and receive-error—that can issue requests independently.
- When not in use, the SCIF can be stopped by halting its clock supply to reduce power consumption.
- Modem control functions ($\overline{\text{RTS}}$ and $\overline{\text{CTS}}$)
- The amount of data in the transmit/receive FIFO registers, and the number of receive errors in the receive data in the receive FIFO register, can be detected.
- A timeout error (DR) can be detected during reception.

Section 15 describes the SCIF.

### 1.6.13 IrDA

The chip has an on-chip Infrared Data Association (IrDA) interface based on the IrDA 1.0 system, enabling infrared communication. By means of a register setting, this interface can also be used as a serial communication interface (SCIF). The IrDA includes the following features.

- Conforms to IrDA 1.0
- Asynchronous serial communication
  - Data length: 8 bits
  - Stop bit length: 1 bit
  - Parity bit: None
- Built-in 16-stage FIFOs for transmit and receive
- On-chip baud rate generator allowing selection of bit rate
- Protection function that prevents receiver from being affected during transmission
- Clock supply is halted when the IrDA is not in use, to reduce power consumption.

Section 16 describes the IrDA.

### 1.6.14 Serial I/O (SIO)

The serial I/O module (SIO) functions mainly as an interface between the chip and a codec or modem analog front-end. The SIO has the following features:

- Full-duplex operation
  Independent transmit/receive registers and independent transmit/receive clocks
- Double-buffered receive/transmit ports
  Continuous transmission/reception possible
- Interval transfer mode and continuous transfer mode
- Memory-mapped receive register, transmit register, control register, and status register
  With the exception of SIRSR and SITSR, these registers are memory-mapped and can be accessed by a MOV instruction.
- Choice of 8- or 16-bit data length
- Data transfer communication by means of polling or interrupts
  Data transfer can be monitored by polling the receive data register full flag (RDRF) and transmit data register empty flag (TDRE) in the serial status register. Interrupt requests can be generated during data transfer by setting the receive interrupt request flag and transmit interrupt request flag.
- MSB-first transfer between SIO and data I/O

Section 17 describes the SIO.

### 1.6.15 16-Bit Timer Pulse Unit (TPU)

The chip has an on-chip 16-bit timer pulse unit (TPU) comprising three 16-bit timer channels. The TPU includes the following features.

- Maximum 8-pulse input/output
- A total of eight timer general registers (TGR) are provided (four for channel 0, and two each for channels 1 and 2).
  - Each register can be set independently as an output compare/input capture register.
  - TGRC and TGRD for channel 0 can be used as buffer registers.
- Choice of seven or eight counter input clocks for each channel
- The following operations can be set for each channel:
  - Waveform output by compare match: Selection of 0, 1, or toggle output
  - Input capture function: Choice of rising edge, falling edge, or both edge detection
  - Counter clear operation: Counter clearing possible by compare match or input capture
  - Synchronous operation: Multiple timer counters (TCNT) can be written to simultaneously
    Simultaneous clearing by compare match and input capture possible

Simultaneous register input possible by counter synchronous operation

— PWM mode: Any PWM output duty can be set
Maximum 7-phase PWM output possible by combination with synchronous operation

- Buffer operation settable for channel 0
  — Input capture register double-buffering possible
  — Automatic rewriting of output compare register possible
- Phase counting mode settable independently for channels 1 and 2
  — Two-phase encoder pulse up/down-count possible
- Fast access via internal 16-bit bus
  — Fast access possible via a 16-bit interface
- 13 interrupt sources
  — For channel 0, four compare match/input capture dual-function interrupts and one overflow interrupt can be requested independently.
  — For channels 1 and 2, two compare match/input capture dual-function interrupts, one overflow interrupt, and one underflow interrupt can be requested independently.

Section 18 describes the TPU.

### 1.6.16 Hitachi User Debug Interface (H-UDI)

The Hitachi user debug interface (H-UDI) provides data transfer and interrupt request functions. The H-UDI performs serial transfer by means of external signal control.

The H-UDI has the following features conforming to the IEEE 1149.1 standard.

- Five test signals (TCK, TDI, TDO, TMS, and $\overline{\text{TRST}}$)
- TAP controller
- Instruction register
- Data register
- Bypass register

The H-UDI has two instructions.

- Bypass mode
  Test mode conforming to IEEE 1149.1
- H-UDI interrupt
  H-UDI interrupt request to INTC

Section 19 describes the H-UDI.

### 1.6.17 Pin Function Controller (PFC)

The pin function controller (PFC) consists of registers to select multiplexed pin functions and input/output direction. The pin function and input/output direction can be selected for individual pins regardless of the operating mode of the chip.

Section 20 describes the PFC.

### 1.6.18 I/O Ports

There are two I/O ports, designated A and B. Port A is a 13-bit input/output port and port B is a 16-bit input/output port. The port pins are multiplexed pins functioning as general input/output and other functions. (The function of multiplexed pins is selected by means of the pin function controller (PFC).) Ports A and B each have a data register for storing pin data.

Section 21 describes the I/O ports.

### 1.6.19 Processing States

The CPU has five processing states: program execution, reset, exception processing, power-down, and bus release. The transitions between the states are shown in figure 1.4.

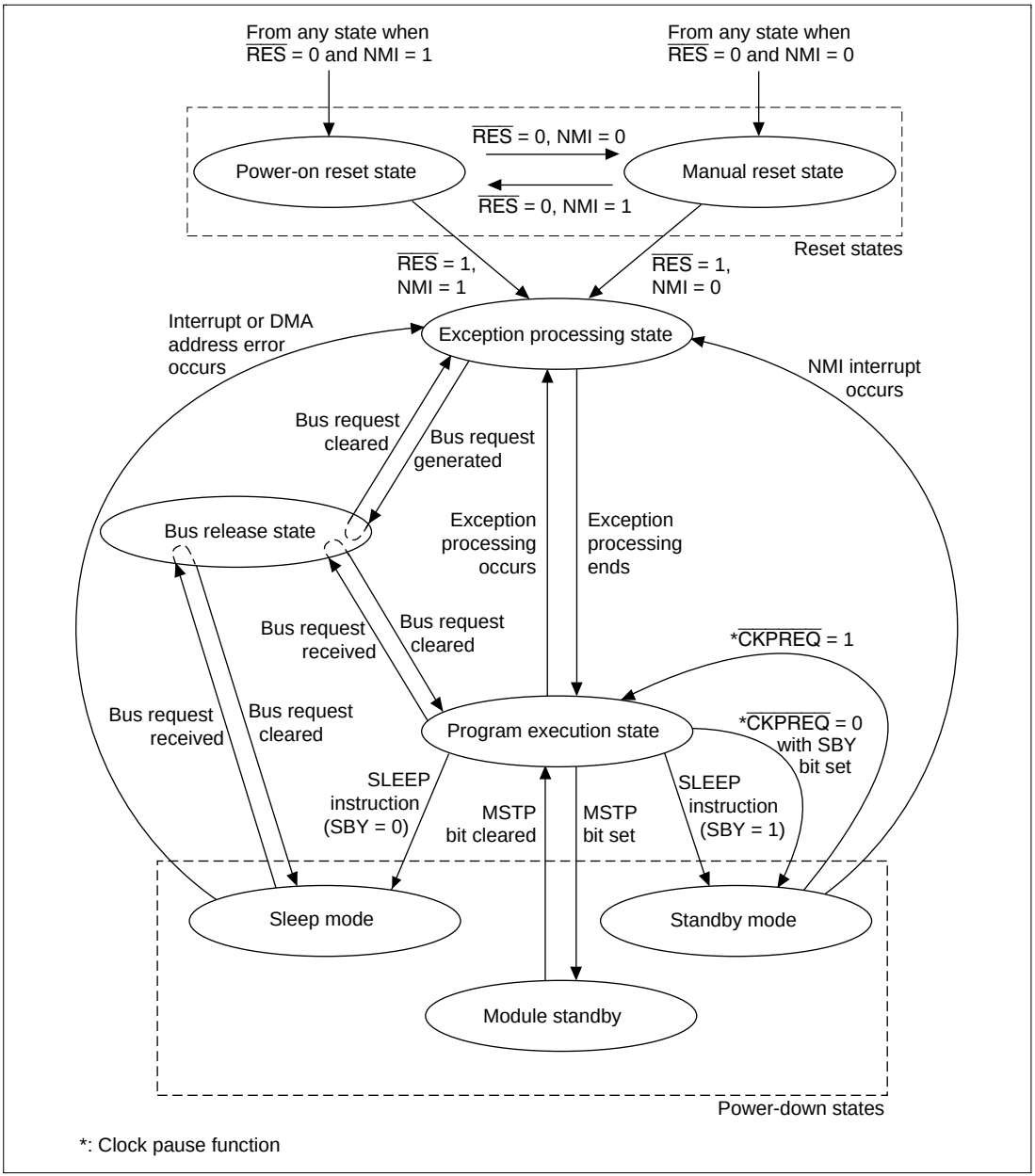



**Figure 1.4 Transitions between Processing States**

**Reset State:** In this state, the CPU is reset. The reset state is entered when the $\overline{\text{RES}}$ pin goes low. The power-on reset state is entered if the NMI pin is high, and the manual reset state is entered if the NMI pin is low.

**Exception Processing State:** The exception processing state is a transient state that occurs when the CPU changes the processing state flow because of a reset, interrupt, or other exception processing source.

For a reset, the execution start address in the form of the initial value of the program counter (PC), and the initial value of the stack pointer (SP), are fetched from the exception-processing vector table and stored. The processor then branches to the execution start address and begins program execution.

For an interrupt or other type of exception, the processor references the SP and saves the PC and status register (SR) contents to the stack. The processor fetches the start address of the exception-processing routine from the exception-processing vector table, then branches to that address and begins program execution.

The processing state then entered is the program execution state.

**Program Execution State:** In the program execution state, the processor sequentially executes program instructions.

**Power-Down State:** In the power-down state, the CPU operation halts and power consumption declines. The SLEEP instruction places the processor in the power-down state. This state has two modes: sleep mode and standby mode. The processor also has a module standby function.

**Bus Release State:** In the bus release state, the CPU releases the bus to a device that has requested it.

**Power-Down State:** In addition to the normal program execution state, another CPU processing state called the power-down state is provided. In this state, CPU operation is halted and power consumption is reduced. The power-down state includes two modes—sleep mode and standby mode—and a module standby function.

- Sleep mode

  When the standby bit SBY in the standby control register 1 (SBYCR1) is cleared to 0 and a SLEEP instruction is executed, the processor transitions from the program execution state to sleep mode. In sleep mode, the CPU halts but the data in the CPU's internal registers, on-chip cache memory, and on-chip RAM is retained. The on-chip peripheral modules do not halt in the sleep mode. A reset, any interrupt, or a DMA address error returns the processor to the ordinary program execution state through the exception processing state.

- Standby mode

  To enter the standby mode, set the SBY bit in the standby control register 1 (SBYCR1) to 1 and execute a SLEEP instruction. In standby mode, all CPU, on-chip peripheral module and oscillator functions are halted.

  When standby mode is entered, the DMAC's DMA master enable bit should be cleared to 0. The cache should be turned off before entering standby mode. Cache and on-chip RAM data is not held in standby mode.

  A reset or an external NMI interrupt returns the processor from standby mode. For resets, the CPU returns to the ordinary program execution state through the reset exception processing state after the oscillator stabilization time has elapsed. For NMI interrupts, the CPU returns to the ordinary program execution state through the exception processing state after the oscillator stabilization time has elapsed.

  When a transition is made to standby mode using the clock pause function, it is possible to change the frequency of the CKIO pin input clock, or to halt the clock itself. When SBY in SBYCR1 is set to 1 and a low level is applied to the $\overline{\text{CKPREQ}}$/CKM pin, a transition is made to standby mode and a low level is output from the $\overline{\text{CKPACK}}$ pin. The clock can then be stopped, or its frequency changed.

  On-chip peripheral module states and pin states are the same as in the normal standby mode entered by means of the SLEEP instruction. A transition to the program execution state is made by applying a high level to the $\overline{\text{CKPREQ}}$/CKM pin. In this mode, power consumption goes low because the oscillator stops.

- Module standby function

  A module standby function is provided for the following on-chip peripheral modules: the direct memory access controller (DMAC), DSP, division unit (DIVU), 16-bit free-running timer (FRT), serial communication interface (SCI), serial communication interface with FIFO (SCIF), serial I/O (SIO), user break controller (UBC), and timer pulse unit (TPU).

  Setting module stop bits 11–0 (MSTP11–MSTP0) to 1 in the standby control register (SBYCR1/2) stops the clock supply to the corresponding on-chip peripheral modules. Use of this function enables the power consumption to be reduced.

  The module standby function is cleared by clearing the corresponding MSTP bits to 0.

  Instructions that use a DSP register or MACH/MACL as an operand cannot be used when the DSP has been placed in the module standby state.

  When using the DMAC module standby function, the direct memory access controller's DMA master enable bit should be cleared to 0.

**Table 1.1 Power-Down Modes**

| Mode | Transition Condition | State | | | | | Canceling Procedure |
|---|---|---|---|---|---|---|---|
| | | Clock | CPU | On-Chip Peripheral Modules | CPU Registers | On-Chip Cache or On-Chip RAM | |
| Sleep mode | Executes SLEEP instruction with SBY bit set to 0 in SBYCR1 | Run | Halt | Run | Held | Held | 1. Interrupt<br>2. DMA address error<br>3. Power-on reset<br>4. Manual reset |
| Standby mode | Executes SLEEP instruction with SBY bit set to 1 in SBYCR1 | Halt | Halt | Halt and initialized*[1] | Held | Undefined | 1. NMI interrupt<br>2. Power-on reset<br>3. Manual reset |
| Module standby function | MSTP bit for individual module is set | Run | Runs (DSP halts) | Clock supply halted to specified modules, and modules initialized*[2] | Held | Held | 1. Clear MSTP bit<br>2. Power-on reset<br>3. Manual reset |

Notes: 1. Depends on individual peripheral module or pin.
2. DMAC, DSP, and DIV registers and specified module interrupt vectors retain their set values.

## 2.1 Register Configuration

The register set consists of sixteen 32-bit general registers, six 32-bit control registers and ten 32-bit system registers.

This chip is upwardly compatible with the SH-1, SH-2 on the object code level. For this reason, several registers have been added to the previous SuperH microcontroller registers. The added registers are the three control registers: repeat start register (RS), repeat end register (RE), and modulo register (MOD) and the eight system registers: DSP status register (DSR), and A0, A1, M0, M1, X0, X1, Y0, and Y1 among the DSP data registers.

The general registers are used in the same manner as the SH-1, SH-2 with regard to SuperH microcontroller-type instructions. With regard to DSP type instructions, they are used as address and index registers for accessing memory.

### 2.1.1 General Registers

There are 16 general registers (Rn) numbered R0–R15, which are 32 bits in length. General registers are used for data processing and address calculation.

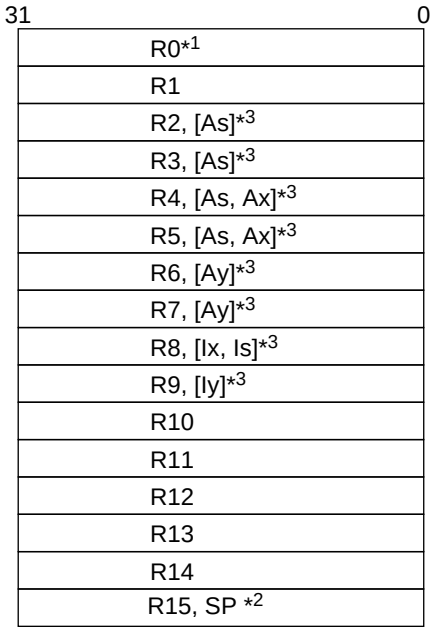
With SuperH microcomputer type instructions, R0 is also used as an index register. Several instructions are limited to use of R0 only. R15 is used as the hardware stack pointer (SP). Saving and recovering the status register (SR) and program counter (PC) in exception processing is accomplished by referencing the stack using R15.

With DSP type instructions, eight of the 16 general registers are used for the addressing of X, Y data memory and data memory (single data) using the C bus.

R4, R5 are used as an X address register (Ax) for X memory accesses, and R8 is used as an X index register (Ix). R6, R7 are used as a Y address register (Ay) for Y memory accesses, and R9 is used as a Y index register (Iy). R2, R3, R4, R5 are used as a single data address register (As) for accessing single data using the C bus, and R8 is used as a single data index register (Is).

DSP type instructions can simultaneously access X and Y data memory. There are two groups of address pointers for designating X and Y data memory addresses.

Figure 2.1 shows the general registers.




Notes:
1. R0 also functions as an index register in the indirect indexed register addressing mode and indirect indexed GBR addressing mode. In some instructions, only the R0 functions as a source register or destination register.
2. R15 functions as a hardware stack pointer (SP) during exception processing.
3. Used as memory address registers, memory index registers with DSP type instructions.

**Figure 2.1 General Register Configuration**

With the assembler, symbol names are used for R2, R3 ... R9. If it is wished to use a name that makes clear the role of a register for DSP type instructions, a different register name (alias) can be used. This is written in the following manner for the assembler.

```
Ix:     .REG (R8)
```

The name Ix is an alias for R8. The other aliases are assigned as follows:

```
Ax0:    .REG (R4)
Ax1:    .REG (R5)
Ix:     .REG (R8)
Ay0:    .REG (R6)
Ay1:    .REG (R7)
Iy:     .REG (R9)
As0:    .REG (R4)   defined when an alias is required for single data transfer
As1:    .REG (R5)   defined when an alias is required for single data transfer
As2:    .REG (R2)   defined when an alias is required for single data transfer
As3:    .REG (R3)   defined when an alias is required for single data transfer
Is:     .REG (R8)   defined when an alias is required for single data transfer
```

### 2.1.2 Control Registers

The six 32-bit control registers consist of the status register (SR), repeat start register (RS), repeat end register (RE), global base register (GBR), vector base register (VBR), and modulo register (MOD).

The SR register indicates processing states.

The GBR register functions as a base address for the indirect GBR addressing mode, and is used for such as on-chip peripheral module register data transfers.

The VBR register functions as the base address of the exception processing vector area (including interrupts).

The RS and RE registers are used for program repeat (loop) control. The repeat count is designated in the SR register repeat counter (RC), the repeat start address in the RS register, and the repeat end address in the RE register. However, note that the address values stored in the RS and RE registers are not necessarily always the same as the physical start and end address values of the repeat.

The MOD register is used for modulo addressing to buffer the repeat data. The modulo addressing designation is made by DMX or DMY, the modulo end address (ME) is designated in the upper 16 bits of the MOD register, and the modulo start address (MS) is designated in the lower 16 bits. Note that the DMX and DMY bits cannot simultaneously designate modulo addressing. Modulo addressing is possible with X and Y data transfer instructions (MOVX, MOVY). It is not possible with single data transfer instructions (MOVS).

Figure 2.2 shows the control registers. Table 2.1 indicates the SR register bits.

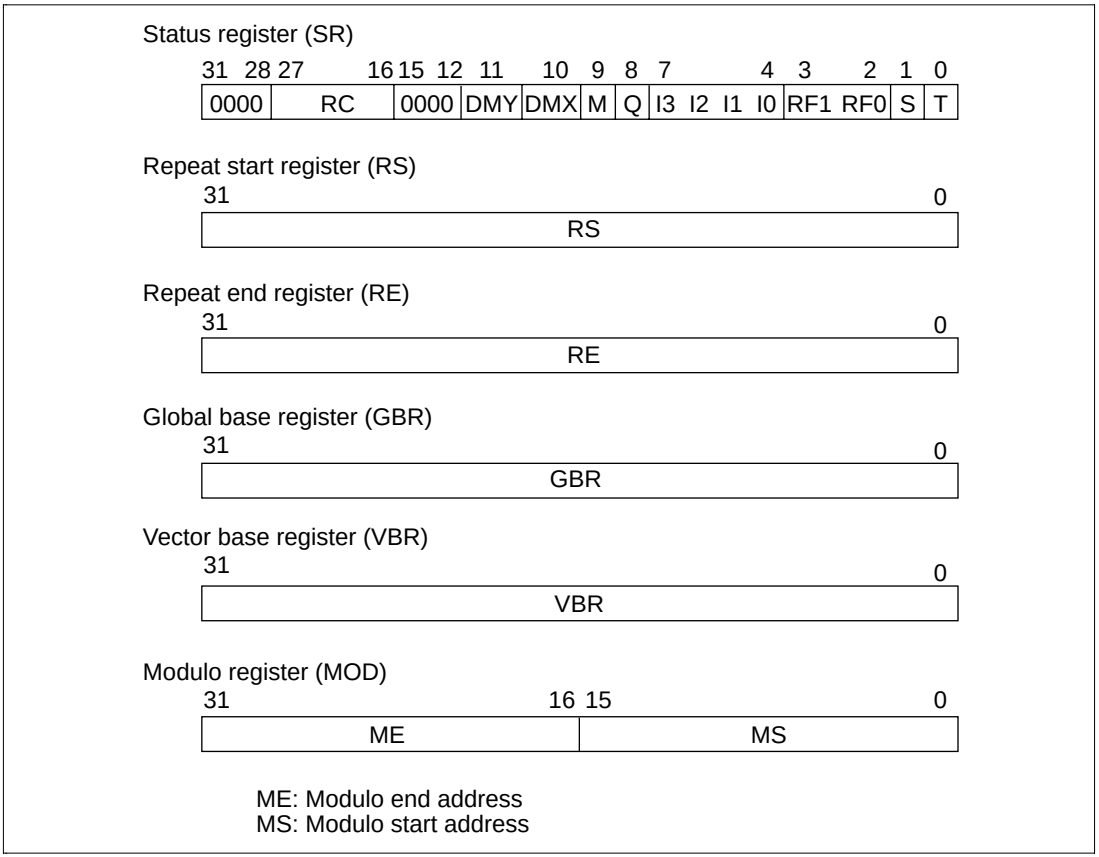



**Figure 2.2 Control Register Configuration**

**Table 2.1 SR Register Bits**

| Bit | Name (Abbreviation) | Function |
|---|---|---|
| 27–16 | Repeat counter (RC) | Designate the repeat count (2–4095) for repeat (loop) control |
| 11 | Y pointer usage modulo addressing designation (DMY) | 1: modulo addressing mode becomes valid for Y memory address pointer, Ay (R6, R7) |
| 10 | X pointer usage modulo addressing designation (DMX) | 1: modulo addressing mode becomes valid for X memory address pointer, Ax (R4, R5) |
| 9 | M bit | Used by the DIV0S/U, DIV1 instructions |
| 8 | Q bit | Used by the DIV0S/U, DIV1 instructions |
| 7–4 | Interrupt request mask (I3–I0) | Indicate the receive level of an interrupt request (0 to 15) |
| 3–2 | Repeat flags (RF1, RF0) | Used in zero overhead repeat (loop) control. Set as below for an SETRC instruction<br>For 1 step repeat 00 RE—RS=–4<br>For 2 step repeat 01 RE—RS=–2<br>For 3 step repeat 11 RE—RS=0<br>For 4 steps or more 10 RE—RS>0 |
| 1 | Saturation arithmetic bit (S) | Used with MAC instructions and DSP instructions<br>1: Designates saturation arithmetic (prevents overflows) |
| 0 | T bit | For MOVT, CMP/cond, TAS, TST, BT, BT/S, BF, BF/S, SETT, CLRT and DT instructions,<br>0: represents false<br>1: represents true |
| | | For ADDV/ADDC, SUBV/SUBC, DIV0U/DIV0S, DIV1, NEGC, SHAR/SHAL, SHLR/SHLL, ROTR/ROTL and ROTCR/ROTCL instructions,<br>1: represents occurrence of carry, borrow, overflow or underflow |
| 31–28<br>15–12 | 0 bit | 0: 0 is always read out; write a 0 |

There are dedicated load/store instructions for accessing the RS, RE and MOD registers. For example, the RS register is accessed as follows.

```
LDC     Rm,RS;          Rm→RS
LDC.L   @Rm+,RS;        (Rm)→RS,Rm+4→Rm
STC     RS,Rn;          RS→Rn
STC.L   RS,@-Rn;        Rn-4→Rn,RS→(Rn)
```

The following instructions set addresses in the RS, RE registers for zero overhead repeat control:

```
LDRS    @(disp,PC);     disp×2 + PC→RS
LDRE    @(disp,PC);     disp×2 + PC→RE
```

The GBR register and VBR register are the same as the previous SuperH microprocessor registers. An RC counter and four control bits (DMX bit, DMY bit, RF1 bit, RF0 bit) have been added to the SR register. The RS, RE and MOD registers are new registers.

### 2.1.3 System Registers

System registers consist of four 32-bit registers: high and low multiply and accumulate registers (MACH and MACL), the procedure register (PR), and the program counter (PC). The MACH and MACL store the results of multiplication or multiply and accumulate operations*. The PR stores the return address from the subroutine procedure. The PC indicates the address of the program in execution; it controls the flow of the processing. The PC indicates the fourth byte after the instruction currently being executed. These registers are the same as those in the SuperH microprocessor.

Note: These are used only when executing an instruction that was supported by SH-1 and SH-2. They are not used for newly added multiplication instructions (PMULS).

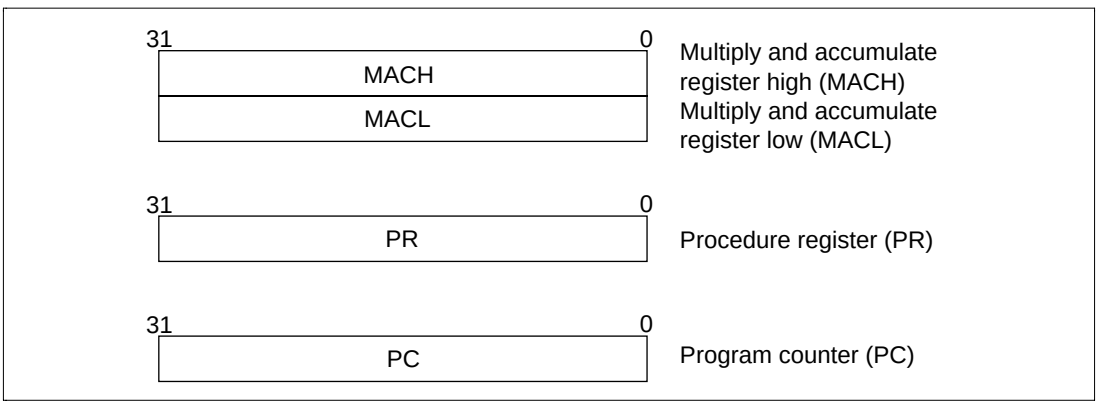



**Figure 2.3 System Register Configuration**

In addition, among the DSP unit usage registers (DSP registers) described in 2.1.4 DSP Registers, the DSP status register (DSR) and the five registers A0, X0, X1, Y0 and Y1 of the eight data registers are treated as system registers. Among these, the A0 is a 40-bit register, but when data is output from the A0 register, the guard bit section (A0G) is disregarded; when data is input to the A0 register, the MSB of the data is copied into the guard bit section (A0G).

### 2.1.4 DSP Registers

The DSP unit has eight data registers and one control register as its DSP registers.

The DSP data registers are comprised of the two 40-bit registers A0 and A1, and the six 32-bit registers M0, M1, X0, X1, Y0 and Y1. The A0 and A1 registers have the 8-bit guard bits A0G and A1G, respectively.

The DSP data registers are used for the transfer and processing of the DSP data of DSP instruction operands. There are three types of instructions that access DSP data registers: those for DSP data processing, and those for X or Y data transfer processing.

The control register is the 32-bit DSP status register (DSR) that represents operation results. The DSR register has bits that represent operation results, a signed greater than bit (GT), a zero bit (Z), a negative value bit (N), an overflow bit (V), a DSP status bit (DC: DSP condition), and a status selection bit (CS: condition select) for controlling DC bit setting.

The DC bit represents one status flag and is very similar to the SuperH CPU core T bit. For conditional DSP type instructions, DSP data processing execution is controlled in accordance with the DC bit. This control is related to execution in the DSP unit only, and only DSP registers are updated. It bears no relation to address calculation or such SuperH microprocessor CPU core execution instructions as load/store instructions. The control bits CS (bits 2 to 0) designate the status for setting the DC bit.

DSP type instructions are comprised of unconditional DSP type instructions and conditional DSP type instructions. The status and DC bits are updated in unconditional DSP type data processing, with the exception of the PMULS, MOVX, MOVY and MOVS instructions. Conditional DSP type instructions are executed according to the status of the DC bit, but regardless of whether or not they are executed, the DSR register is not updated.

Figure 2.4 shows the DSP registers. The DSR register bit functions are shown in table 2.2.

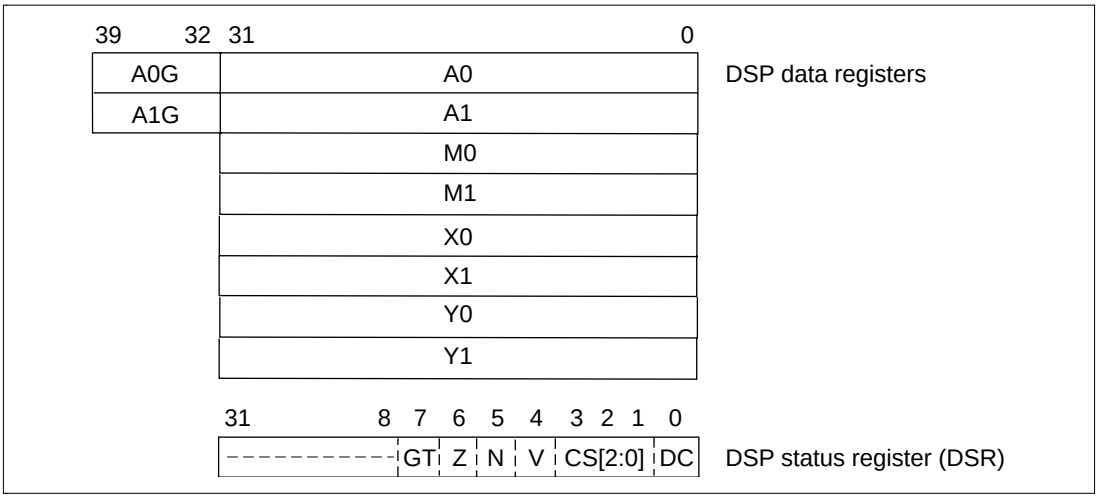



**Figure 2.4 DSP Register Configuration**

**Table 2.2 DSR Register Bits**

| Bit | Name (Abbreviation) | Function |
|---|---|---|
| 31–8 | Reserved bits | 0: Always read out; always use 0 as a write value |
| 7 | Signed greater than bit (GT) | Indicates that the operation result is positive (excepting 0), or that operand 1 is greater than operand 2<br>1: Operation result is positive, or operand 1 is greater |
| 6 | Zero bit (Z) | Indicates that the operation result is zero (0), or that operand 1 is equal to operand 2<br>1: Operation result is zero (0), or equivalence |
| 5 | Negative bit (N) | Indicates that the operation result is negative, or that operand 1 is smaller than operand 2<br>1: Operation result is negative, or operand 1 is smaller |
| 4 | Overflow bit (V) | Indicates that the operation result has overflowed<br>1: Operation result has overflowed |
| 3–1 | Status selection bits (CS) | Designate the mode for selecting the operation result status set in the DC bit<br>Do not set either 110 or 111<br>000: Carry/borrow mode<br>001: Negative value mode<br>010: Zero mode<br>011: Overflow mode<br>100: Signed greater mode<br>101: Signed above mode |
| 0 | DSP status bit (DC) | Sets the status of the operation result in the mode designated by the CS bits<br>0: Designated mode status not realized (unrealized)<br>1: Designated mode status realized |

A0, X0, X1, Y0, Y1, and the DSR registers are treated as system registers by CPU core instructions.

```
LDS     Rn,DSR;
LDS.L   @Rn+,DSR;
```

### 2.1.5 Notes on Guard Bits and Overflow Treatment

DSP unit data operations are fundamentally performed as 32-bit, but these operations are always executed with a 40-bit length including the 8-bit guard section. When the guard bit section does not match the value of the 32-bit section MSB, the operation result is treated as an overflow. In this case, the N bit indicates the correct status of the operation result regardless of the existence or not of an overflow. This is so even if the destination operand is a 32-bit length register. The 8-bit section guard bits are always presupposed and each status flag is updated.

When place overflows occur so that the correct result cannot be displayed even when the guard bits are used, the N flag cannot indicate the correct status.

### 2.1.6 Initial Values of Registers

Table 2.3 lists the values of the registers after reset.

**Table 2.3 Initial Values of Registers**

| Classification | Register | Initial Value |
|---|---|---|
| General registers | R0–R14 | Undefined |
| | R15 (SP) | Value of the SP in the vector address table |
| Control registers | SR | Bits I3–I0 are 1111 (H'F), the reserved bits, RC, DMY, and DMX are 0, and other bits are undefined |
| | RS | Undefined |
| | RE | |
| | GBR | Undefined |
| | VBR | H'00000000 |
| | MOD | Undefined |
| System registers | MACH, MACL, PR | Undefined |
| | PC | Value of the PC in the vector address table |
| DSP registers | A0, A0G, A1, A1G, M0, M1, X0, X1, Y0, Y1 | Undefined |
| | DSR | H'00000000 |

## 2.2 Data Formats

### 2.2.1 Data Format in Registers

Register operand data size is always longword (32 bits). When loading data from memory into a register, if the memory operand is a byte (8 bits) or a word (16 bits), it is sign-extended into a longword, then loaded into the register.

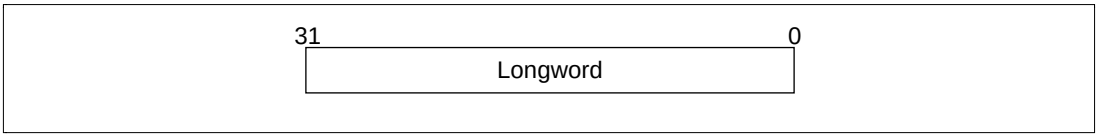



**Figure 2.5 Register Data Format**

### 2.2.2 Data Formats in Memory

These formats are classified into bytes, words, and longwords.

Place byte data in any address, word data from 2n addresses, and longword data from 4n addresses. An address error will occur if accesses are made from any other boundary. In such cases, the access results cannot be guaranteed. In particular, the stack area referred to by the hardware stack pointer (SP, R15) stores the program counter (PC) and status register (SR) as longwords, so establish the hardware stack pointer so that a 4n value will always result.

To enable sharing of the processor accessing memory in little-endian mode and memory, the CS2 space (area 2) and CS4 space (area4) have a function that allows access in little-endian mode. The order of byte data differs between little-endian mode and normal big-endian mode.

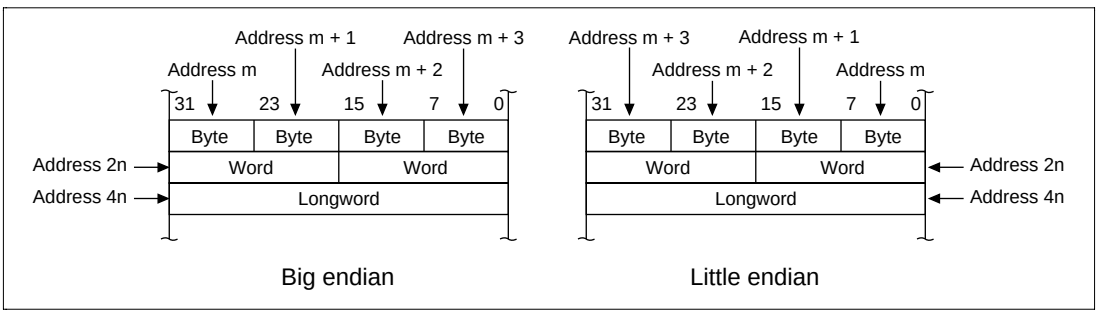



**Figure 2.6 Data Formats in Memory**

### 2.2.3 Immediate Data Format

Byte immediate data is placed in an instruction code.

With the MOV, ADD, and CMP/EQ instructions, immediate data is sign-extended and operated in registers as longword data. Immediate data accessed by the TST, AND, OR, and XOR instructions is zero-extended and handled as longword data. Consequently, AND instructions with immediate data always clear the upper 24 bits of the destination register.

Word or longword immediate data is not located in the instruction code; it should be placed in a memory table. Use an immediate data transfer instruction (MOV) to refer the memory table using the PC relative addressing mode with displacement.

### 2.2.4 DSP Type Data Formats

This chip has three different types of data format that correspond to various instructions. These are the fixed-point data format, the integer data format, and the logical data format.

The DSP type fixed-point data format has a binary point fixed between bits 31 and 30. There are three types: with guard bits, without guard bits, and multiplication input; each with different valid bit lengths and value ranges.

The DSP type integer data format has a binary point fixed between bits 16 and 15. There are three types: with guard bits, without guard bits, and shift amount; each with different valid bit lengths and value ranges. The shift amount of the arithmetic shift (PSHA) has a 7 bit range and can express values from –64 to +63, but the actual valid values are from –32 to +32. In the same manner, the shift amount of the logical shift has a 6 bit range, but the actual valid values are from –16 to +16.

The DSP type logical data format does not have a decimal point.

The data format and valid data length are determined by the instructions and DSP registers.

Figure 2.7 shows the three DSP type data formats and binary point positions. The SuperH type data format is also shown for reference.

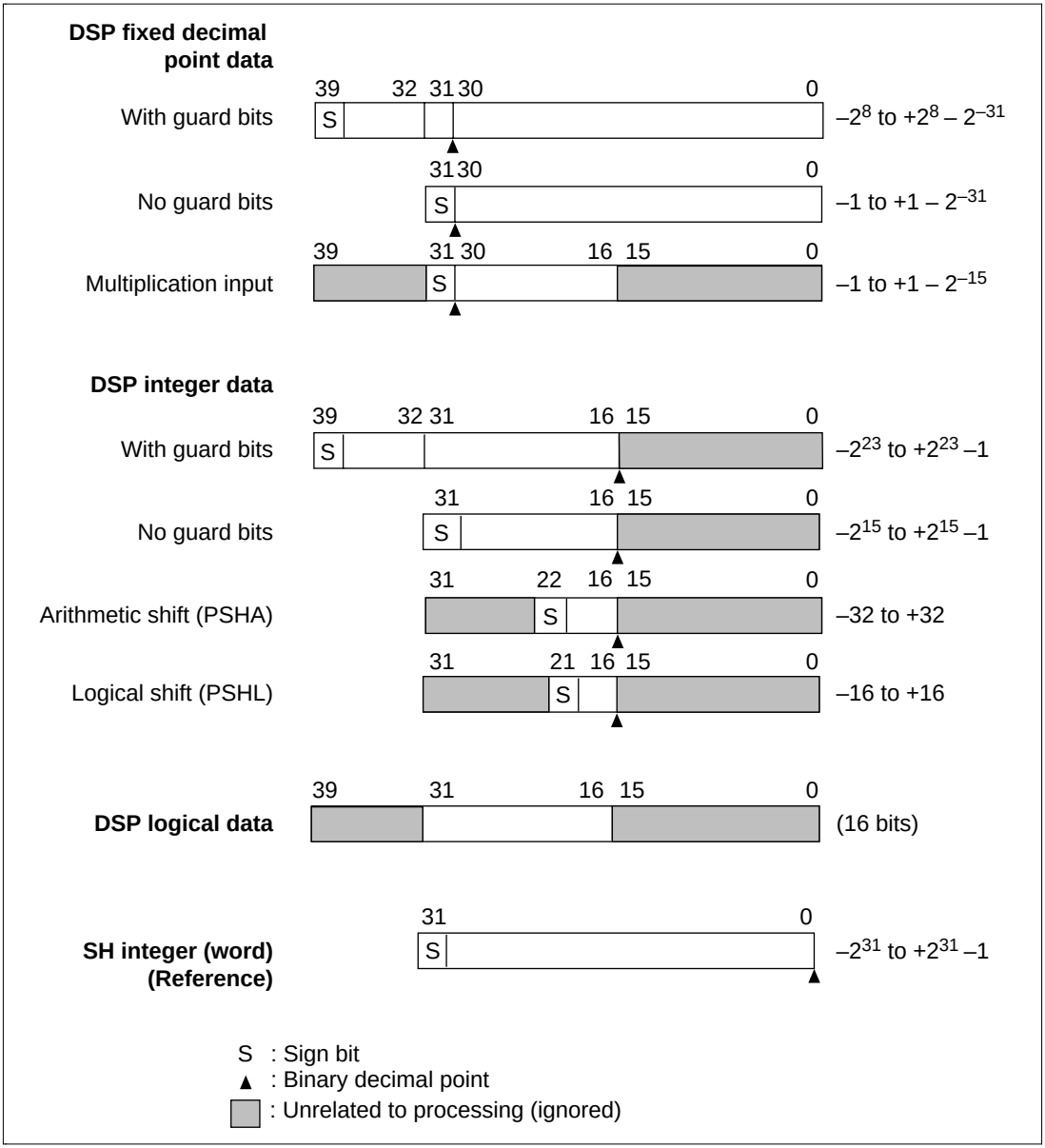



**Figure 2.7 DSP Type Data Formats**

### 2.2.5 DSP Type Instructions and Data Formats

The DSP data format and valid data length are determined by DSP type instructions and DSP registers. There are three types of instructions that access DSP data registers, DSP data processing, X, Y data transfer processing, and single data transfer processing instructions.

**DSP Data Processing:** The guard bits (bits 39–32) are valid when the A0 and A1 registers are used as source registers in DSP fixed-point data processing. When any registers other than A0, A1 (e.i., M0, M1, X0, X1, Y0, Y1 registers) are used as source registers, the sign-extended part of that register data becomes the bits 39 to 32 data. When the A0 and A1 registers are used as destination registers, the guard bits (bits 39–32) are valid. When any registers other than A0, A1 are used as destination registers, bits 39 to 32 of the result data are disregarded.

Processing for DSP integer data is the same as the DSP fixed-point data processing. However, the lower word (the lower 16 bits, bits 15–0) of the source register is disregarded. The lower word of the destination register is cleared to 0.

In DSP logical data processing, the upper word (the upper 16 bits, bits 31–16) of the source register is valid. The lower word and the guard bits of the A0, A1 registers are disregarded. The upper word of the destination register is valid. The lower word and the guard bits of the A0, A1 registers are cleared to 0.

**X, Y Data Transfers:** The MOVX.W and MOVY.W instructions access X, Y memory via the 16-bit X, Y data buses. The data loaded into registers and data stored from registers is always the upper word (the upper 16 bits, bits 31–16).

When loading, the MOVX.W instruction loads X memory, with the X0 and X1 registers as the destination registers. The MOVY.W instruction loads Y memory, with the Y0 and Y1 registers as the destination registers. Data is stored in the upper word of the register; the lower word is cleared to 0.

The upper word data of the A0, A1 registers can be stored in X or Y memory with these data transfer instructions, but storing is not possible from any other registers. The guard bits and the lower word of the A0, A1 registers are disregarded.

**Single Data Transfers:** The MOVS.W and MOVS.L instructions can access any memory via the data bus (CDB). All DSP registers are connected to the CDB bus, and they can become source or destination registers during data transfers. The two data transfer modes are word and longword.

In word mode, data is loaded to and stored in the upper word of the DSP register, with the exception of the A0G, A1G registers. In longword mode, data is loaded to and stored in the 32 bits of the DSP register, with the exception of the A0G, A1G registers. The A0G, A1G registers can be treated as independent registers during single data transfers. The load/store data length for the A0G, A1G registers is 8 bits.

If DSP registers are used as source registers in word mode, when data is stored from any registers other than A0G, A1G, the data in the upper word of the register is transferred. In the case of the A0, A1 registers, the guard bits are disregarded. When the A0G, A1G registers are the source registers in word mode, only 8 bits of the data are stored from the registers; the upper bits are sign-extended.

If the DSP registers are used as destination registers in word mode, the load is to the upper word of the register, with the exception of A0G, A1G. When data is loaded to any register other than A0G, A1G, the lower word of the register is cleared to 0. In the case of the A0, A1 registers, the data sign is extended and stored in the guard bits; the lower word is cleared to 0. When the A0G, A1G registers are the destination registers in word mode, the least significant 8 bits of the data are loaded into the registers; the A0, A1 registers are not zero cleared but retain their previous values.

If the DSP registers are used as source registers in longword mode, when data is stored from any registers other than A0G, A1G, the 32 bits (data) of the register are transferred. When the A0, A1 registers are used as the source registers the guard bits are disregarded. When the A0G, A1G registers are the source registers in longword mode, only 8 bits of the data are stored from the registers; the upper bits are sign-extended.

If the DSP registers are used as destination registers in longword mode, the load is to the 32 bits of the register, with the exception of A0G, A1G. In the case of the A0, A1 registers, the data sign is extended and stored in the guard bits. When the A0G, A1G registers are the destination registers in longword mode, the least significant 8 bits of the data are loaded into the registers; the A0, A1 registers are not zero cleared but retain their previous values.

Tables 2.4 and 2.5 indicate the register data formats for DSP instructions. Some registers cannot be accessed by certain instructions. For example, the PMULS instruction can designate the A1 register as a source register but cannot designate A0 as such. Refer to the instruction explanations for details.

Figure 2.8 shows the relationship between the buses and the DSP registers during transfers.

**Table 2.4 Source Register Data Formats for DSP Instructions**

| Register | Instruction | | Guard Bits 39–32 | Register Bits 31–16 | Register Bits 15–0 |
|---|---|---|---|---|---|
| A0, A1 | DSP operation | Fixed decimal, PDMSB, PSHA | 40-bit data | 40-bit data | 40-bit data |
| | | Integer | 24-bit data | 24-bit data | — |
| | | Logic, PSHL, PMULS | — | 16-bit data | |
| | Data transfer | MOVX.W, MOVY.W, MOVS.W | | | |
| | | MOVS.L | | 32-bit data | 32-bit data |
| A0G, A1G | Data transfer | MOVS.W | Data | — | — |
| | | MOVS.L | Data | | |
| X0, X1, Y0, Y1, M0, M1 | DSP operation | Fixed decimal, PDMSB, PSHA | Sign* | 32-bit data | 32-bit data |
| | | Integer | | 16-bit data | — |
| | | Logic, PSHL, PMULS | — | | |
| | Data transfer | MOVS.W | | | |
| | | MOVS.L | | 32-bit data | 32-bit data |

Note: * The sign is extended and stored in the ALU's guard bits.

**Table 2.5 Destination Register Data Formats for DSP Instructions**

| Register | Instruction | | Guard Bits 39–32 | Register Bits 31–16 | Register Bits 15–0 |
|---|---|---|---|---|---|
| A0, A1 | DSP operation | Fixed decimal, PSHA, PMULS | (Sign extend) | 40-bit result | 40-bit result |
| | | Integer, PDMSB | (Sign extend) | 24-bit result | Clear to 0 |
| | | Logic, PSHL | Clear to 0 | 16-bit result | |
| | Data transfer | MOVS.W | Sign extend | 16-bit data | |
| | | MOVS.L | Sign extend | 32-bit data | 32-bit data |
| A0G, A1G | Data transfer | MOVS.W | Data | Not updated | Not updated |
| | | MOVS.L | Data | Not updated | |
| X0, X1, Y0, Y1, M0, M1 | DSP operation | Fixed decimal, PSHA, PMULS | — | 32-bit result | 32-bit result |
| | | Integer, logic, PDMSB, PSHL | | 16-bit result | Clear to 0 |
| | Data transfer | MOVX.W, MOVY.W, MOVS.W | | 16-bit data | |
| | | MOVS.L | | 32-bit data | 32-bit data |

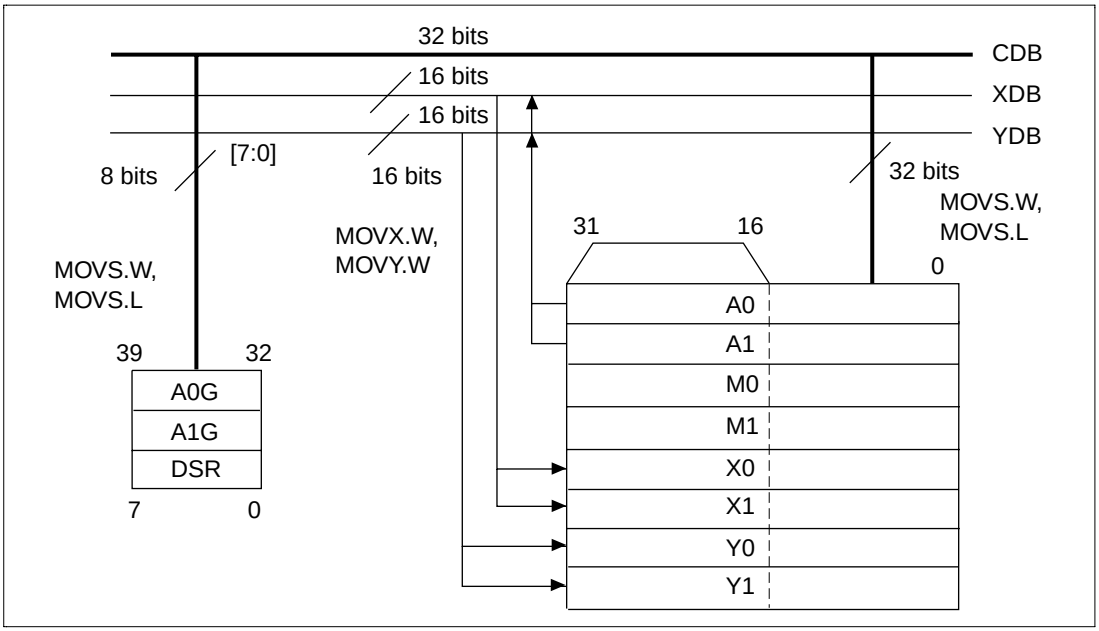



**Figure 2.8 DSP Register-Bus Relationship during Data Transfers**

## 2.3 CPU Core Instruction Features

The CPU core instructions are RISC type. The characteristics are as follows.

**16-Bit Fixed Length:** All instructions are 16 bits long, increasing program code efficiency.

**One Instruction per Cycle:** The microprocessor can execute basic instructions in one cycle using the pipeline system. One state equals 16.7 ns when operating at 60 MHz.

**Data Length:** Longword is the basic data length for all operations. Memory can be accessed in bytes, words, or longwords. Byte or word data accessed from memory is sign-extended and handled as longword data. Immediate data is sign-extended for arithmetic operations or zero-extended for logic operations. It also is handled as longword data.

**Table 2.6 Sign Extension of Word Data**

| SH7612 CPU | | Description | Example of Conventional CPU | |
|---|---|---|---|---|
| MOV.W | @(disp,PC),R1 | Data is sign-extended to 32 bits, and R1 becomes H'00001234. It is next operated upon by an ADD instruction | ADD.W | #H'1234,R0 |
| ADD | R1,R0 | | | |
| ..... | | | | |
| .DATA.W | H'1234 | | | |

Note: @(disp, PC) accesses the immediate data.

**Load-Store Architecture:** Basic operations are executed between registers. For operations that involve memory access, data is loaded to the registers and executed (load-store architecture). However, Instructions such as AND manipulating bits, are executed directly in memory.

**Delayed Branches:** Such instructions as unconditional branches are delayed branch instructions. In the case of delayed branch instructions, the branch occurs after execution of the instruction immediately following the delayed branch instruction (slot instruction). This reduces pipeline disruption during branching.

The branching operation of the delayed branch occurs after execution of the slot instruction. However, with the exception of such branch operations as register updating, execution of instructions is performed with the order of delayed branch instruction, then delayed slot instruction.

For example, even if the contents of a register storing a branch destination address are modified by a delayed slot, the branch destination address will still be the contents of the register before the modification.

**Table 2.7 Delayed Branch Instructions**

| SH7612 CPU | | Description | Example of Conventional CPU | |
|---|---|---|---|---|
| BRA | TRGET | Executes an ADD before branching to TRGET | ADD.W | R1,R0 |
| ADD | R1,R0 | | BRA | TRGET |

**Multiplication/Multiply-Accumulate Operation:** $16 \times 16 \rightarrow 32$ multiplications execute in one to three cycles, and $16 \times 16 + 64 \rightarrow 64$ multiply-accumulate operations execute in two to three cycles. $32 \times 32 \rightarrow 64$ multiplications and $32 \times 32 + 64 \rightarrow 64$ multiply-accumulate operations execute in two to four cycles.

**T Bit:** The T bit in the status register (SR) changes according to the result of a comparison, and conditional branches occur in accordance with its true or false status. The number of instructions modifying the T bit is kept to a minimum to improve the processing speed.

**Table 2.8 T Bit**

| SH7612 CPU | | Description | Example of Conventional CPU | |
|---|---|---|---|---|
| CMP/GE | R1,R0 | T bit is set when R0 ≥ R1. The program branches to TRGET0 when R0 ≥ R1. The program branches to TRGET1 when R0 < R1 | CMP.W | R1,R0 |
| BT | TRGET0 | | BGE | TRGET0 |
| BF | TRGET1 | | BLT | TRGET1 |
| ADD | #–1,R0 | T bit is not changed by ADD. T bit is set when R0 = 0. The program branches when R0 = 0 | SUB.W | #1,R0 |
| CMP/EQ | #0,R0 | | BEQ | TRGET |
| BT | TRGET | | | |

**Immediate Data:** Byte immediate data resides in instruction code. Word or longword immediate data is not input in instruction codes but is stored in a memory table. An immediate data transfer instruction (MOV) accesses the memory table using the PC relative addressing mode with displacement.

**Table 2.9 Immediate Data Accessing**

| Classification | SH7612 CPU | | Example of Conventional CPU | |
|---|---|---|---|---|
| 8-bit immediate | MOV | #H'12,R0 | MOV.B | #H'12,R0 |
| 16-bit immediate | MOV.W | @(disp,PC),R0 | MOV.W | #H'1234,R0 |
| | ..... | | | |
| | .DATA.W | H'1234 | | |
| 32-bit immediate | MOV.L | @(disp,PC),R0 | MOV.L | #H'12345678,R0 |
| | ..... | | | |
| | .DATA.L | H'12345678 | | |

Note: @(disp, PC) accesses the immediate data.

**Absolute Address:** When data is accessed by absolute address, the value already in the absolute address is placed in the memory table. Loading the immediate data when the instruction is executed transfers that value to the register and the data is accessed in the indirect register addressing mode.

**Table 2.10 Absolute Address Accessing**

| Classification | SH7612 CPU | Example of Conventional CPU |
|---|---|---|
| Absolute address | MOV.L @(disp,PC),R1<br>MOV.B @R1,R0<br>.....<br>.DATA.L H'12345678 | MOV.B @H'12345678,R0 |

**16-Bit/32-Bit Displacement:** When data is accessed by 16-bit or 32-bit displacement, the pre-existing displacement value is placed in the memory table. Loading the immediate data when the instruction is executed transfers that value to the register and the data is accessed in the indirect indexed register addressing mode.

**Table 2.11 Displacement Accessing**

| Classification | SH7612 CPU | Example of Conventional CPU |
|---|---|---|
| 16-bit displacement | MOV.W @(disp,PC),R0<br>MOV.W @(R0,R1),R2<br>.....<br>.DATA.W H'1234 | MOV.W @(H'1234,R1),R2 |

## 2.4 Instruction Formats

### 2.4.1 CPU Instruction Addressing Modes

The addressing modes and effective address calculation for instructions executed by the CPU core are listed in table 2.12.

**Table 2.12 CPU Instruction Addressing Modes and Effective Addresses**

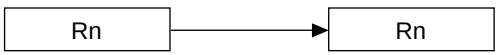
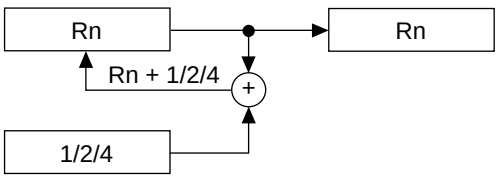
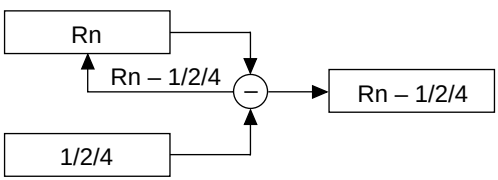
| Addressing Mode | Instruction Format | Effective Addresses Calculation | Equation |
|---|---|---|---|
| Direct register addressing | Rn | The effective address is register Rn<br>(The operand is the contents of register Rn) | — |
| Indirect register addressing | @Rn | The effective address is the content of register Rn<br>  | Rn |
| Post-increment indirect register addressing | @Rn+ | The effective address is the content of register Rn. A constant is added to the content of Rn after the instruction is executed. 1 is added for a byte operation, 2 for a word operation, and 4 for a longword operation<br>  | Rn<br>(After the instruction executes)<br>Byte: Rn + 1 → Rn<br>Word: Rn + 2 → Rn<br>Longword: Rn + 4 → Rn |
| Pre-decrement indirect register addressing | @–Rn | The effective address is the value obtained by subtracting a constant from Rn. 1 is subtracted for a byte operation, 2 for a word operation, and 4 for a longword operation<br>  | Byte: Rn – 1 → Rn<br>Word: Rn – 2 → Rn<br>Longword: Rn – 4 → Rn (Instruction executed with Rn after calculation) |

**Table 2.12 CPU Instruction Addressing Modes and Effective Addresses (cont)**

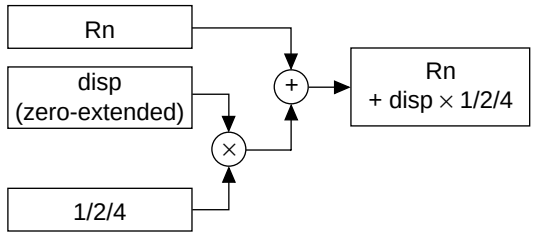
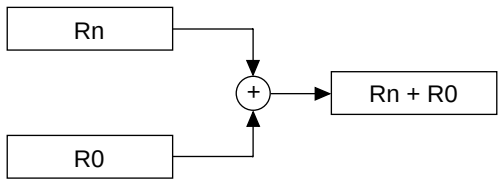
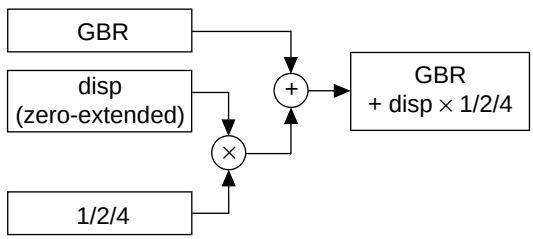
| Addressing Mode | Instruction Format | Effective Addresses Calculation | Equation |
|---|---|---|---|
| Indirect register addressing with displacement | @(disp:4, Rn) | The effective address is Rn plus a 4-bit displacement (disp). The value of disp is zero-extended, and remains the same for a byte operation, is doubled for a word operation, and is quadrupled for a longword operation   | Byte: Rn + disp<br>Word: Rn + disp × 2<br>Longword: Rn + disp × 4 |
| Indirect indexed register addressing | @(R0, Rn) | The effective address is the Rn value plus R0   | Rn + R0 |
| Indirect GBR addressing with displacement | @(disp:8, GBR) | The effective address is the GBR value plus an 8-bit displacement (disp). The value of disp is zero-extended, and remains the same for a byte operation, is doubled for a word operation, and is quadrupled for a longword operation   | Byte: GBR + disp<br>Word: GBR + disp × 2<br>Longword: GBR + disp × 4 |

**Table 2.12 CPU Instruction Addressing Modes and Effective Addresses (cont)**

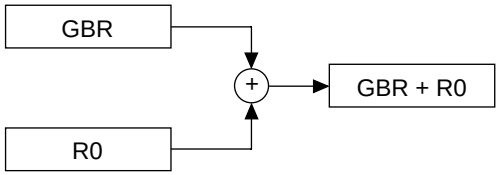
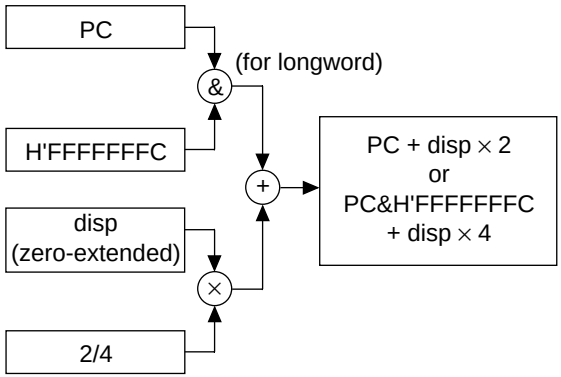
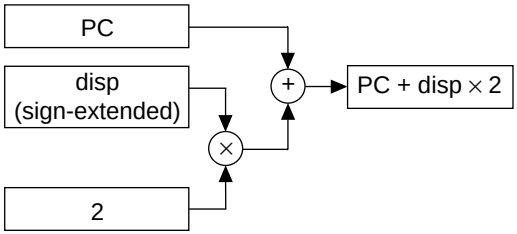
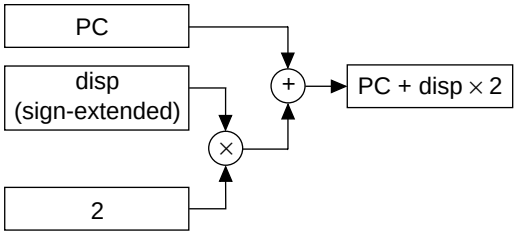
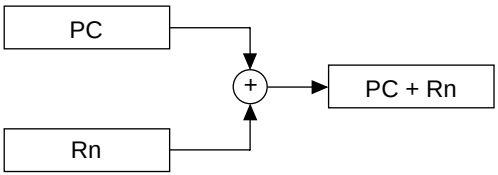
| Addressing Mode | Instruction Format | Effective Addresses Calculation | Equation |
|---|---|---|---|
| Indirect indexed GBR addressing | @(R0, GBR) | The effective address is the GBR value plus the R0<br>  | GBR + R0 |
| PC relative addressing with displacement | @(disp:8, PC) | The effective address is the PC value plus an 8-bit displacement (disp). The value of disp is zero-extended, is doubled for a word operation, and is quadrupled for a longword operation. For a longword operation, the lowest two bits of the PC value are masked<br>  | Word: PC + disp × 2<br>Longword: PC & H'FFFFFFFC + disp × 4 |

**Table 2.12 CPU Instruction Addressing Modes and Effective Addresses (cont)**

| | | | |
|---|---|---|---|
| PC relative addressing | disp:8 | The effective address is the PC value sign-extended with an 8-bit displacement (disp), doubled, and added to the PC value<br>  | PC + disp × 2 |
| | disp:12 | The effective address is the PC value sign-extended with a 12-bit displacement (disp), doubled, and added to the PC value<br>  | PC + disp × 2 |
| | Rn | The effective address is the register PC value plus Rn<br>  | PC + Rn |
| Immediate addressing | #imm:8 | The 8-bit immediate data (imm) for the TST, AND, OR, and XOR instructions are zero-extended | — |
| | #imm:8 | The 8-bit immediate data (imm) for the MOV, ADD, and CMP/EQ instructions are sign-extended | — |
| | #imm:8 | The 8-bit immediate data (imm) for the TRAPA instruction is zero-extended and is quadrupled | — |

### 2.4.2 DSP Data Addressing

There are two different kinds of memory accesses with DSP instructions. One type is with the X, Y data transfer instructions (MOVX.W, MOVY.W), and the other is with the single data transfer instructions (MOVS.W, MOVS.L). The data addressing differs between these two types of instructions. Table 2.13 shows a summary of the data transfer instructions.

**Table 2.13 Overview of Data Transfer Instructions**

| Classification | X, Y Data Transfer Processing (MOVX.W, MOVY.W) | Single Data Transfer Processing (MOVS.W, MOVS.L) |
|---|---|---|
| Address registers | Ax: R4, R5; Ay: R6, R7 | As: R2, R3, R4, R5 |
| Index registers | Ix: R8, Iy: R9 | Is: R8 |
| Addressing | Nop/Inc(+2)/index addition: post-increment | Nop/Inc(+2,+4)/index addition: post-increment |
| | — | Dec(–2,–4): pre-decrement |
| Modulo addressing | Possible | Not possible |
| Data bus | XDB, YDB | CDB |
| Data length | 16 bit (word) | 16 bit/32 bit (word/longword) |
| Bus contention | None | Yes |
| Memory | X, Y data memory | All memory spaces |
| Source registers | Dx, Dy: A0, A1 | Ds: A0/A1, M0/M1, X0/X1, Y0/Y1, A0G, A1G |
| Destination registers | Dx: X0/X1; Dy: Y0/Y1 | Ds: A0/A1, M0/M1, X0/X1, Y0/Y1, A0G, A1G |

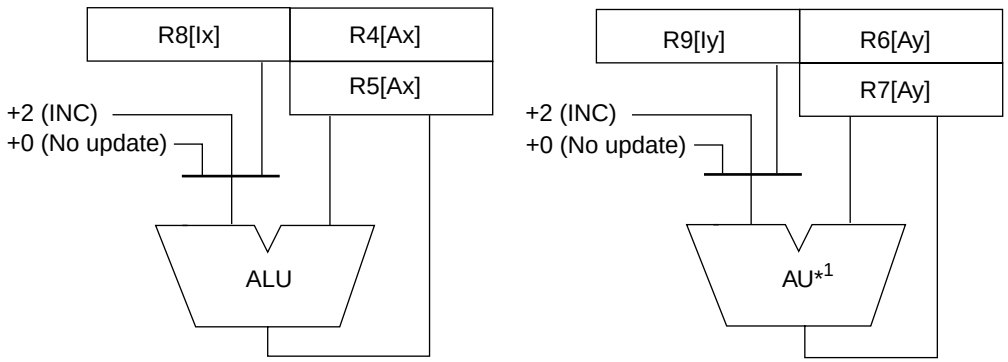
**X, Y Data Addressing:** Among the DSP instructions, the MOVX.W and MOVY.W instructions can be used to simultaneously access X, Y data memory. The DSP instructions have two address pointers for simultaneous accessing of X, Y data memory. Only pointer addressing is possible with DSP instructions; there is no immediate addressing. The address registers are divided into two; the R4, R5 registers become the X memory address register (Ax), and the R6, R7 registers become the Y memory address register (Ay). The following three types of addressing exist with X, Y data transfer instructions.

1. Non-updated address registers: The Ax, Ay registers are address pointers. They are not updated.
2. Add index registers: The Ax, Ay registers are address pointers. The Ix, Iy register values are added to them, respectively, after the data transfer (post-increment).
3. Increment address registers: The Ax, Ay registers are address pointers. The value +2 is added to each of them after the data transfer (post-increment).

Each of the address pointers has an index register. The R8 register becomes the index register (Ix) of the X memory address register (Ax), and the R9 register becomes the index register (Iy) of the Y memory address register (Ay).

The X, Y data transfer instructions are processed in word lengths. X, Y data memory is accessed in 16 bit lengths. This is why the increment processing adds 2 to the address registers. In order to decrement, set –2 in the index register and designate add index register addressing. During X, Y data addressing, only bits 1 to 15 of the address pointer are valid. Always write a 0 to bit 0 of the address pointer and the index register during X, Y data addressing.

Figure 2.9 shows the X, Y data transfer addressing. When X memory and Y memory are accessed using the X, Y bus, the upper word of Ax (R4 or R5) and Ay (R6 or R7) is ignored. The result of @Ay+ and @Ay+Iy is stored in the lower word of Ay, and the upper word retains its original value.




Notes: All three addressing methods (increment, index register addition (Ix, Iy), and no increment) are post-updating methods. To decrement the address pointer, set the index register to –2 or –4.

1. Adder added for DSP addressing.

**Figure 2.9 X, Y Data Transfer Addressing**

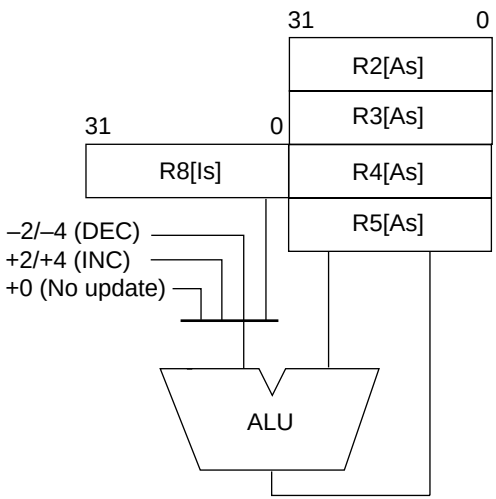
**Single Data Addressing:** Among the DSP instructions, the single data transfer instructions (MOVS.W and MOVS.L) are used to either load data into DSP registers or to store it from them. With these instructions, the registers R2 to R5 are used as address registers (As) for the single data transfers.

The four following data addressing instructions exist for single data transfer instructions.

1. Non-updated address registers: The As registers are address pointers. They are not incremented.
2. Add index registers: The As registers are address pointers. The Is register values are added to them after the data transfer (post-increment).
3. Increment address registers: The As registers are address pointers. The value +2 or +4 is added after the data transfer (post-increment).
4. Decrement address registers: The As registers are address pointers. The value –2 or –4 is added (+2 or +4 is subtracted) before the data transfer (pre-decrement).

The address pointer (As) uses the R8 register as an index register (Is).

Figure 2.10 shows the single data transfer addressing.




Note: There are four addressing methods (no update, index register addition (Is), increment, and decrement). Index register addition and increment are post-increment methods. Decrement is a pre-decrement method.

**Figure 2.10 Single Data Transfer Addressing**

**Modulo Addressing:** The chip has a modulo addressing mode, just as other DSPs do. Address registers are updated in the same manner as with other modes. When the address pointer value becomes the same as a previously established modulo end address, the address pointer becomes the modulo start address.

Modulo addressing is valid only with X, Y data transfer instructions (MOVX.W, MOVY.W). When the DMX bit of the SR register is set, the X address register enters modulo addressing mode; when the DMY bit of the SR register is set, the Y address register does so. Modulo addressing is valid only for either the X or the Y address register; it is not possible to make them both modulo addressing mode at the same time. Therefore, do not simultaneously set the DMX and DMY. If they happen to be set at the same time, only the DMY side is valid.

The MOD register is used to designate the start and end addresses of the modulo address area; it stores the MS (modulo start) and ME (modulo end). An example of MOD register (MS, ME) usage is indicated below.

```
            MOV.L ModAddr,Rn;             Rn=ModEnd, ModStart
            LDC Rn,MOD;                   ME=ModEnd, MS=ModStart
ModAddr:    .DATA.W    mEnd;              ModEnd
            .DATA.W    mStart;            ModStart

ModStart:   .DATA
              :
ModEnd:     .DATA
```

Designate the start and end addresses in MS and ME, and then set the DMX or DMY bit to 1. The contents of the address register are compared with ME. If they match ME, the start address MS is stored in the address register. The lower 16 bits of the address register are compared with ME. The maximum modulo size is 64 kbytes. This is sufficient for X, Y data memory accesses. Figure 2.11 shows a block diagram of modulo addressing.

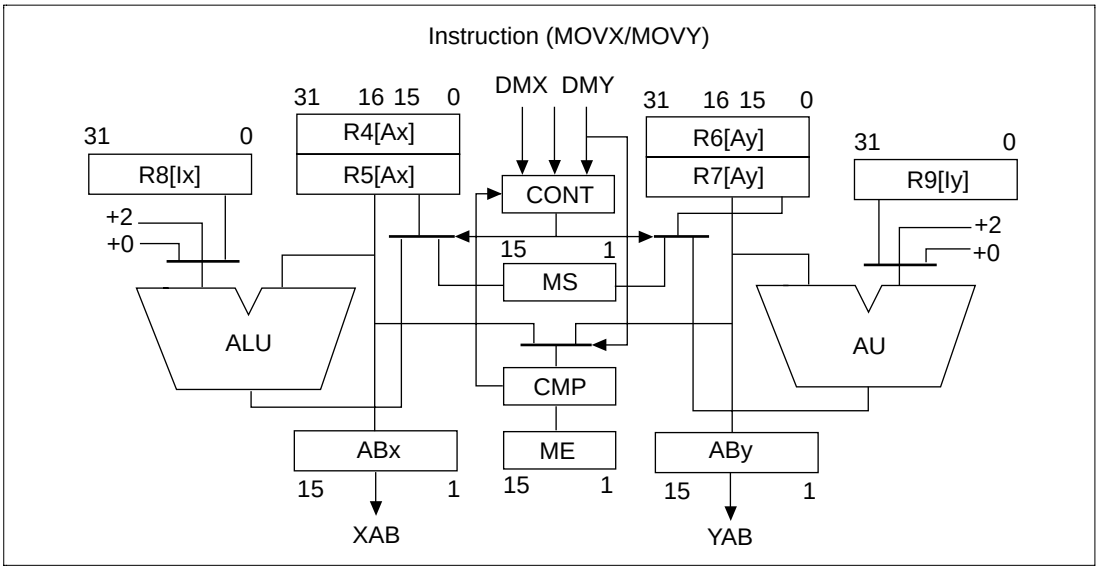



**Figure 2.11 Modulo Addressing**

An example of modulo addressing is indicated below:

```
MS=H'E008; ME=H'E00C; R4=H'1000E008;
DMX=1; DMY=0;
```
(sets modulo addressing for address register Ax (R4, R5))

The R4 register changes as follows due to the above settings.

```
        R4: H'1000E008
Inc.    R4: H'1000E00A
Inc.    R4: H'1000E00C
Inc.    R4: H'1000E008
```
(becomes the modulo start address because the modulo end address occurred)

Data is placed so that the upper 16 bits of the modulo start and end addresses become identical. This is so because the modulo start address replaces only the lower 15 bits of the address register, excepting bit 0.

Note: When using add index with DSP data addressing, there are cases where the value is exceeded without the address pointer matching the ME. In such cases, the address pointer does not return to the modulo start address. Bit 0 is disregarded not only for modulo addressing, but also during X, Y data addressing, so always write 0 to the 0 bits of the address pointer, index register, MS, and ME.

**DSP Addressing Operation:** The DSP addressing operation in the item stage (EX) of the pipeline, including modulo addressing, is indicated below.

```
if ( Operation is MOVX.W MOVY.W ) {
    ABx=Ax; ABy=Ay;
    /* memory access cycle uses ABx and ABy. The addresses to be used have
not been updated */

    /* Ax is one of R4,5 */
    if ( DMX==0 || DMX==1 && DMY==1 )} Ax=Ax+(+2 or R8[Ix} or +0);
    /* Inc,Index,Not-Update */
    else if (!not-update) Ax=modulo( Ax, (+2 or R8[Ix]) );

    /* Ay is one of R6,7 */
    if ( DMY==0 ) Ay=Ay+(+2 or R9[Iy] or +0; /* Inc,Index,Not-Update */
    else if (! not-update) Ay=modulo( Ay, (+2 or R9[Iy]) );
}
else if ( Operation is MOVS.W or MOVS.L ) {
    if ( Addressing is Nop, Inc, Add-index-reg ) {
        MAB=As;
        /* memory access cycle uses MAB. The address to be used has not been
updated */
        /* As is one of R2–5 */
        As=As+(+2 or +4 or R8[Is] or +0); /* Inc.Index,Not-Update */
    else { /* Decrement, Pre-update */
    /* As is one of R2–5 */
    As=As+(–2 or –4);
    MAB=As;
    /* memory access cycle uses MAB. The address to be used has been updated
*/
}

/* The value to be added to the address register depends on addressing
operations.
For example, (+2 or R8[Ix] or +0) means that
        +2:         if operation is increment
        R8[Ix}:     if operation is add-index-reg
        +0:         if operation is not-update
*/
```

```
function modulo ( AddrReg, Index ) {
    if ( AdrReg[15:0]==ME ) AdrReg[15:0]==MS;
    else AdrReg=AdrReg+Index;
    return AddrReg;
}
```

### 2.4.3 Instruction Formats for CPU Instructions

The instruction format of instructions executed by the CPU core and the meanings of the source and destination operands are indicated below. The meaning of the operand depends on the instruction code. The symbols are used as follows:

- xxxx: Instruction code
- mmmm: Source register
- nnnn: Destination register
- iiii: Immediate data
- dddd: Displacement

**Table 2.14 Instruction Formats for CPU Instructions**

| Instruction Formats | Source Operand | Destination Operand | Example |
|---|---|---|---|
| 0 format<br>15 0<br>xxxx xxxx xxxx xxxx | — | — | NOP |
| n format<br>15 0<br>xxxx nnnn xxxx xxxx | — | nnnn: Direct register | MOVT Rn |
| | Control register or system register | nnnn: Direct register | STS MACH,Rn |
| | Control register or system register | nnnn: Indirect pre-decrement register | STC.L SR,@-Rn |
| m format<br>15 0<br>xxxx mmmm xxxx xxxx | mmmm: Direct register | Control register or system register | LDC Rm,SR |
| | mmmm: Indirect post-increment register | Control register or system register | LDC.L @Rm+,SR |
| | mmmm: Indirect register | — | JMP @Rm |
| | mmmm: PC relative using Rm | — | BRAF Rm |

**Table 2.14 Instruction Formats for CPU Instructions (cont)**

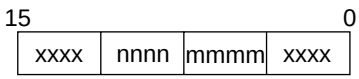
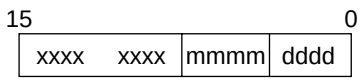
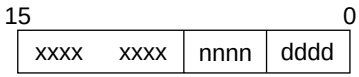
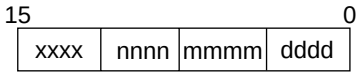
| Instruction Formats | Source Operand | Destination Operand | Example |
|---|---|---|---|
| nm format<br>  | mmmm: Direct register | nnnn: Direct register | ADD Rm,Rn |
| | mmmm: Direct register | nnnn: Indirect register | MOV.L Rm,@Rn |
| | mmmm: Indirect post-increment register (multiply/ accumulate)<br>nnnn: Indirect post-increment register (multiply/ accumulate)* | MACH, MACL | MAC.W @Rm+,@Rn+ |
| | mmmm: Indirect post-increment register | nnnn: Direct register | MOV.L @Rm+,Rn |
| | mmmm: Direct register | nnnn: Indirect pre-decrement register | MOV.L Rm,@-Rn |
| | mmmm: Direct register | nnnn: Indirect indexed register | MOV.L Rm,@(R0,Rn) |
| md format<br>  | mmmmdddd: indirect register with displacement | R0 (Direct register) | MOV.B @(disp,Rm),R0 |
| nd4 format<br>  | R0 (Direct register) | nnnndddd: Indirect register with displacement | MOV.B R0,@(disp,Rn) |
| nmd format<br>  | mmmm: Direct register | nnnndddd: Indirect register with displacement | MOV.L Rm,@(disp,Rn) |
| | mmmmdddd: Indirect register with displacement | nnnn: Direct register | MOV.L @(disp,Rm),Rn |

**Table 2.14 Instruction Formats for CPU Instructions (cont)**

| Instruction Formats | Source Operand | Destination Operand | Example |
|---|---|---|---|
| d format<br>15 0<br>xxxx xxxx dddd dddd | dddddddd: Indirect GBR with displacement | R0 (Direct register) | MOV.L @(disp,GBR),R0 |
| | R0(Direct register) | dddddddd: Indirect GBR with displacement | MOV.L R0,@(disp,GBR) |
| | dddddddd: PC relative with displacement | R0 (Direct register) | MOVA @(disp,PC),R0 |
| | dddddddd: PC relative | — | BF label |
| d12 format<br>15 0<br>xxxx dddd dddd dddd | dddddddddddd: PC relative | — | BRA label<br>(label=disp+PC) |
| nd8 format<br>15 0<br>xxxx nnnn dddd dddd | dddddddd: PC relative with displacement | nnnn: Direct register | MOV.L @(disp,PC),Rn |
| i format<br>15 0<br>xxxx xxxx iiii iiii | iiiiiiii: Immediate | Indirect indexed GBR | AND.B #imm,@(R0,GBR) |
| | iiiiiiii: Immediate | R0 (Direct register) | AND #imm,R0 |
| | iiiiiiii: Immediate | — | TRAPA #imm |
| ni format<br>15 0<br>xxxx nnnn iiii iiii | iiiiiiii: Immediate | nnnn: Direct register | ADD #imm,Rn |

Note: In multiply/accumulate instructions, nnnn is the source register.

### 2.4.4 Instruction Formats for DSP Instructions

New instructions have been added for digital signal processing. The new instructions are divided into the two following types.

1. Memory and DSP register double, single data transfer instructions (16 bit length)
2. Parallel processing instructions processed by the DSP unit (32 bit length)

Figure 2.12 shows each of the instruction formats.

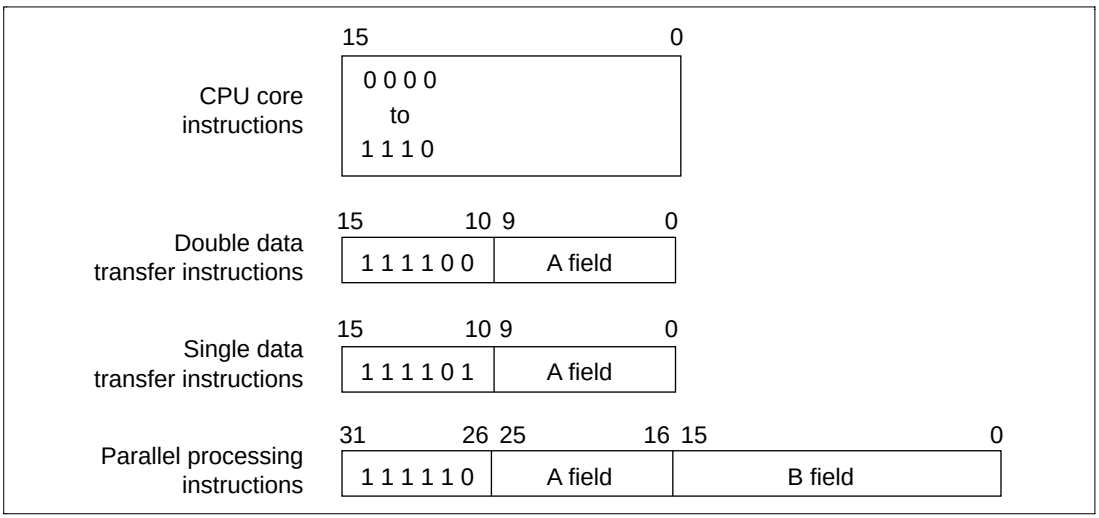



**Figure 2.12 Instruction Formats for DSP Instructions**

**Double, Single Data Transfer Instructions:** Table 2.15 indicates the data formats for double data transfer instructions, and table 2.16 indicates the data formats for single data transfer instructions.

**Table 2.15 Instruction Formats for Double Data Transfers**

| Category | Mnemonic | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| X memory data transfers | NOPX | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| | MOVX.W | @Ax,Dx | | | | | | | Ax | |
| | MOVX.W | @Ax+,Dx | | | | | | | | |
| | MOVX.W | @Ax+Ix,Dx | | | | | | | | |
| | MOVX.W | Da,@Ax | | | | | | | | |
| | MOVX.W | Da,@Ax+ | | | | | | | | |
| | MOVX.W | Da,@Ax+Ix | | | | | | | | |
| Y memory data transfers | NOPY | | 1 | 1 | 1 | 1 | 0 | 0 | | 0 |
| | MOVY.W | @Ay,Dy | | | | | | | | Ay |
| | MOVY.W | @Ay+,Dy | | | | | | | | |
| | MOVY.W | @Ay+Iy,Dy | | | | | | | | |
| | MOVY.W | Da,@Ay | | | | | | | | |
| | MOVY.W | Da,@Ay+ | | | | | | | | |
| | MOVY.W | Da,@Ay+Iy | | | | | | | | |

**Table 2.15 Instruction Formats for Double Data Transfers (cont)**

| Category | Mnemonic | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| X memory data transfers | NOPX | | 0 | | 0 | | 0 | 0 | | |
| | MOVX.W | @Ax,Dx | Dx | | 0 | | 0 | 1 | | |
| | MOVX.W | @Ax+,Dx | | | | | 1 | 0 | | |
| | MOVX.W | @Ax+Ix,Dx | | | | | 1 | 1 | | |
| | MOVX.W | Da,@Ax | Da | | 1 | | 0 | 1 | | |
| | MOVX.W | Da,@Ax+ | | | | | 1 | 0 | | |
| | MOVX.W | Da,@Ax+Ix | | | | | 1 | 1 | | |
| Y memory data transfers | NOPY | | | 0 | | 0 | | | 0 | 0 |
| | MOVY.W | @Ay,Dy | | Dy | | 0 | | | 0 | 1 |
| | MOVY.W | @Ay+,Dy | | | | | | | 1 | 0 |
| | MOVY.W | @Ay+Iy,Dy | | | | | | | 1 | 1 |
| | MOVY.W | Da,@Ay | | Da | | 1 | | | 0 | 1 |
| | MOVY.W | Da,@Ay+ | | | | | | | 1 | 0 |
| | MOVY.W | Da,@Ay+Iy | | | | | | | 1 | 1 |

Ax: 0=R4, 1=R5 Ay: 0=R6, 1=R7 Dx: 0=X0, 1=X1 Dy: 0=Y0, 1=Y1 Da: 0=A0, 1=A1

**Table 2.16 Instruction Formats for Single Data Transfers**

| Category | Mnemonic | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| Single data transfer | MOVS.W | @–As,Ds | 1 | 1 | 1 | 1 | 0 | 1 | As | |
| | MOVS.W | @As,Ds | | | | | | | 0: R4 | |
| | MOVS.W | @As+,Ds | | | | | | | 1: R5 | |
| | MOVS.W | @As+Ix,Ds | | | | | | | 2: R2 | |
| | MOVS.W | Ds,@-As | | | | | | | 3: R3 | |
| | MOVS.W | Ds,@As | | | | | | | | |
| | MOVS.W | Ds,@As+ | | | | | | | | |
| | MOVS.W | Ds,@As+Ix | | | | | | | | |
| | MOVS.L | @–As,Ds | | | | | | | | |
| | MOVS.L | @As,Ds | | | | | | | | |
| | MOVS.L | @As+,Ds | | | | | | | | |
| | MOVS.L | @As+Ix,Ds | | | | | | | | |
| | MOVS.L | Ds,@-As | | | | | | | | |
| | MOVS.L | Ds,@As | | | | | | | | |
| | MOVS.L | Ds,@As+ | | | | | | | | |
| | MOVS.L | Ds,@As+Ix | | | | | | | | |

**Table 2.16 Instruction Formats for Single Data Transfers (cont)**

| Category | Mnemonic | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Single data transfer | MOVS.W | @–As,Ds | | Ds | | 0: (*) | 0 | 0 | 0 | 0 |
| | MOVS.W | @As,Ds | | | | 1: (*) | 0 | 1 | | |
| | MOVS.W | @As+,Ds | | | | 2: (*) | 1 | 0 | | |
| | MOVS.W | @As+Ix,Ds | | | | 3: (*) | 1 | 1 | | |
| | MOVS.W | Ds,@-As | | | | 4: (*) | 0 | 0 | 0 | 1 |
| | MOVS.W | Ds,@As | | | | 5: A1 | 0 | 1 | | |
| | MOVS.W | Ds,@As+ | | | | 6: (*) | 1 | 0 | | |
| | MOVS.W | Ds,@As+Ix | | | | 7: A0 | 1 | 1 | | |
| | MOVS.L | @–As,Ds | | | | 8: X0 | 0 | 0 | 1 | 0 |
| | MOVS.L | @As,Ds | | | | 9: X1 | 0 | 1 | | |
| | MOVS.L | @As+,Ds | | | | A: Y0 | 1 | 0 | | |
| | MOVS.L | @As+Ix,Ds | | | | B: Y1 | 1 | 1 | | |
| | MOVS.L | Ds,@-As | | | | C: M0 | 0 | 0 | 1 | 1 |
| | MOVS.L | Ds,@As | | | | D: A1G | 0 | 1 | | |
| | MOVS.L | Ds,@As+ | | | | E: M1 | 1 | 0 | | |
| | MOVS.L | Ds,@As+Ix | | | | F: A0G | 1 | 1 | | |

Note: System reserved code

**Parallel Processing Instructions:** The parallel processing instructions allow for more efficient execution of digital signal processing using the DSP unit. They are 32 bit length, allowing simultaneously in parallel four processes, ALU operations, multiplications or 2 data transfers.

The parallel processing instructions are divided into A fields and B fields. The A field defines data transfer instructions; the B field defines ALU operation instructions and multiplication instructions. These instructions can be defined independently, the processes can be independent, and furthermore, they can be executed simultaneously in parallel. Table 2.17 indicates the A field parallel data transfer instructions, and table 2.18 indicates the B field ALU operation instructions and multiplication instructions.

**Table 2.17 Field A Parallel Data Transfer Instructions**

| Category | Mnemonic | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| X memory data transfers | NOPX | | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | 0 |
| | MOVX.W | @Ax,Dx | | | | | | | Ax | | Dx |
| | MOVX.W | @Ax+,Dx | | | | | | | | | |
| | MOVX.W | @Ax+Ix,Dx | | | | | | | | | |
| | MOVX.W | Da,@Ax | | | | | | | | | Da |
| | MOVX.W | Da,@Ax+ | | | | | | | | | |
| | MOVX.W | Da,@Ax+Ix | | | | | | | | | |
| Y memory data transfers | NOPY | | | | | | | | | 0 | |
| | MOVY.W | @Ay,Dy | | | | | | | | Ay | |
| | MOVY.W | @Ay+,Dy | | | | | | | | | |
| | MOVY.W | @Ay+Iy,Dy | | | | | | | | | |
| | MOVY.W | Da,@Ay | | | | | | | | | |
| | MOVY.W | Da,@Ay+ | | | | | | | | | |
| | MOVY.W | Da,@Ay+Iy | | | | | | | | | |

**Table 2.17 Field A Parallel Data Transfer Instructions (cont)**

| Category | Mnemonic | | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15–0 |
|---|---|---|---|---|---|---|---|---|---|---|
| X memory data transfers | NOPX | | | 0 | | 0 | 0 | | | Field B |
| | MOVX.W | @Ax,Dx | | 0 | | 0 | 1 | | | |
| | MOVX.W | @Ax+,Dx | | | | 1 | 0 | | | |
| | MOVX.W | @Ax+Ix,Dx | | | | 1 | 1 | | | |
| | MOVX.W | Da,@Ax | | 1 | | 0 | 1 | | | |
| | MOVX.W | Da,@Ax+ | | | | 1 | 0 | | | |
| | MOVX.W | Da,@Ax+Ix | | | | 1 | 1 | | | |
| Y memory data transfers | NOPY | | 0 | | 0 | | | 0 | 0 | |
| | MOVY.W | @Ay,Dy | Dy | | 0 | | | 0 | 1 | |
| | MOVY.W | @Ay+,Dy | | | | | | 1 | 0 | |
| | MOVY.W | @Ay+Iy,Dy | | | | | | 1 | 1 | |
| | MOVY.W | Da,@Ay | Da | | 1 | | | 0 | 1 | |
| | MOVY.W | Da,@Ay+ | | | | | | 1 | 0 | |
| | MOVY.W | Da,@Ay+Iy | | | | | | 1 | 1 | |

Ax: 0=R4, 1=R5 Ay: 0=R6, 1=R7 Dx: 0=X0, 1=X1 Dy: 0=Y0, 1=Y1 Da: 0=A0, 1=A1

**Table 2.18 Field B ALU Operation Instructions, Multiplication Instructions**

| Category | Mnemonic | 31–27 | 26 | 25–16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Imm. shift | PSHL #imm, Dz | 1 | 0 | Field A | 0 | 0 | 0 | 0 | 0 | –16 ≤ imm ≤ +16 | | | | | | | Dz | | | |
| | PSHA #imm, Dz | | | | 0 | 0 | 0 | 1 | 0 | – 32 ≤ imm ≤ +32 | | | | | | | | | | |
| | Reserved | | | | 0 | 0 | 0 | | 1 | | | | | | | | | | | |
| | | | | | 0 | 0 | 1 | | | | | | | | | | | | | |
| Six operand parallel instruction | PMULS Se, Sf, Dg | | | | 0 | 1 | 0 | 0 | Se | | Sf | | Sx | | Sy | | Dg | | Du | |
| | Reserved | | | | 0 | 1 | 0 | 1 | 0:X0<br>1:X1 | | 0:Y0<br>1:Y1 | | 0:X0<br>1:X1 | | 0:Y0<br>1:Y1 | | 0:M0<br>1:M1 | | 0:X0<br>1:Y0 | |
| | PSUB Sx, Sy, Du<br>PMULS Se, Sf, Dg | | | | 0 | 1 | 1 | 0 | 2:Y0<br>3:A1 | | 2:X0<br>3:A1 | | 2:A0<br>3:A1 | | 2:M0<br>3:M1 | | 2:A0<br>3:A1 | | 2:A0<br>3:A1 | |
| | PADD Sx, Sy, Du<br>PMULS Se, Sf, Dg | | | | 0 | 1 | 1 | 1 | | | | | | | | | | | | |
| Three operand instructions | Reserved | | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | Dz | | | |
| | | | | | | | | 1 | | | | | | | | | | | | |
| | PSUBC Sx, Sy, Dz | | | | | | 1 | 0 | | | | | | | | | 0: (*1) | | | |
| | PADDC Sx, Sy, Dz | | | | | | 1 | 1 | | | | | | | | | 1: (*1) | | | |
| | PCMP Sx, Sy | | | | | | 0 | 0 | 0 | 1 | | | | | | | 2: (*1) | | | |
| | Reserved | | | | | | 0 | 1 | | | | | | | | | 3: (*1) | | | |
| | | | | | | | 1 | 0 | | | | | | | | | 4: (*1) | | | |
| | | | | | | | 1 | 1 | | | | | | | | | 5: A1 | | | |
| | PABS Sx, Dz | | | | | | 0 | 0 | 1 | 0 | | | | | | | 6: (*1) | | | |
| | PRND Sx, Dz | | | | | | 0 | 1 | | | | | | | | | 7: A0 | | | |
| | PABS Sy, Dz | | | | | | 1 | 0 | | | | | | | | | 8: X0 | | | |
| | PRND Sy, Dz | | | | | | 1 | 1 | | | | | | | | | 9: X1 | | | |
| | Reserved | | | | | | 0 | 0 | 1 | 1 | | | | | | | A: Y0 | | | |
| | | | | | | | 0 | 1 | | | | | | | | | B: Y1 | | | |
| | | | | | | | 1 | 0 | | | | | | | | | C: M0 | | | |
| | | | | | | | 1 | 1 | | | | | | | | | D: (*1) | | | |
| | | | | | | | | | | | | | | | | | E: M1 | | | |
| | | | | | | | | | | | | | | | | | F: (*1) | | | |

**Table 2.18 Field B ALU Operation Instructions, Multiplication Instructions (cont)**

| Category | Mnemonic | 31–27 | 26 | 25–16 | 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 1 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Conditional three operand instructions | (if cc) PSHL Sx, Sy, Dz | 1 | 0 | Field A | | 0 0 | 0 0 | if cc | | | |
| | (if cc) PSHA Sx, Sy, Dz | | | | | 0 1 | | | | | |
| | (if cc) PSUB Sx, Sy, Dz | | | | | 1 0 | | | | | |
| | (if cc) PADD Sx, Sy, Dz | | | | | 1 1 | | 01: Uncon-dition | | | |
| | Reserved | | | | | 0 0 | 0 1 | | | | |
| | (if cc) PAND Sx, Sy, Dz | | | | | 0 1 | | | | | |
| | (if cc) PXOR Sx, Sy, Dz | | | | | 1 0 | | | | | |
| | (if cc) POR Sx, Sy, Dz | | | | | 1 1 | | 10:DCT | | | |
| | (if cc) PDEC Sx, Dz | | | | | 0 0 | 1 0 | | | | |
| | (if cc) PINC Sx, Dz | | | | | 0 1 | | | | | |
| | (if cc) PDEC Sy, Dz | | | | | 1 0 | | | | | |
| | (if cc) PINC Sy, Dz | | | | | 1 1 | | 11:DCF | | | |
| | (if cc) PCLR Dz | | | | | 0 0 | 1 1 | | | | |
| | (if cc) PDMSB Sx, Dz | | | | | 0 1 | | | | | |
| | Reserved | | | | | 1 0 | | | | | |
| | (if cc) PDMSB Sy, Dz | | | | | 1 1 | | | | | |
| | (if cc) PNEG Sx, Dz | | | | 1 1 | 0 0 | 1 0 | | | | |
| | (if cc) PCOPY Sx, Dz | | | | | 0 1 | | | | | |
| | (if cc) PNEG Sy, Dz | | | | | 1 0 | | | | | |
| | (if cc) PCOPY Sy, Dz | | | | | 1 1 | | | | | |
| | Reserved | | | | | | | 0 0 | | | |
| | (if cc) PSTS MACH, Dz | | | | | 0 0 | 1 1 | if cc | | | |
| | (if cc) PSTS MACL, Dz | | | | | 0 1 | | | | | |
| | (if cc) PLDS Dz, MACH | | | | | 1 0 | | | | | |
| | (if cc) PLDS Dz, MACL | | | | | 1 1 | | | | | |
| | Reserved*2 | | | | | | | 0 0 | | | |
| | | | | | | | 0 * | | | | |
| | Reserved | 1 | | | | | | | | | |

Notes: 1. System reserved code
2. (if cc): DCT (DC bit true), DCF (DC bit false), or none (unconditional instruction)

## 2.5 Instruction Set

The instructions are divided into three groups: CPU instructions executed by the CPU core, DSP data transfer instructions executed by the DSP unit, and DSP operation instructions. There are a number of CPU instructions for supporting the DSP functions. The instruction set is explained below in terms of each of the three groups.

### 2.5.1 CPU Instruction Set

Table 2.19 lists the CPU instructions by classification.

**Table 2.19 Classification of CPU Instructions**

| Classification | Types | Operation Code | Function | No. of Instructions |
|---|---|---|---|---|
| Data transfer | 5 | MOV | Data transfer, immediate data transfer, peripheral module data transfer, structure data transfer | 39 |
| | | MOVA | Effective address transfer | |
| | | MOVT | T bit transfer | |
| | | SWAP | Swap of upper and lower bytes | |
| | | XTRCT | Extraction of the middle of registers connected | |
| Arithmetic operations | 21 | ADD | Binary addition | 33 |
| | | ADDC | Binary addition with carry | |
| | | ADDV | Binary addition with overflow | |
| | | CMP/cond | Comparison | |
| | | DIV1 | Division | |
| | | DIV0S | Initialization of signed division | |
| | | DIV0U | Initialization of unsigned division | |
| | | DMULS | Signed double-length multiplication | |
| | | DMULU | Unsigned double-length multiplication | |
| | | DT | Decrement and test | |
| | | EXTS | Sign extension | |
| | | EXTU | Zero extension | |
| | | MAC | Multiply/accumulate, double-length multiply/accumulate operation | |
| | | MUL | Double-length multiply operation | |
| | | MULS | Signed multiplication | |
| | | MULU | Unsigned multiplication | |
| | | NEG | Negation | |
| | | NEGC | Negation with borrow | |
| | | SUB | Binary subtraction | |
| | | SUBC | Binary subtraction with borrow | |
| | | SUBV | Binary subtraction with underflow | |

**Table 2.19 Classification of CPU Instructions (cont)**

| Classification | Types | Operation Code | Function | No. of Instructions |
|---|---|---|---|---|
| Logic operations | 6 | AND | Logical AND | 14 |
| | | NOT | Bit inversion | |
| | | OR | Logical OR | |
| | | TAS | Memory test and bit set | |
| | | TST | Logical AND and T bit set | |
| | | XOR | Exclusive OR | |
| Shift | 10 | ROTCL | One-bit left rotation with T bit | 14 |
| | | ROTCR | One-bit right rotation with T bit | |
| | | ROTL | One-bit left rotation | |
| | | ROTR | One-bit right rotation | |
| | | SHAL | One-bit arithmetic left shift | |
| | | SHAR | One-bit arithmetic right shift | |
| | | SHLL | One-bit logical left shift | |
| | | SHLLn | n-bit logical left shift | |
| | | SHLR | One-bit logical right shift | |
| | | SHLRn | n-bit logical right shift | |
| Branch | 9 | BF | Conditional branch, conditional branch with delay (Branch when T = 0) | 11 |
| | | BT | Conditional branch, conditional branch with delay (Branch when T = 1) | |
| | | BRA | Unconditional branch | |
| | | BRAF | Unconditional branch | |
| | | BSR | Branch to subroutine procedure | |
| | | BSRF | Branch to subroutine procedure | |
| | | JMP | Unconditional branch | |
| | | JSR | Branch to subroutine procedure | |
| | | RTS | Return from subroutine procedure | |

**Table 2.19 Classification of CPU Instructions (cont)**

| Classification | Types | Operation Code | Function | No. of Instructions |
|---|---|---|---|---|
| System control | 14 | CLRMAC | MAC register clear | 71 |
| | | CLRT | T bit clear | |
| | | LDC | Load to control register | |
| | | LDRE | Load to repeat end register | |
| | | LDRS | Load to repeat start register | |
| | | LDS | Load to system register | |
| | | NOP | No operation | |
| | | RTE | Return from exception processing | |
| | | SETRC | Repeat count setting | |
| | | SETT | T bit set | |
| | | SLEEP | Shift into power-down mode | |
| | | STC | Storing control register data | |
| | | STS | Storing system register data | |
| | | TRAPA | Trap exception handling | |
| Total: | 65 | | | 182 |

The instruction codes, operation, and execution states of the CPU instructions are listed by classification with the formats listed in below.

| Instruction | Instruction Code | Operation | Execution Cycles | T Bit |
|---|---|---|---|---|
| Indicated by mnemonic | Indicated in MSB ↔ LSB order | Indicates summary of operation | Value when no wait states are inserted*1 | Value of T bit after instruction is executed |
| Explanation of Symbols | Explanation of Symbols | Explanation of Symbols | | |
| OP.Sz SRC, DEST<br>OP: Operation code<br>Sz: Size<br>SRC: Source<br>DEST: Destination | mmmm: Source register<br>nnnn: Destination register<br>0000: R0<br>0001: R1<br>.....<br>1111: R15 | →, ←: Transfer direction<br>(xx): Memory operand<br>M/Q/T: Flag bits in the SR<br>&: Logical AND of each bit<br>\|: Logical OR of each bit<br>^: Exclusive OR of each bit<br>~: Logical NOT of each bit<br><<n: n-bit left shift<br>>>n: n-bit right shift | | Explanation of Symbols<br>—: No change |
| Rm: Source register | | | | |
| Rn: Destination register | iiii: Immediate data | | | |
| imm: Immediate data | dddd: Displacement | | | |
| disp: Displacement*2 | | | | |

Notes: 1. Instruction execution cycles: The execution cycles shown in the table are minimums. The actual number of cycles may be increased when (1) contention occurs between instruction fetches and data access, or (2) when the destination register of the load instruction (memory → register) and the register used by the next instruction are the same.

2. Depending on the instruction's operand size, scaling is ×1, ×2, or ×4. For details, see the *SH-1/SH-2/SH-DSP Programming Manual*.

**Table 2.20 Data Transfer Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| MOV | #imm,Rn | 1110nnnniiiiiiii | imm → Sign extension → Rn | 1 | — |
| MOV.W | @(disp,PC),Rn | 1001nnnndddddddd | (disp × 2 + PC) → Sign extension → Rn | 1 | — |
| MOV.L | @(disp,PC),Rn | 1101nnnndddddddd | (disp × 4 + PC) → Rn | 1 | — |
| MOV | Rm,Rn | 0110nnnnmmmm0011 | Rm → Rn | 1 | — |
| MOV.B | Rm,@Rn | 0010nnnnmmmm0000 | Rm → (Rn) | 1 | — |
| MOV.W | Rm,@Rn | 0010nnnnmmmm0001 | Rm → (Rn) | 1 | — |
| MOV.L | Rm,@Rn | 0010nnnnmmmm0010 | Rm → (Rn) | 1 | — |
| MOV.B | @Rm,Rn | 0110nnnnmmmm0000 | (Rm) → Sign extension → Rn | 1 | — |
| MOV.W | @Rm,Rn | 0110nnnnmmmm0001 | (Rm) → Sign extension → Rn | 1 | — |
| MOV.L | @Rm,Rn | 0110nnnnmmmm0010 | (Rm) → Rn | 1 | — |
| MOV.B | Rm,@–Rn | 0010nnnnmmmm0100 | Rn–1 → Rn, Rm → (Rn) | 1 | — |
| MOV.W | Rm,@–Rn | 0010nnnnmmmm0101 | Rn–2 → Rn, Rm → (Rn) | 1 | — |
| MOV.L | Rm,@–Rn | 0010nnnnmmmm0110 | Rn–4 → Rn, Rm → (Rn) | 1 | — |
| MOV.B | @Rm+,Rn | 0110nnnnmmmm0100 | (Rm) → Sign extension → Rn,Rm + 1 → Rm | 1 | — |
| MOV.W | @Rm+,Rn | 0110nnnnmmmm0101 | (Rm) → Sign extension → Rn,Rm + 2 → Rm | 1 | — |
| MOV.L | @Rm+,Rn | 0110nnnnmmmm0110 | (Rm) → Rn,Rm + 4 → Rm | 1 | — |
| MOV.B | R0,@(disp,Rn) | 10000000nnnndddd | R0 → (disp + Rn) | 1 | — |
| MOV.W | R0,@(disp,Rn) | 10000001nnnndddd | R0 → (disp × 2 + Rn) | 1 | — |
| MOV.L | Rm,@(disp,Rn) | 0001nnnnmmmmdddd | Rm → (disp × 4 + Rn) | 1 | — |
| MOV.B | @(disp,Rm),R0 | 10000100mmmmdddd | (disp + Rm) → Sign extension → R0 | 1 | — |
| MOV.W | @(disp,Rm),R0 | 10000101mmmmdddd | (disp × 2 + Rm) → Sign extension → R0 | 1 | — |
| MOV.L | @(disp,Rm),Rn | 0101nnnnmmmmdddd | (disp × 4 + Rm) → Rn | 1 | — |
| MOV.B | Rm,@(R0,Rn) | 0000nnnnmmmm0100 | Rm → (R0 + Rn) | 1 | — |
| MOV.W | Rm,@(R0,Rn) | 0000nnnnmmmm0101 | Rm → (R0 + Rn) | 1 | — |

**Table 2.20 Data Transfer Instructions (cont)**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| MOV.L | Rm,@(R0,Rn) | 0000nnnnmmmm0110 | Rm → (R0 + Rn) | 1 | — |
| MOV.B | @(R0,Rm),Rn | 0000nnnnmmmm1100 | (R0 + Rm) → Sign extension → Rn | 1 | — |
| MOV.W | @(R0,Rm),Rn | 0000nnnnmmmm1101 | (R0 + Rm) → Sign extension → Rn | 1 | — |
| MOV.L | @(R0,Rm),Rn | 0000nnnnmmmm1110 | (R0 + Rm) → Rn | 1 | — |
| MOV.B | R0,@(disp,GBR) | 11000000dddddddd | R0 → (disp + GBR) | 1 | — |
| MOV.W | R0,@(disp,GBR) | 11000001dddddddd | R0 → (disp × 2 + GBR) | 1 | — |
| MOV.L | R0,@(disp,GBR) | 11000010dddddddd | R0 → (disp × 4 + GBR) | 1 | — |
| MOV.B | @(disp,GBR),R0 | 11000100dddddddd | (disp + GBR) → Sign extension → R0 | 1 | — |
| MOV.W | @(disp,GBR),R0 | 11000101dddddddd | (disp × 2 + GBR) → Sign extension → R0 | 1 | — |
| MOV.L | @(disp,GBR),R0 | 11000110dddddddd | (disp × 4 + GBR) → R0 | 1 | — |
| MOVA | @(disp,PC),R0 | 11000111dddddddd | disp × 4 + PC → R0 | 1 | — |
| MOVT | Rn | 0000nnnn00101001 | T → Rn | 1 | — |
| SWAP.B | Rm,Rn | 0110nnnnmmmm1000 | Rm → Swap the bottom two bytes → Rn | 1 | — |
| SWAP.W | Rm,Rn | 0110nnnnmmmm1001 | Rm → Swap upper and lower words → Rn | 1 | — |
| XTRCT | Rm,Rn | 0010nnnnmmmm1101 | Rm: Middle 32 bits of Rn → Rn | 1 | — |

**Table 2.21 Arithmetic Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| ADD | Rm,Rn | 0011nnnnmmmm1100 | Rn + Rm → Rn | 1 | — |
| ADD | #imm,Rn | 0111nnnniiiiiiii | Rn + imm → Rn | 1 | — |
| ADDC | Rm,Rn | 0011nnnnmmmm1110 | Rn + Rm + T → Rn, Carry → T | 1 | Carry |
| ADDV | Rm,Rn | 0011nnnnmmmm1111 | Rn + Rm → Rn, Overflow → T | 1 | Overflow |
| CMP/EQ | #imm,R0 | 10001000iiiiiiii | If R0 = imm, 1 → T | 1 | Comparison result |
| CMP/EQ | Rm,Rn | 0011nnnnmmmm0000 | If Rn = Rm, 1 → T | 1 | Comparison result |
| CMP/HS | Rm,Rn | 0011nnnnmmmm0010 | If Rn ≥ Rm with unsigned data, 1 → T | 1 | Comparison result |
| CMP/GE | Rm,Rn | 0011nnnnmmmm0011 | If Rn ≥ Rm with signed data, 1 → T | 1 | Comparison result |
| CMP/HI | Rm,Rn | 0011nnnnmmmm0110 | If Rn > Rm with unsigned data, 1 → T | 1 | Comparison result |
| CMP/GT | Rm,Rn | 0011nnnnmmmm0111 | If Rn > Rm with signed data, 1 → T | 1 | Comparison result |
| CMP/PL | Rn | 0100nnnn00010101 | If Rn > 0, 1 → T | 1 | Comparison result |
| CMP/PZ | Rn | 0100nnnn00010001 | If Rn ≥ 0, 1 → T | 1 | Comparison result |
| CMP/STR | Rm,Rn | 0010nnnnmmmm1100 | If Rn and Rm contain an identical byte, 1 → T | 1 | Comparison result |
| DIV1 | Rm,Rn | 0011nnnnmmmm0100 | Single-step division (Rn/Rm) | 1 | Calculation result |
| DIV0S | Rm,Rn | 0010nnnnmmmm0111 | MSB of Rn → Q, MSB of Rm → M, M ^ Q → T | 1 | Calculation result |
| DIV0U | | 0000000000011001 | 0 → M/Q/T | 1 | 0 |

**Table 2.21 Arithmetic Instructions (cont)**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| DMULS.L | Rm,Rn | 0011nnnnmmmm1101 | Signed operation of Rn × Rm → MACH, MACL 32 × 32 → 64 bits | 2 to 4* | — |
| DMULU.L | Rm,Rn | 0011nnnnmmmm0101 | Unsigned operation of Rn × Rm → MACH, MACL 32 × 32 → 64 bits | 2 to 4* | — |
| DT | Rn | 0100nnnn00010000 | Rn – 1 → Rn, when Rn is 0, 1 → T<br>When Rn is nonzero, 0 → T | 1 | Comparison result |
| EXTS.B | Rm,Rn | 0110nnnnmmmm1110 | A byte in Rm is sign-extended → Rn | 1 | — |
| EXTS.W | Rm,Rn | 0110nnnnmmmm1111 | A word in Rm is sign-extended → Rn | 1 | — |
| EXTU.B | Rm,Rn | 0110nnnnmmmm1100 | A byte in Rm is zero-extended → Rn | 1 | — |
| EXTU.W | Rm,Rn | 0110nnnnmmmm1101 | A word in Rm is zero-extended → Rn | 1 | — |
| MAC.L | @Rm+,@Rn+ | 0000nnnnmmmm1111 | Signed operation of (Rn) × (Rm) + MAC → MAC 32 × 32 + 64 → 64 bits | 3/(2 to 4)* | — |
| MAC.W | @Rm+,@Rn+ | 0100nnnnmmmm1111 | Signed operation of (Rn) × (Rm) + MAC → MAC 16 × 16 + 64 → 64 bits | 3/(2)* | — |
| MUL.L | Rm,Rn | 0000nnnnmmmm0111 | Rn × Rm → MACL, 32 × 32 → 32 bits | 2 to 4* | — |
| MULS.W | Rm,Rn | 0010nnnnmmmm1111 | Signed operation of Rn × Rm → MAC 16 × 16 → 32 bits | 1 to 3* | — |
| MULU.W | Rm,Rn | 0010nnnnmmmm1110 | Unsigned operation of Rn × Rm → MAC 16 × 16 → 32 bits | 1 to 3* | — |
| NEG | Rm,Rn | 0110nnnnmmmm1011 | 0–Rm → Rn | 1 | — |
| NEGC | Rm,Rn | 0110nnnnmmmm1010 | 0–Rm–T → Rn, Borrow → T | 1 | Borrow |

**Table 2.21 Arithmetic Instructions (cont)**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| SUB | Rm,Rn | 0011nnnnmmmm1000 | Rn–Rm → Rn | 1 | — |
| SUBC | Rm,Rn | 0011nnnnmmmm1010 | Rn–Rm–T → Rn, Borrow → T | 1 | Borrow |
| SUBV | Rm,Rn | 0011nnnnmmmm1011 | Rn–Rm → Rn, Underflow → T | 1 | Underflow |

Note: The normal number of execution cycles. The number in parentheses is the number of execution cycles in the case of contention with preceding or following instructions.

**Table 2.22 Logic Operation Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| AND | Rm,Rn | 0010nnnnmmmm1001 | Rn & Rm → Rn | 1 | — |
| AND | #imm,R0 | 11001001iiiiiiii | R0 & imm → R0 | 1 | — |
| AND.B | #imm,@(R0,GBR) | 11001101iiiiiiii | (R0 + GBR) & imm → (R0 + GBR) | 3 | — |
| NOT | Rm,Rn | 0110nnnnmmmm0111 | ~Rm → Rn | 1 | — |
| OR | Rm,Rn | 0010nnnnmmmm1011 | Rn \| Rm → Rn | 1 | — |
| OR | #imm,R0 | 11001011iiiiiiii | R0 \| imm → R0 | 1 | — |
| OR.B | #imm,@(R0,GBR) | 11001111iiiiiiii | (R0 + GBR) \| imm → (R0 + GBR) | 3 | — |
| TAS.B | @Rn | 0100nnnn00011011 | If (Rn) is 0, 1 → T, 1 → MSB of (Rn) | 4 | Test result |
| TST | Rm,Rn | 0010nnnnmmmm1000 | Rn & Rm, if the result is 0, 1 → T | 1 | Test result |
| TST | #imm,R0 | 11001000iiiiiiii | R0 & imm, if the result is 0, 1 → T | 1 | Test result |
| TST.B | #imm,@(R0,GBR) | 11001100iiiiiiii | (R0 + GBR) & imm, if the result is 0, 1 → T | 3 | Test result |
| XOR | Rm,Rn | 0010nnnnmmmm1010 | Rn ^ Rm → Rn | 1 | — |
| XOR | #imm,R0 | 11001010iiiiiiii | R0 ^ imm → R0 | 1 | — |
| XOR.B | #imm,@(R0,GBR) | 11001110iiiiiiii | (R0 + GBR) ^ imm → (R0 + GBR) | 3 | — |

**Table 2.23 Shift Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| ROTL | Rn | 0100nnnn00000100 | T ← Rn ← MSB | 1 | MSB |
| ROTR | Rn | 0100nnnn00000101 | LSB → Rn → T | 1 | LSB |
| ROTCL | Rn | 0100nnnn00100100 | T ← Rn ← T | 1 | MSB |
| ROTCR | Rn | 0100nnnn00100101 | T → Rn → T | 1 | LSB |
| SHAL | Rn | 0100nnnn00100000 | T ← Rn ← 0 | 1 | MSB |
| SHAR | Rn | 0100nnnn00100001 | MSB → Rn → T | 1 | LSB |
| SHLL | Rn | 0100nnnn00000000 | T ← Rn ← 0 | 1 | MSB |
| SHLR | Rn | 0100nnnn00000001 | 0 → Rn → T | 1 | LSB |
| SHLL2 | Rn | 0100nnnn00001000 | Rn<<2 → Rn | 1 | — |
| SHLR2 | Rn | 0100nnnn00001001 | Rn>>2 → Rn | 1 | — |
| SHLL8 | Rn | 0100nnnn00011000 | Rn<<8 → Rn | 1 | — |
| SHLR8 | Rn | 0100nnnn00011001 | Rn>>8 → Rn | 1 | — |
| SHLL16 | Rn | 0100nnnn00101000 | Rn<<16 → Rn | 1 | — |
| SHLR16 | Rn | 0100nnnn00101001 | Rn>>16 → Rn | 1 | — |

**Table 2.24 Branch Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| BF | label | 10001011dddddddd | If T = 0, disp × 2 + PC → PC, if T = 1, nop | 3/1* | — |
| BF/S | label | 10001111dddddddd | Delayed branch, if T = 0, disp × 2 + PC → PC, if T = 1, nop | 2/1* | — |
| BT | label | 10001001dddddddd | If T = 1, disp × 2 + PC → PC, if T = 0, nop | 3/1* | — |
| BT/S | label | 10001101dddddddd | Delayed branch, if T = 1, disp × 2 + PC → PC, if T = 0, nop | 2/1* | — |
| BRA | label | 1010dddddddddddd | Delayed branch, disp × 2 + PC → PC | 2 | — |
| BRAF | Rm | 0000mmmm00100011 | Delayed branch, Rm + PC → PC | 2 | — |
| BSR | label | 1011dddddddddddd | Delayed branch, PC → PR, disp × 2 + PC → PC | 2 | — |
| BSRF | Rm | 0000mmmm00000011 | Delayed branch, PC → PR, Rm + PC → PC | 2 | — |
| JMP | @Rm | 0100mmmm00101011 | Delayed branch, Rm → PC | 2 | — |
| JSR | @Rm | 0100mmmm00001011 | Delayed branch, PC → PR, Rm → PC | 2 | — |
| RTS | | 0000000000001011 | Delayed branch, PR → PC | 2 | — |

Note: * One state when it does not branch.

**Table 2.25 System Control Instructions**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| CLRMAC | | 0000000000101000 | 0 → MACH, MACL | 1 | — |
| CLRT | | 0000000000001000 | 0 → T | 1 | 0 |
| LDC | Rm,SR | 0100mmmm00001110 | Rm → SR | 1 | LSB |
| LDC | Rm,GBR | 0100mmmm00011110 | Rm → GBR | 1 | — |
| LDC | Rm,VBR | 0100mmmm00101110 | Rm → VBR | 1 | — |
| LDC | Rm,MOD | 0100mmmm01011110 | Rm → MOD | 1 | — |
| LDC | Rm,RE | 0100mmmm01111110 | Rm → RE | 1 | — |
| LDC | Rm,RS | 0100mmmm01101110 | Rm → RS | 1 | — |
| LDC.L | @Rm+,SR | 0100mmmm00000111 | (Rm) → SR, Rm + 4 → Rm | 3 | LSB |
| LDC.L | @Rm+,GBR | 0100mmmm00010111 | (Rm) → GBR, Rm + 4 → Rm | 3 | — |
| LDC.L | @Rm+,VBR | 0100mmmm00100111 | (Rm) → VBR, Rm + 4 → Rm | 3 | — |
| LDC.L | @Rm+,MOD | 0100mmmm01010111 | (Rm) → MOD, Rm + 4 → Rm | 3 | — |
| LDC.L | @Rm+,RE | 0100mmmm01110111 | (Rm) → RE, Rm + 4 → Rm | 3 | — |
| LDC.L | @Rm+,RS | 0100mmmm01100111 | (Rm) → RS, Rm + 4 → Rm | 3 | — |
| LDRE | @(disp,PC) | 10001110dddddddd | disp × 2 + PC → RE | 1 | — |
| LDRS | @(disp,PC) | 10001100dddddddd | disp × 2 + PC → RS | 1 | — |
| LDS | Rm,MACH | 0100mmmm00001010 | Rm → MACH | 1 | — |
| LDS | Rm,MACL | 0100mmmm00011010 | Rm → MACL | 1 | — |
| LDS | Rm,PR | 0100mmmm00101010 | Rm → PR | 1 | — |
| LDS | Rm,DSR | 0100mmmm01101010 | Rm → DSR | 1 | — |
| LDS | Rm,A0 | 0100mmmm01111010 | Rm → A0 | 1 | — |
| LDS | Rm,X0 | 0100mmmm10001010 | Rm → X0 | 1 | — |
| LDS | Rm,X1 | 0100mmmm10011010 | Rm → X1 | 1 | — |
| LDS | Rm,Y0 | 0100mmmm10101010 | Rm → Y0 | 1 | — |
| LDS | Rm,Y1 | 0100mmmm10111010 | Rm → Y1 | 1 | — |
| LDS.L | @Rm+,MACH | 0100mmmm00000110 | (Rm) → MACH,<br>Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,MACL | 0100mmmm00010110 | (Rm) → MACL,<br>Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,PR | 0100mmmm00100110 | (Rm) → PR, Rm + 4 → Rm | 1 | — |

**Table 2.25 System Control Instructions (cont)**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| LDS.L | @Rm+,DSR | 0100mmmm01100110 | (Rm) → DSR, Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,A0 | 0100mmmm01110110 | (Rm) → A0, Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,X0 | 0100mmmm10000110 | (Rm) → X0, Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,X1 | 0100mmmm10010110 | (Rm) → X1, Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,Y0 | 0100mmmm10100110 | (Rm) → Y0, Rm + 4 → Rm | 1 | — |
| LDS.L | @Rm+,Y1 | 0100mmmm10110110 | (Rm) → Y1, Rm + 4 → Rm | 1 | — |
| NOP | | 0000000000001001 | No operation | 1 | — |
| RTE | | 0000000000101011 | Delayed branch,<br>stack area → PC/SR | 4 | LSB |
| SETRC | Rm | 0100mmmm00010100 | RE–RS operation result<br>(repeat status) → RF1, RF0<br>Rm[11:0] → RC (SR[27:16]) | 1 | — |
| SETRC | #imm | 10000010iiiiiiii | RE–RS operation result<br>(repeat status) → RF1, RF0<br>imm → RC (SR[23:16]), zeros<br>→ SR[27:24] | 1 | 1 |
| SETT | | 0000000000011000 | 1 → T | 1 | 1 |
| SLEEP | | 0000000000011011 | Sleep | 3* | — |
| STC | SR,Rn | 0000nnnn00000010 | SR → Rn | 1 | — |
| STC | GBR,Rn | 0000nnnn00010010 | GBR → Rn | 1 | — |
| STC | VBR,Rn | 0000nnnn00100010 | VBR → Rn | 1 | — |
| STC | MOD,Rn | 0000nnnn01010010 | MOD → Rn | 1 | — |
| STC | RE,Rn | 0000nnnn01110010 | RE → Rn | 1 | — |
| STC | RS,Rn | 0000nnnn01100010 | RS → Rn | 1 | — |
| STC.L | SR,@–Rn | 0100nnnn00000011 | Rn–4 → Rn, SR → (Rn) | 2 | — |
| STC.L | GBR,@–Rn | 0100nnnn00010011 | Rn–4 → Rn, GBR → (Rn) | 2 | — |
| STC.L | VBR,@–Rn | 0100nnnn00100011 | Rn–4 → Rn, VBR → (Rn) | 2 | — |
| STC.L | MOD,@–Rn | 0100nnnn01010011 | Rn–4 → Rn, MOD → (Rn) | 2 | — |
| STC.L | RE,@–Rn | 0100nnnn01110011 | Rn–4 → Rn, RE → (Rn) | 2 | — |
| STC.L | RS,@–Rn | 0100nnnn01100011 | Rn–4 → Rn, RS → (Rn) | 2 | — |

**Table 2.25 System Control Instructions (cont)**

| Instruction | | Instruction Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| STS | MACH,Rn | 0000nnnn00001010 | MACH → Rn | 1 | — |
| STS | MACL,Rn | 0000nnnn00011010 | MACL → Rn | 1 | — |
| STS | PR,Rn | 0000nnnn00101010 | PR → Rn | 1 | — |
| STS | DSR,Rn | 0000nnnn01101010 | DSR → Rn | 1 | — |
| STS | A0,Rn | 0000nnnn01111010 | A0 → Rn | 1 | — |
| STS | X0,Rn | 0000nnnn10001010 | X0 → Rn | 1 | — |
| STS | X1,Rn | 0000nnnn10011010 | X1 → Rn | 1 | — |
| STS | Y0,Rn | 0000nnnn10101010 | Y0 → Rn | 1 | — |
| STS | Y1,Rn | 0000nnnn10111010 | Y1 → Rn | 1 | — |
| STS.L | MACH,@–Rn | 0100nnnn00000010 | Rn–4 → Rn, MACH → (Rn) | 1 | — |
| STS.L | MACL,@–Rn | 0100nnnn00010010 | Rn–4 → Rn, MACL → (Rn) | 1 | — |
| STS.L | PR,@–Rn | 0100nnnn00100010 | Rn–4 → Rn, PR → (Rn) | 1 | — |
| STS.L | DSR,@–Rn | 0100nnnn01100010 | Rn–4 → Rn, DSR → (Rn) | 1 | — |
| STS.L | A0,@–Rn | 0100nnnn01110010 | Rn–4 → Rn, A0 → (Rn) | 1 | — |
| STS.L | X0,@–Rn | 0100nnnn10000010 | Rn–4 → Rn, X0 → (Rn) | 1 | — |
| STS.L | X1,@–Rn | 0100nnnn10010010 | Rn–4 → Rn, X1 → (Rn) | 1 | — |
| STS.L | Y0,@–Rn | 0100nnnn10100010 | Rn–4 → Rn, Y0 → (Rn) | 1 | — |
| STS.L | Y1,@–Rn | 0100nnnn10110010 | Rn–4 → Rn, Y1 → (Rn) | 1 | — |
| TRAPA | #imm | 11000011iiiiiiii | PC/SR → stack area, (imm × 4 + VBR) → PC | 8 | — |

Note: * The number of execution cycles before the chip enters sleep mode.

**Precautions Concerning the Number of Instruction Execution Cycles:** The execution cycles listed in the tables are minimum values. In practice, the number of execution cycles increases under such conditions as 1) when the instruction fetch is in contention with a data access, 2) when the destination register of a load instruction (memory → register) is the same as the register used by the next instruction, 3) when the branch destination address of a branch instruction is a 4n + 2 address.

**CPU Instructions That Support DSP Functions:** A number of system control instructions have been added to the CPU core instructions to support DSP functions. The RS, RE and MOD registers have been added to support repeat control and modulo addressing, and the repeat counter (RC) has been added to the status register (SR). The LDC and STC instructions have been added in order to access the aforementioned. The LDS and STS instructions have been added in order to access the DSP registers DSR, A0, X0, X1, Y0 and Y1.

The SETRC instruction has been added to set the repeat counter (RC, bits 27 to 16) and repeat flags (RF1, RF0, bits 3 and 2) of the SR register. When the SETRC instruction operand is immediate, the 8-bit immediate data is stored in bits 23 to 16 of the SR register and bits 27 to 24 are cleared to 0. When the operand is a register, bits 11 to 0 (12 bits) of the register are stored in bits 27 to 16 of the SR register. Additionally, the status of 1 instruction repeat (00), 2 instruction repeat (01), 3 instruction repeat (11) or 4 instruction or greater repeat (10) is set from the RS and RE set values.

In addition to the LDC instruction, the LDRS and LDRE instructions have been added for establishing the repeat start and repeat end addresses in the RS and RE registers.

The added instructions are listed in table 2.26.

**Table 2.26 Added CPU Instructions**

| Instruction | | Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| LDC | Rm,MOD | 0100mmmm01011110 | Rm→MOD | 1 | — |
| LDC | Rm,RE | 0100mmmm01111110 | Rm→RE | 1 | — |
| LDC | Rm,RS | 0100mmmm01101110 | Rm→RS | 1 | — |
| LDC.L | @Rm+,MOD | 0100mmmm01010111 | (Rm)→MOD,Rm+4→Rm | 3 | — |
| LDC.L | @Rm+,RE | 0100mmmm01110111 | (Rm)→RE,Rm+4→Rm | 3 | — |
| LDC.L | @Rm+,RS | 0100mmmm01100111 | (Rm)→RS,Rm+4→Rm | 3 | — |
| STC | MOD,Rn | 0000nnnn01010010 | MOD→Rn | 1 | — |
| STC | RE,Rn | 0000nnnn01110010 | RE→Rn | 1 | — |
| STC | RS,Rn | 0000nnnn01100010 | RS→Rn | 1 | — |
| STC.L | MOD,@-Rn | 0100nnnn01010011 | Rn–4→Rn,MOD→(Rn) | 2 | — |
| STC.L | RE,@-Rn | 0100nnnn01110011 | Rn–4→Rn,RE→(Rn) | 2 | — |
| STC.L | RS,@-Rn | 0100nnnn01100011 | Rn–4→Rn,RS→(Rn) | 2 | — |
| LDS | Rm,DSR | 0100mmmm01101010 | Rm→DSR | 1 | — |
| LDS.L | @Rm+,DSR | 0100mmmm01100110 | (Rm)→DSR,Rm+4→Rm | 1 | — |
| LDS | Rm,A0 | 0100mmmm01111010 | (Rm)→A0 | 1 | — |
| LDS.L | @Rm+,A0 | 0100mmmm01110110 | (Rm)→A0,Rm+4→Rm | 1 | — |
| LDS | Rm,X0 | 0100mmmm10001010 | (Rm)→X0 | 1 | — |
| LDS.L | @Rm+,X0 | 0100mmmm10000110 | (Rm)→X0,Rm+4→Rm | 1 | — |
| LDS | Rm,X1 | 0100mmmm10011010 | (Rm)→X1 | 1 | — |
| LDS.L | @Rm+,X1 | 0100mmmm10010110 | (Rm)→X1,Rm+4→Rm | 1 | — |
| LDS | Rm,Y0 | 0100mmmm10101010 | (Rm)→Y0 | 1 | — |
| LDS.L | @Rm+,Y0 | 0100mmmm10100110 | (Rm)→Y0,Rm+4→Rm | 1 | — |
| LDS | Rm,Y1 | 0100mmmm10111010 | (Rm)→Y1 | 1 | — |
| LDS.L | @Rm+,Y1 | 0100mmmm10110110 | (Rm)→Y1,Rm+4→Rm | 1 | — |
| STS | DSR,Rn | 0000nnnn01101010 | DSR→Rn | 1 | — |
| STS.L | DSR,@-Rn | 0100nnnn01100010 | Rn–4→Rn,DSR→(Rn) | 1 | — |
| STS | A0,Rn | 0000nnnn01111010 | A0→Rn | 1 | — |
| STS.L | A0,@-Rn | 0100nnnn01110010 | Rn–4→Rn,A0→(Rn) | 1 | — |
| STS | X0,Rn | 0000nnnn10001010 | X0→Rn | 1 | — |
| STS.L | X0,@-Rn | 0100nnnn10000010 | Rn–4→Rn,X0→(Rn) | 1 | — |
| STS | X1,Rn | 0000nnnn10011010 | X1→Rn | 1 | — |

**Table 2.26 Added CPU Instructions (cont)**

| Instruction | | Code | Operation | Cycles | T Bit |
|---|---|---|---|---|---|
| STS.L | X1,@-Rn | 0100nnnn10010010 | Rn–4→Rn,X1→(Rn) | 1 | — |
| STS | Y0,Rn | 0000nnnn10101010 | Y0→Rn | 1 | — |
| STS.L | Y0,@-Rn | 0100nnnn10100010 | Rn–4→Rn,Y0→(Rn) | 1 | — |
| STS | Y1,Rn | 0000nnnn10111010 | Y1→Rn | 1 | — |
| STS.L | Y1,@-Rn | 0100nnnn10110010 | Rn–4→Rn,Y1→(Rn) | 1 | — |
| SETRC | Rm | 0100mmmm00010100 | Rm[11:0]→RC (SR[27:16]) | 1 | — |
| SETRC | #imm | 10000010iiiiiiii | imm→RC(SR[23:16]), 0→SR[27:24] | 1 | — |
| LDRS | @(disp,PC) | 10001100dddddddd | disp × 2+PC→RS | 1 | — |
| LDRE | @(disp,PC) | 10001110dddddddd | disp × 2+PC→RE | 1 | — |

### 2.5.2 DSP Data Transfer Instruction Set

Table 2.27 lists the DSP data transfer instructions by classification.

**Table 2.27 Classification of DSP Data Transfer Instructions**

| Classification | Types | Operation Code | Function | No. of Instructions |
|---|---|---|---|---|
| Double data transfer instructions | 4 | NOPX | X memory no operation | 14 |
| | | MOVX | X memory data transfer | |
| | | NOPY | Y memory no operation | |
| | | MOVY | Y memory data transfer | |
| Single data transfer instructions | 1 | MOVS | Single data transfer | 16 |
| | Total: 5 | | | Total: 30 |

The data transfer instructions are divided into two groups, double data transfers and single data transfers. Double data transfers can be combined with DSP operation instructions to perform DSP parallel processing. The parallel processing instructions are 32 bit length, and the double data transfer instructions are incorporated into their A fields. Double data transfers that are not parallel processing instructions are 16 bit length, as are the single data transfer instructions.

The X memory and Y memory can be accessed simultaneously in parallel in double data transfers. One instruction each is designated from among the X and Y memory data accesses. The Ax

pointer is used to access X memory; the Ay pointer is used to access Y memory. Double data transfers can only access X, Y memory.

Single data transfers can be accessed from any area. Single data transfers use the Ax pointer and two other pointers as an As pointer.

**Table 2.28 Double Data Transfer Instructions (X Memory Data)**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| NOPX | No Operation | 1111000*0*0*00** | 1 | — |
| MOVX.W @Ax,Dx | (Ax)→MSW of Dx,0→LSW of Dx | 111100A*D*0*01** | 1 | — |
| MOVX.W @Ax+,Dx | (Ax)→MSW of Dx,0→LSW of Dx,Ax+2→Ax | 111100A*D*0*10** | 1 | — |
| MOVX.W @Ax+Ix,Dx | (Ax)→MSW of Dx,0→LSW of Dx,Ax+Ix→Ax | 111100A*D*0*11** | 1 | — |
| MOVX.W Da,@Ax | MSW of Da→(Ax) | 111100A*D*1*01** | 1 | — |
| MOVX.W Da,@Ax+ | MSW of Da→(Ax),Ax+2→Ax | 111100A*D*1*10** | 1 | — |
| MOVX.W Da,@Ax+Ix | MSW of Da→(Ax),Ax+Ix→Ax | 111100A*D*1*11** | 1 | — |

**Table 2.29 Double Data Transfer Instructions (Y Memory Data)**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| NOPY | No Operation | 111100*0*0*0**00 | 1 | — |
| MOVY.W @Ay,Dy | (Ay)→MSW of Dy,0→LSW of Dy | 111100*A*D*0**01 | 1 | — |
| MOVY.W @Ay+,Dy | (Ay)→MSW of Dy,0→LSW of Dy, Ay+2→Ay | 111100*A*D*0**10 | 1 | — |
| MOVY.W @Ay+Iy,Dy | (Ay)→MSW of Dy,0→LSW of Dy, Ay+Iy→Ay | 111100*A*D*0**11 | 1 | — |
| MOVY.W Da,@Ay | MSW of Da→(Ay) | 111100*A*D*1**01 | 1 | — |
| MOVY.W Da,@Ay+ | MSW of Da→(Ay),Ay+2→Ay | 111100*A*D*1**10 | 1 | — |
| MOVY.W Da,@Ay+Iy | MSW of Da→(Ay),Ay+Iy→Ay | 111100*A*D*1**11 | 1 | — |

**Table 2.30 Single Data Transfer Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| MOVS.W @-As,Ds | As–2→As,(As)→MSW of Ds,0→LSW of Ds | 111101AADDDD0000 | 1 | — |
| MOVS.W @As,Ds | (As)→MSW of Ds,0→LSW of Ds | 111101AADDDD0100 | 1 | — |
| MOVS.W @As+,Ds | (As)→MSW of Ds,0→LSW of Ds, As+2→As | 111101AADDDD1000 | 1 | — |
| MOVS.W @As+Ix,Ds | (As)→MSW of Ds,0→LSW of Ds, As+Ix→As | 111101AADDDD1100 | 1 | — |
| MOVS.W Ds,@-As | As–2→As,MSW of Ds→(As)* | 111101AADDDD0001 | 1 | — |
| MOVS.W Ds,@As | MSW of Ds→(As)* | 111101AADDDD0101 | 1 | — |
| MOVS.W Ds,@As+ | MSW of Ds→(As)*,As+2→As | 111101AADDDD1001 | 1 | — |
| MOVS.W Ds,@As+Is | MSW of Ds→(As)*,As+Is→As | 111101AADDDD1101 | 1 | — |
| MOVS.L @-As,Ds | As–4→As,(As)→Ds | 111101AADDDD0010 | 1 | — |
| MOVS.L @As,Ds | (As)→Ds | 111101AADDDD0110 | 1 | — |
| MOVS.L @As+,Ds | (As)→Ds,As+4→As | 111101AADDDD1010 | 1 | — |
| MOVS.L @As+Is,Ds | (As)→Ds,As+Is→As | 111101AADDDD1110 | 1 | — |
| MOVS.L Ds, @-As | As–4→As,Ds→(As)* | 111101AADDDD0011 | 1 | — |
| MOVS.L Ds,@As | Ds→(As)* | 111101AADDDD0111 | 1 | — |
| MOVS.L Ds,@As+ | Ds→(As)*,As+4→As | 111101AADDDD1011 | 1 | — |
| MOVS.L Ds,@As+Is | Ds→(As)*,As+Is→As | 111101AADDDD1111 | 1 | — |

Note: * When guard bit registers A0G and A1G are specified for the source operand Ds, data is sign-extended before being transferred.

Table 2.31 shows the correspondence between the DSP data transfer operands and registers. CPU core registers are used as pointer addresses indicating memory addresses.

**Table 2.31 Correspondence between DSP Data Transfer Operands and Registers**

| | SH (CPU Core) Registers | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Oper-and** | **R0** | **R1** | **R2 (As2)** | **R3 (As3)** | **R4 (Ax0) (As0)** | **R5 (Ax1) (As0)** | **R6 (Ay0)** | **R7 (Ay1)** | **R8 (Ix) (Is)** | **R9 (Iy)** |
| Ax | — | — | — | — | Yes | Yes | — | — | — | — |
| Ix (Is) | — | — | — | — | — | — | — | — | Yes | — |
| Dx | — | — | — | — | — | — | — | — | — | — |
| Ay | — | — | — | — | — | — | Yes | Yes | — | — |
| Iy | — | — | — | — | — | — | — | — | — | Yes |
| Dy | — | — | — | — | — | — | — | — | — | — |
| Da | — | — | — | — | — | — | — | — | — | — |
| As | — | — | Yes | Yes | Yes | Yes | — | — | — | — |
| Ds | — | — | — | — | — | — | — | — | — | — |

| | DSP Registers | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Oper-and** | **X0** | **X1** | **Y0** | **Y1** | **M0** | **M1** | **A0** | **A1** | **A0G** | **A1G** |
| Ax | — | — | — | — | — | — | — | — | — | — |
| Ix (Is) | — | — | — | — | — | — | — | — | — | — |
| Dx | Yes | Yes | — | — | — | — | — | — | — | — |
| Ay | — | — | — | — | — | — | — | — | — | — |
| Iy | — | — | — | — | — | — | — | — | — | — |
| Dy | — | — | Yes | Yes | — | — | — | — | — | — |
| Da | — | — | — | — | — | — | Yes | Yes | — | — |
| As | — | — | — | — | — | — | — | — | — | — |
| Ds | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

Note: Yes indicates that the register can be set.

### 2.5.3 DSP Operation Instruction Set

DSP operation instructions are digital signal processing instructions processed by the DSP unit. These instructions use 32-bit instruction codes, and multiple instructions are executed in parallel. The instruction codes are divided into an A field and a B field; parallel data transfer instructions are designated in the A field, and single or double data operation instructions are designated in the B field. Instructions can be independently designated and execution can also be carried out independently. A parallel data transfer instruction designated in the A field is exactly the same as a double data transfer instruction.

The B field data operation instructions are divided into three groups: double data operation instructions, conditional single data operation instructions, and unconditional single data operation instructions. Table 2.32 lists the instruction formats of the DSP operation instructions. Each of the operands can be independently selected from the DSP registers. Table 2.33 shows the correspondence between the DSP operation instruction operands and registers.

**Table 2.32 DSP Operation Instruction Formats**

| Classification | | Instruction Forms | Instruction |
|---|---|---|---|
| Double data operation instructions (6 operands) | | ALUop. Sx, Sy, Du<br>MLTop. Se, Sf, Dg | PADD PMULS,<br>PSUB PMULS |
| Conditional single data operation instructions | 3 operands | ALUop. Sx, Sy, Dz<br>DCT ALUop. Sx, Sy, Dz<br>DCF ALUop. Sx, Sy, Dz | PADD, PAND, POR, PSHA, PSHL, PSUB, PXOR |
| | 2 operands | ALUop. Sx, Dz<br>DCT ALUop. Sx, Dz<br>DCF ALUop. Sx, Dz<br>ALUop. Sy, Dz<br>DCT ALUop. Sy, Dz<br>DCF ALUop. Sy, Dz | PCOPY, PDEC, PDMSB, PINC, PLDS, PSTS, PNEG |
| | 1 operand | ALUop. Dz<br>DCT ALUop. Dz<br>DCF ALUop. Dz | PCLR |
| Unconditional single data operation instructions | 3 operands | ALUop. Sx, Sy, Du<br>MLTop. Se, Sf, Dg | PADDC, PSUBC, PMULS |
| | 2 operands | ALUop. Sx, Dz<br>ALUop. Sy, Dz<br>ALUop. Sx, Sy | PCMP, PABS, PRND |
| | 1 operand | ALUop. Dz | PSHA #imm,<br>PSHL #imm |

**Table 2.33 Correspondence between DSP Instruction Operands and Registers**

| | ALU and BPU Instructions | | | | Multiplication Instructions | | |
|---|---|---|---|---|---|---|---|
| **Register** | **Sx** | **Sy** | **Dz** | **Du** | **Se** | **Sf** | **Dg** |
| A0 | Yes | — | Yes | Yes | — | — | Yes |
| A1 | Yes | — | Yes | Yes | Yes | Yes | Yes |
| M0 | — | Yes | Yes | — | — | — | Yes |
| M1 | — | Yes | Yes | — | — | — | Yes |
| X0 | Yes | — | Yes | Yes | Yes | Yes | — |
| X1 | Yes | — | Yes | — | Yes | — | — |
| Y0 | — | Yes | Yes | Yes | Yes | Yes | — |
| Y1 | — | Yes | Yes | — | — | Yes | — |

When writing parallel instructions, write the B field instructions first, then write the A field instructions:

```
     PADD A0,M0,A0 PMULS X0,Y0,M0   MOVX.W @R4+,X0       MOVY.W @R6+,Y0[;]
DCF  PINC X1,A1                     MOVX.W A0,@R5+R8    MOVY.W@R7+,Y0[;]
     PCMP X1,M0                     MOVX.W @R4          [NOPY][;]
```

Text in brackets ([]) can be omitted. The no operation instructions NOPX and NOPY can be omitted. Semicolons (;) are used to demarcate instruction lines, but can be omitted. If semicolons are used, the space after the semicolon can be used for comments.

The individual status codes (DC, N, Z, V, GT) of the DSR register are always updated by unconditional ALU operation instructions and shift operation instructions. Conditional instructions do not update the status codes, even if the conditions have been met. Multiplication instructions also do not update the status codes. DC bit definitions are determined by the specifications of the CS bits in the DSR register.

Table 2.34 lists the DSP operation instructions by classification.

**Table 2.34 Classification of DSP Instructions**

| Classification | | Instruction Types | Operation Code | Function | No. of Instructions |
|---|---|---|---|---|---|
| ALU arithmetic operation instructions | ALU fixed decimal point operation instructions | 11 | PABS | Absolute value operation | 28 |
| | | | PADD | Addition | |
| | | | PADD PMULS | Addition and signed multiplication | |
| | | | PADDC | Addition with carry | |
| | | | PCLR | Clear | |
| | | | PCMP | Compare | |
| | | | PCOPY | Copy | |
| | | | PNEG | Invert sign | |
| | | | PSUB | Subtraction | |
| | | | PSUB PMULS | Subtraction and signed multiplication | |
| | | | PSUBC | Subtraction with borrow | |
| | ALU integer operation instructions | 2 | PDEC | Decrement | 12 |
| | | | PINC | Increment | |
| | MSB detection instruction | 1 | PDMSB | MSB detection | 6 |
| | Rounding operation instruction | 1 | PRND | Rounding | 2 |
| ALU logical operation instructions | | 3 | PAND | Logical AND | 9 |
| | | | POR | Logical OR | |
| | | | PXOR | Logical exclusive OR | |
| Fixed decimal point multiplication instruction | | 1 | PMULS | Signed multiplication | 1 |
| Shift | Arithmetic shift operation instruction | 1 | PSHA | Arithmetic shift | 4 |
| | Logical shift operation instruction | 1 | PSHL | Logical shift | 4 |
| System control instructions | | 2 | PLDS | System register load | 12 |
| | | | PSTS | Store from system register | |
| | | Total 23 | | | Total 78 |

### 2.5.4 Various Operation Instructions

**ALU Arithmetic Operation Instructions:** Tables 2.35 to 2.44 list various operation instructions.

**Table 2.35 ALU Fixed Point Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PABS Sx,Dz | If Sx≥0,Sx→Dz | 111110********** | 1 | Update |
| | If Sx<0,0– Sx→Dz | 10001000xx00zzzz | | |
| PABS Sy,Dz | If Sy≥0,Sy→Dz | 111110********** | 1 | Update |
| | If Sy<0,0–Sy→Dz | 1010100000yyzzzz | | |
| PADD Sx,Sy,Dz | Sx+Sy→Dz | 111110********** | 1 | Update |
| | | 10110001xxyyzzzz | | |
| DCT PADD Sx,Sy,Dz | if DC=1,Sx+Sy→Dz if 0,nop | 111110********** | 1 | — |
| | | 10110010xxyyzzzz | | |
| DCF PADD Sx,Sy,Dz | if DC=0,Sx+Sy→Dz if 1,nop | 111110********** | 1 | — |
| | | 10110011xxyyzzzz | | |
| PADD Sx,Sy,Du | Sx+Sy→Du | 111110********** | 1 | Update |
| PMULS Se,Sf,Dg | MSW of Se × MSW of Sf→Dg | 0111eeffxxyygguu | | |
| PADDC Sx,Sy,Dz | Sx+Sy+DC→Dz | 111110********** | 1 | Update |
| | | 10110000xxyyzzzz | | |
| PCLR Dz | H'00000000→Dz | 111110********** | 1 | Update |
| | | 1000110100000zzzz | | |
| DCT PCLR Dz | if DC=1,H'00000000→Dz | 111110********** | 1 | — |
| | if 0,nop | 1000111000000zzzz | | |
| DCF PCLR Dz | if DC=0,H'00000000→Dz | 111110********** | 1 | — |
| | if 1,nop | 1000111100000zzzz | | |
| PCMP Sx,Sy | Sx–Sy | 111110********** | 1 | Update |
| | | 10000100xxyy0000 | | |
| PCOPY Sx,Dz | Sx→Dz | 111110********** | 1 | Update |
| | | 11011001xx00zzzz | | |
| PCOPY Sy,Dz | Sy→Dz | 111110********** | 1 | Update |
| | | 1111100100yyzzzz | | |
| DCT PCOPY Sx,Dz | if DC=1,Sx→Dz if 0,nop | 111110********** | 1 | — |
| | | 11011010xx00zzzz | | |

**Table 2.35 ALU Fixed Point Operation Instructions (cont)**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| DCT PCOPY Sy,Dz | if DC=1,Sy→Dz if 0,nop | 111110********** | 1 | — |
| | | 1111101000yyzzzz | | |
| DCF PCOPY Sx,Dz | if DC=0,Sx→Dz if 1,nop | 111110********** | 1 | — |
| | | 11011011xx00zzzz | | |
| DCF PCOPY Sy,Dz | if DC=0,Sy→Dz if 1,nop | 111110********** | 1 | — |
| | | 1111101100yyzzzz | | |
| PNEG Sx,Dz | 0–Sx→Dz | 111110********** | 1 | Update |
| | | 11001001xx00zzzz | | |
| PNEG Sy,Dz | 0–Sy→Dz | 111110********** | 1 | Update |
| | | 1110100100yyzzzz | | |
| DCT PNEG Sx,Dz | if DC=1,0–Sx→Dz | 111110********** | 1 | — |
| | if 0,nop | 11001010xx00zzzz | | |
| DCT PNEG Sy,Dz | if DC=1,0–Sy→Dz | 111110********** | 1 | — |
| | if 0,nop | 1110101000yyzzzz | | |
| DCF PNEG Sx,Dz | if DC=0,0–Sx→Dz | 111110********** | 1 | — |
| | if 1,nop | 11001011xx00zzzz | | |
| DCF PNEG Sy,Dz | if DC=0,0–Sy→Dz | 111110********** | 1 | — |
| | if 1,nop | 1110101100yyzzzz | | |
| PSUB Sx,Sy,Dz | Sx–Sy→Dz | 111110********** | 1 | Update |
| | | 10100001xxyyzzzz | | |
| DCT PSUB Sx,Sy,Dz | if DC=1,Sx–Sy→Dz if 0,nop | 111110********** | 1 | — |
| | | 10100010xxyyzzzz | | |
| DCF PSUB Sx,Sy,Dz | if DC=0,Sx–Sy→Dz if 1,nop | 111110********** | 1 | — |
| | | 10100011xxyyzzzz | | |
| PSUB Sx,Sy,Du | Sx–Sy→Du | 111110********** | 1 | Update |
| PMULS Se,Sf,Dg | MSW of Se × MSW of Sf→Dg | 0110eeffxxyygguu | | |
| PSUBC Sx,Sy,Dz | Sx–Sy–DC→Dz | 111110********** | 1 | Update |
| | | 10100000xxyyzzzz | | |

**Table 2.36 ALU Integer Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PDEC Sx,Dz | MSW of Sx – 1 → MSW of Dz, clear LSW of Dz | 111110********** 10001001xx00zzzz | 1 | Update |
| PDEC Sy,Dz | MSW of Sy – 1 → MSW of Dz, clear LSW of Dz | 111110********** 1010100100yyzzzz | 1 | Update |
| DCT PDEC Sx,Dz | If DC=1, MSW of Sx – 1 → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 10001010xx00zzzz | 1 | — |
| DCT PDEC Sy,Dz | If DC=1, MSW of Sy – 1 → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 1010101000yyzzzz | 1 | — |
| DCF PDEC Sx,Dz | If DC=0, MSW of Sx – 1 → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 10001011xx00zzzz | 1 | — |
| DCF PDEC Sy,Dz | If DC=0, MSW of Sy – 1 → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 1010101100yyzzzz | 1 | — |
| PINC Sx,Dz | MSW of Sx + 1 → MSW of Dz, clear LSW of Dz | 111110********** 10011001xx00zzzz | 1 | Update |
| PINC Sy,Dz | MSW of Sy + 1 → MSW of Dz, clear LSW of Dz | 111110********** 1011100100yyzzzz | 1 | Update |
| DCT PINC Sx,Dz | If DC=1, MSW of Sx + 1 → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 10011010xx00zzzz | 1 | — |
| DCT PINC Sy,Dz | If DC=1, MSW of Sy + 1 → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 1011101000yyzzzz | 1 | — |
| DCF PINC Sx,Dz | If DC=0, MSW of Sx + 1 → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 10011011xx00zzzz | 1 | — |
| DCF PINC Sy,Dz | If DC=0, MSW of Sy + 1 → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 1011101100yyzzzz | 1 | — |

**Table 2.37 MSB Detection Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PDMSB Sx,Dz | Sx data MSB position → MSW of Dz, clear LSW of Dz | 111110********** 10011101xx00zzzz | 1 | Update |
| PDMSB Sy,Dz | Sy data MSB position → MSW of Dz, clear LSW of Dz | 111110********** 1011110100yyzzzz | 1 | Update |
| DCT PDMSB Sx,Dz | If DC=1, Sx data MSB position → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 10011110xx00zzzz | 1 | — |
| DCT PDMSB Sy,Dz | If DC=1, Sy data MSB position → MSW of Dz, clear LSW of Dz; if 0, nop | 111110********** 1011111000yyzzzz | 1 | — |
| DCF PDMSB Sx,Dz | If DC=0, Sx data MSB position → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 10011111xx00zzzz | 1 | — |
| DCF PDMSB Sy,Dz | If DC=0, Sy data MSB position → MSW of Dz, clear LSW of Dz; if 1, nop | 111110********** 1011111100yyzzzz | 1 | — |

**Table 2.38 Rounding Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PRND Sx,Dz | Sx+H'00008000→Dz clear LSW of Dz | 111110********** 10011000xx00zzzz | 1 | Update |
| PRND Sy,Dz | Sy+H'00008000→Dz clear LSW of Dz | 111110********** 1011100000yyzzzz | 1 | Update |

**Table 2.39 ALU Logical Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PAND Sx,Sy,Dz | Sx & Sy → Dz, clear LSW of Dz | 111110********** 10010101xxyyzzzz | 1 | Update |
| DCT PAND Sx,Sy,Dz | If DC=1, Sx & Sy → Dz, clear LSW of Dz; if 0, nop | 111110********** 10010110xxyyzzzz | 1 | — |
| DCF PAND Sx,Sy,Dz | If DC=0, Sx & Sy → Dz, clear LSW of Dz; if 1, nop | 111110********** 10010111xxyyzzzz | 1 | — |
| POR Sx,Sy,Dz | Sx \| Sy → Dz, clear LSW of Dz | 111110********** 10110101xxyyzzzz | 1 | Update |
| DCT POR Sx,Sy,Dz | If DC=1, Sx \| Sy → Dz, clear LSW of Dz; if 0, nop | 111110********** 10110110xxyyzzzz | 1 | — |
| DCF POR Sx,Sy,Dz | If DC=0, Sx \| Sy → Dz, clear LSW of Dz; if 1, nop | 111110********** 10110111xxyyzzzz | 1 | — |
| PXOR Sx,Sy,Dz | Sx ^ Sy → Dz, clear LSW of Dz | 111110********** 10100101xxyyzzzz | 1 | Update |
| DCT PXOR Sx,Sy,Dz | If DC=1, Sx ^ Sy → Dz, clear LSW of Dz; if 0, nop | 111110********** 10100110xxyyzzzz | 1 | — |
| DCF PXOR Sx,Sy,Dz | If DC=0, Sx ^ Sy → Dz, clear LSW of Dz; if 1, nop | 111110********** 10100111xxyyzzzz | 1 | — |

**Table 2.40 Fixed Point Multiplication Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PMULS Se,Sf,Dg | MSW of Se × MSW of Sf→Dg | 111110********** 0100eeff0000gg00 | 1 | — |

**Table 2.41 Arithmetic Shift Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PSHA Sx,Sy,Dz | if Sy≥0,Sx<<Sy→Dz<br>if Sy<0,Sx>>Sy→Dz | 111110**********<br>10010001xxyyzzzz | 1 | Update |
| DCT PSHA Sx,Sy,Dz | if DC=1 & Sy≥0,Sx<<Sy→Dz<br>if DC=1 & Sy<0,Sx>>Sy→Dz<br>if DC=0,nop | 111110**********<br>10010010xxyyzzzz | 1 | — |
| DCF PSHA Sx,Sy,Dz | if DC=0 & Sy≥0,Sx<<Sy→Dz<br>if DC=0 & Sy<0,Sx>>Sy→Dz<br>if DC=1,nop | 111110**********<br>10010011xxyyzzzz | 1 | — |
| PSHA #imm,Dz | if imm≥0,Dz<<imm→Dz<br>if imm<0,Dz>>imm→Dz | 111110**********<br>00010iiiiiiizzzz | 1 | Update |

**Table 2.42 Logical Shift Operation Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PSHL Sx,Sy,Dz | if Sy≥0,Sx<<Sy→Dz, clear LSW of Dz<br>if Sy<0,Sx>>Sy→Dz, clear LSW of Dz | 111110**********<br>10000001xxyyzzzz | 1 | Update |
| DCT PSHL Sx,Sy,Dz | if DC=1 & Sy≥0,Sx<<Sy→Dz, clear LSW of Dz<br>if DC=1 & Sy<0,Sx>>Sy→Dz, clear LSW of Dz<br>if DC=0,nop | 111110**********<br>10000010xxyyzzzz | 1 | — |
| DCF PSHL Sx,Sy,Dz | if DC=0 & Sy≥0,Sx<<Sy→Dz, clear LSW of Dz<br>if DC=0 & Sy<0,Sx>>Sy→Dz, clear LSW of Dz<br>if DC=1,nop | 111110**********<br>10000011xxyyzzzz | 1 | — |
| PSHL #imm,Dz | if imm≥0,Dz<<imm→Dz, clear LSW of Dz<br>if imm<0,Dz>>imm→Dz, clear LSW of Dz | 111110**********<br>00000iiiiiiizzzz | 1 | Update |

**Table 2.43 System Control Instructions**

| Instruction | Operation | Code | Cycles | DC Bit |
|---|---|---|---|---|
| PLDS Dz,MACH | Dz→MACH | 111110********** | 1 | — |
| | | 1110110100000zzzz | | |
| PLDS Dz,MACL | Dz→MACL | 111110********** | 1 | — |
| | | 1111110100000zzzz | | |
| DCT PLDS Dz,MACH | if DC=1,Dz→MACH | 111110********** | 1 | — |
| | if 0,nop | 1110111000000zzzz | | |
| DCT PLDS Dz,MACL | if DC=1,Dz→MACL | 111110********** | 1 | — |
| | if 0,nop | 1111111000000zzzz | | |
| DCF PLDS Dz,MACH | if DC=0,Dz→MACH | 111110********** | 1 | — |
| | if 1,nop | 1110111100000zzzz | | |
| DCF PLDS Dz,MACL | if DC=0,Dz→MACL | 111110********** | 1 | — |
| | if 1,nop | 1111111100000zzzz | | |
| PSTS MACH,Dz | MACH→Dz | 111110********** | 1 | — |
| | | 1100110100000zzzz | | |
| PSTS MACL,Dz | MACL→Dz | 111110********** | 1 | — |
| | | 1101110100000zzzz | | |
| DCT PSTS MACH,Dz | if DC=1,MACH→Dz | 111110********** | 1 | — |
| | if 0,nop | 1100111000000zzzz | | |
| DCT PSTS MACL,Dz | if DC=1,MACL→Dz | 111110********** | 1 | — |
| | if 0,nop | 1101111000000zzzz | | |
| DCF PSTS MACH,Dz | if DC=0,MACH→Dz | 111110********** | 1 | — |
| | if 1,nop | 1100111100000zzzz | | |
| DCF PSTS MACL,Dz | if DC=0,MACL→Dz | 111110********** | 1 | — |
| | if 1,nop | 1101111100000zzzz | | |

When there are no data transfer instructions being processed simultaneously in parallel with DSP operation instructions, it is possible to either write NOPX, NOPY instructions or to omit the instructions. The instruction codes are the same regardless of whether the NOPX, NOPY instructions are written or omitted. Table 2.44 gives some examples of NOPX and NOPY instruction codes.

**Table 2.44 NOPX and NOPY Instruction Codes**

| Instruction | Code |
|---|---|
| PADD X0, Y0, A0 MOVX.W @R4+, X0 MOVY.W @R6+R9, Y0 | 1111100000001011<br>1011000100000111 |
| PADD X0, Y0, A0 NOPX MOVY.W @R6+R9, Y0 | 1111100000000011<br>1011000100000111 |
| PADD X0, Y0, A0 NOPX NOPY | 1111100000000000<br>1011000100000111 |
| PADD X0, Y0, A0 NOPX | 1111100000000000<br>1011000100000111 |
| PADD X0, Y0, A0 | 1111100000000000<br>1011000100000111 |
| MOVX.W @R4+, X0 MOVY.W @R6+R9, Y0 | 1111000000001011 |
| MOVX.W @R4+, X0 NOPY | 1111000000001000 |
| MOVS.W @R4+, X0 | 1111010010001000 |
| NOPX MOVY.W @R6+R9, Y0 | 1111000000000011 |
| MOVY.W @R6+R9, Y0 | 1111000000000011 |
| NOPX NOPY | 1111000000000000 |
| NOP | 0000000000001001 |

# Section 3 Oscillator Circuits and Operating Modes

## 3.1 Overview

Operation of the on-chip clock pulse generator, and CS0 area bus width specification, are controlled by the operating mode pins. A crystal resonator or external clock can be selected as the clock source.

## 3.2 On-Chip Clock Pulse Generator and Operating Modes

### 3.2.1 Clock Pulse Generator

A block diagram of the on-chip clock pulse generator circuit is shown in figure 3.1.

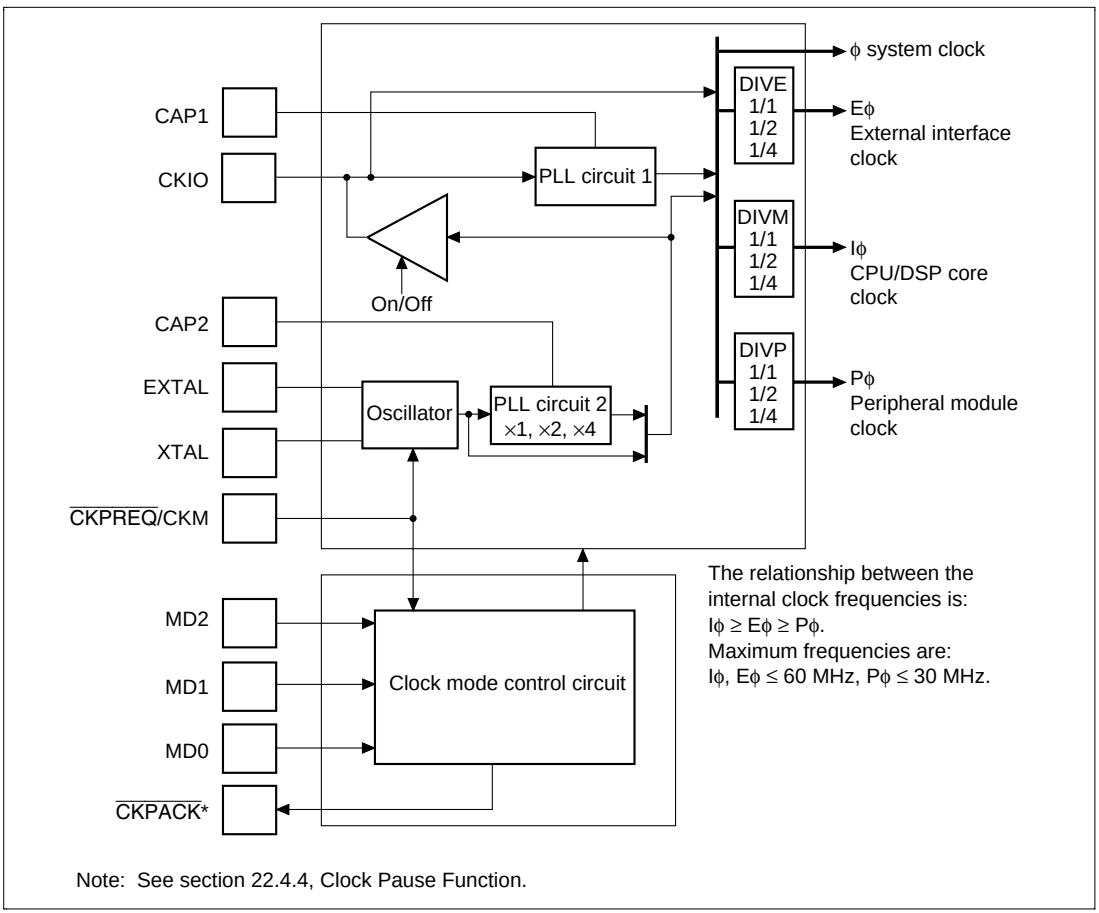



**Figure 3.1 Block Diagram of Clock Pulse Generator Circuit**

**Pin Configuration:** Table 3.1 lists the functions relating to the pins relating to the oscillator circuit.

**Table 3.1 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| CKIO | I/O | External clock input pin or internal clock output pin |
| XTAL | O | Connects to the crystal resonator |
| EXTAL | I | Connects to the crystal resonator or to the external clock input when using PLL circuit 2 |
| CAP1 | I | Connects to capacitance for operating PLL circuit 1 |
| CAP2 | I | Connects to capacitance for operating PLL circuit 2 |
| MD0 | I | The level applied to these pins specifies the clock mode |
| MD1 | I | |
| MD2 | I | |
| $\overline{\text{CKPREQ}}$/CKM | I | Used as the clock pause request pin, or specifies operation of the crystal resonator |

**PLL Circuit 1:** PLL circuit 1 eliminates phase differences between external clocks and clocks supplied internally within the chip. In high-speed operation, the phase difference between the reference clocks and operating clocks in the chip directly affects the interface margin with peripheral devices. On-chip PLL circuit 1 is provided to eliminate this effect.

**PLL Circuit 2:** PLL circuit 2 either leaves unchanged, doubles, or quadruples the frequency of clocks provided from the crystal resonator or the EXTAL pin external clock input for the chip operating frequency. The frequency modification register sets the clock frequency multiplication factor.

### 3.2.2 Clock Operating Mode Settings

Table 3.2 lists the functions and operation of clock modes 0 to 6.

**Table 3.2 Operating Modes**

| Clock Mode | Function/Operation | Clock Source |
|---|---|---|
| 0 | PLL circuits 1 and 2 operate. A clock is output with the same phase (with the same frequency as Eø) as the internal clocks (Iø, Eø, Pø) from the CKIO pin<br>PLL circuits 1 and 2 can be switched between the operating and halted states by means of control bits in the frequency modification register (FMR). The CKIO pin can also be placed in the high-impedance state<br>Normally, mode 0 should be used. | Crystal resonator/ external clock input |
| 1 | PLL circuits 1 and 2 operate. A clock (with the same frequency as Eø) 1/4 ø cycle in advance of the chip's internal system clock ø is output from the CKIO pin.<br>PLL circuits 1 and 2 can be switched between the operating and halted states by means of control bits in the frequency modification register (FMR). The CKIO pin can also be placed in the high-impedance state. However, clock phase shifting is not performed when PLL circuit 1 is halted.<br>Normally, mode 0 should be used. | |
| 2 | Only PLL circuit 2 operates. The clock from PLL circuit 2 is output from the CKIO pin (having the same frequency as the Eø). As PLL circuit 1 does not operate, phases are not matched in this mode<br>PLL circuit 2 can be switched between the operating and halted states by means of a control bit in the frequency modification register (FMR). The CKIO pin can also be placed in the high-impedance state | |
| 3 | Only PLL circuit 2 operates. The CKIO pin is high-impedance<br>PLL circuit 2 can be switched between the operating and halted states by means of a control bit in the frequency modification register (FMR) | |
| 4 | Only PLL circuit 1 operates. Operate PLL circuit 1 when operating with synchronization of the phases of the clock input from the CKIO pin and the internal clocks (Iø, Eø, Pø). PLL circuit 2 does not operate in this mode<br>PLL circuit 1 can be switched between the operating and halted states by means of a control bit in the frequency modification register (FMR) | External clock input |

**Table 3.2 Operating Modes (cont)**

| Clock Mode | Function/Operation | Clock Source |
|---|---|---|
| 5 | Only PLL circuit 1 operates. Operate PLL circuit 1 when operating with a 1/4 ϕ cycle lag of the clock input from the CKIO pin and the internal clocks (Iø, Eø, Pø) with respect to system clock ø. PLL circuit 2 does not operate in this mode<br>PLL circuit 1 can be switched between the operating and halted states by means of control bits in the frequency modification register (FMR). However, clock phase shifting is not performed when PLL circuit 1 is halted.<br>Normally, mode 4 should be used. | External clock input |
| 6 | PLL circuits 1 and 2 do not operate. Set this mode when a clock having a frequency equal to that of clocks the clock input from the CKIO pin is used | |

The internal clock frequency can be changed in each clock mode (see section 3.2.5, Operating Frequency Selection by Register).

In clock modes 4 to 6, the frequency of the clock input from the CKIO pin can be changed, or the clock can be stopped (see section 22.4.4, Clock Pause Function).

Table 3.3 lists the relationship between pins MD2 to MD0 and the clock operating mode. Do not switch the MD2–MD0 pins while they are operating. Switching will cause operating errors.

**Table 3.3 Clock Mode Pin Settings and States**

| | Pin | | | | | | |
|---|---|---|---|---|---|---|---|
| **Clock Mode** | **MD2** | **MD1** | **MD0** | **$\overline{\text{CKPREQ}}$/CKM** | **EXTAL** | **XTAL** | **CKIO** |
| 0 | 0 | 0 | 0 | 0 | Clock input | Open | Output/high impedance |
| | | | | 1 | Crystal oscillation | Crystal oscillation | |
| 1 | 0 | 0 | 1 | 0 | Clock input | Open | Output/high impedance |
| | | | | 1 | Crystal oscillation | Crystal oscillation | |
| 2 | 0 | 1 | 0 | 0 | Clock input | Open | Output/high impedance |
| | | | | 1 | Crystal oscillation | Crystal oscillation | |
| 3 | 0 | 1 | 1 | 0 | Clock input | Open | High impedance |
| | | | | 1 | Crystal oscillation | Crystal oscillation | |
| 4 | 1 | 0 | 0 | * | Open | Open | Clock input |
| 5 | 1 | 0 | 1 | | Open | Open | Clock input |
| 6 | 1 | 1 | 0 | | Open | Open | Clock input |

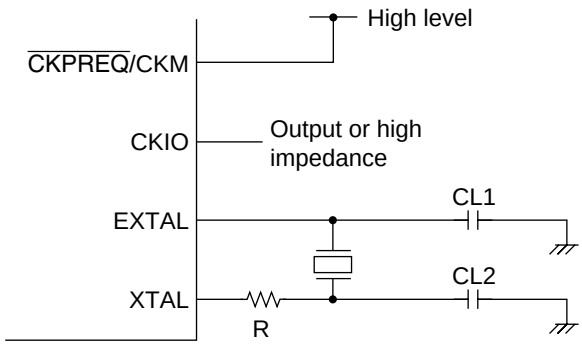
Note: Do not use in combinations other than those listed.

* In clock modes 4, 5, and 6, $\overline{\text{CKPREQ}}$/CKM functions as the clock pause request pin.

### 3.2.3 Connecting a Crystal Resonator

**Connecting a Crystal Resonator:** Figure 3.2 shows an example of crystal resonator connection. The values of damping resistance R and load capacitances CL1 and CL2 should be decided after investigating the components in collaboration with the manufacturer of the crystal oscillator to be used. The crystal resonator should be an AT-cut parallel-oscillator type. 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 to prevent induction from interfering with correct oscillation.




Notes: 1. The CKIO pin is an output in clock modes 0,1, and 2. In mode 3, it is high impedance.
2. The values for CL1, CL2, and the damping resistance should be determined after consultation with the crystal resonator manufacturer.

**Figure 3.2 Example of Crystal Oscillator Connection**

### 3.2.4 External Clock Input

An external clock is input from the EXTAL pin or the CKIO pin, depending on the clock mode.

**Clock Input from EXTAL Pin:** This method can be used in clock modes 0, 1, 2, and 3.

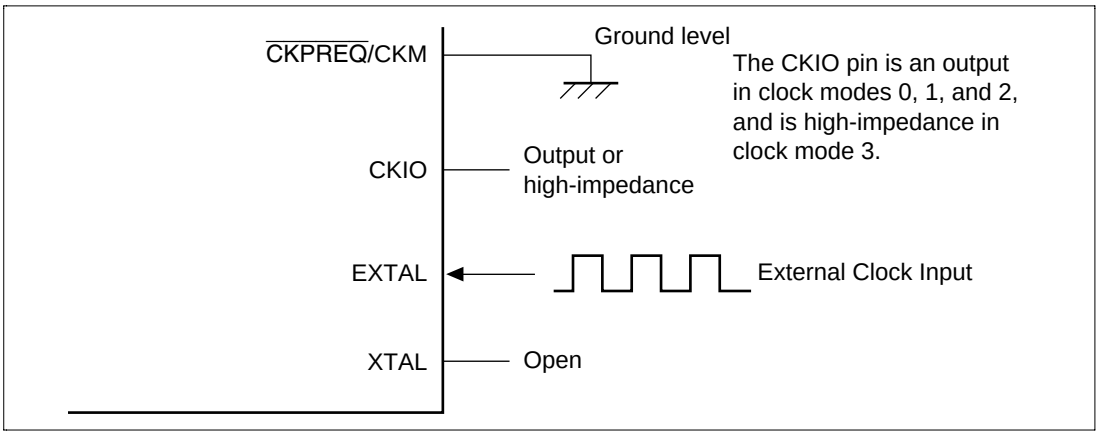



**Figure 3.3 External Clock Input Method**

**Clock Input from CKIO Pin:** This method can be used in clock modes 4, 5, and 6.

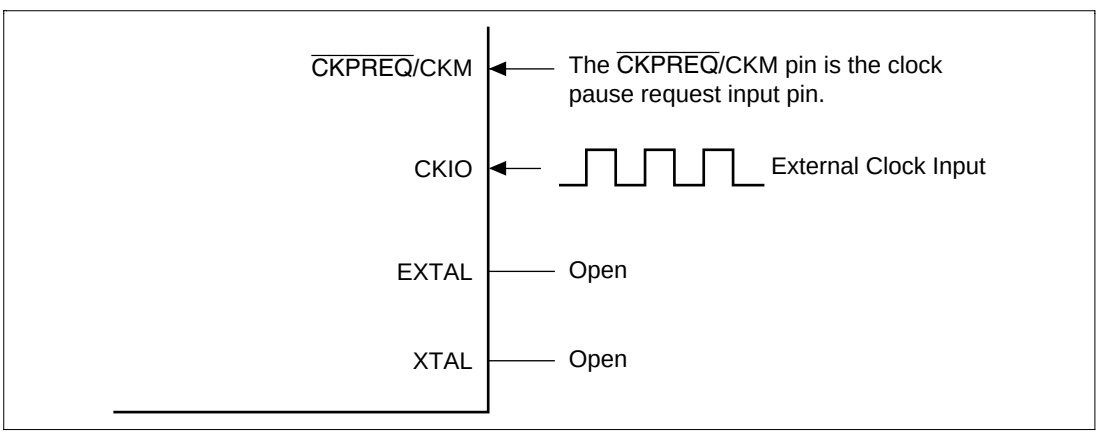



**Figure 3.4 External Clock Input Method**

### 3.2.5 Operating Frequency Selection by Register

Using the frequency modification register (FMR), it is possible to specify the operating frequency division ratio for the internal clocks (Iϕ, Eϕ, Pϕ). The internal clock frequency is determined under the control of PLL circuits 1 and 2 and dividers DIVM, DIVE, and DIVP.

**Frequency Modification Register (FMR):** The frequency modification register is initialized only by a power-on reset via the $\overline{\text{RES}}$ pin, and not by an internal reset resulting from WDT overflow. Its initial value depends on the settings of pins MD2–MD0. Table 3.4 shows the relationship between the MD2–MD0 pin combinations and the initial value of the frequency modification register.

**Table 3.4 Relationship between Clock Mode Pin Settings and Initial Value of Frequency Modification Register**

| Clock Mode | MD2 | MD1 | MD0 | Initial Value |
|---|---|---|---|---|
| Mode 0 | 0 | 0 | 0 | H'00 |
| Mode 1 | 0 | 0 | 1 | |
| Mode 2 | 0 | 1 | 0 | H'40 |
| Mode 3 | 0 | 1 | 1 | H'60 |
| Mode 4 | 1 | 0 | 0 | H'A6 |
| Mode 5 | 1 | 0 | 1 | |
| Mode 6 | 1 | 1 | 0 | H'E0 |

The register configuration is shown in table 3.5.

**Table 3.5 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| Frequency modification register | FMR | R/W | See table 3.4 | H'FFFFFE90 |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PLL2ST | PLL1ST | CKIOST | — | FR3 | FR2 | FR1 | FR0 |
| Reset: | — | — | — | 0 | 0 | — | — | 0 |
| R/W: | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |

Bit 7—PLL2ST: Switching is possible in modes 0 to 3. In modes 4 to 6, PLL circuit 2 cannot be used. In these modes, this bit always reads 1.

| Bit 7: PLL2ST | Description |
|---|---|
| 0 | PLL circuit 2 used |
| 1 | PLL circuit 2 not used |

Bit 6—PLL1ST: Switching is possible in modes 0, 1, 4, and 5. In modes 2, 3, and 6, PLL circuit 1 cannot be used. In these modes, this bit always reads 1.

| Bit 6: PLL1ST | Description |
|---|---|
| 0 | PLL circuit 1 is used |
| 1 | PLL circuit 1 is not used |

Bit 5—CKIOST: Setting is possible in modes 0 to 3. In modes 4 to 6, the CKIO pin is an input pin. In these modes, this bit always reads 1.

| Bit 5: CKIOST | Description |
|---|---|
| 0 | The CKIO pin outputs Eϕ |
| 1 | The CKIO pin is in the high-impedance state (Do not place CKIO in the high-impedance state when PLL circuit 1 is operating) |

Bit 4—Reserved: This bit always reads 0. The write value should always be 0.

Bits 3 to 0—FR3 to FR0: The internal clock frequency and CKIO output frequency (modes 0–2) can be set by frequency setting bits FR3–FR0. The values that can be set in bits FR3–FR0 depend on the mode and whether PLL circuit 1 and PLL circuit 2 are operating or halted. The following tables show the values that can be set in FR3–FR0, and the internal clock and CKIO output frequency ratios, taking the external input clock frequency as 1.

- Modes 0 and 1
  PLL circuits 1 and 2 operating
  EXTAL input or crystal resonator used

| FR3 | FR2 | FR1 | FR0 | φ | Iφ | Eφ | Pφ | CKIO |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ×4 | ×1 | ×1 | ×1 | Eφ |
| 0 | 1 | 0 | 0 | ×4 | ×2 | ×1 | ×1 | Eφ |
| 0 | 1 | 0 | 1 | ×4 | ×2 | ×2 | ×1 | Eφ |
| 0 | 1 | 1 | 0 | ×4 | ×2 | ×2 | ×2 | Eφ |
| 1 | 0 | 0 | 0 | ×4 | ×4 | ×1 | ×1 | Eφ |
| 1 | 0 | 0 | 1 | ×4 | ×4 | ×2 | ×1 | Eφ |
| 1 | 0 | 1 | 0 | ×4 | ×4 | ×2 | ×2 | Eφ |
| 1 | 1 | 0 | 0 | ×4 | ×4 | ×4 | ×1 | Eφ |
| 1 | 1 | 1 | 0 | ×4 | ×4 | ×4 | ×2 | Eφ |

Note: Do not use combinations other than those shown above.

- Modes 0 to 3
  PLL circuit 1 halted, PLL circuit 2 operating
  EXTAL input or crystal resonator used

| FR3 | FR2 | FR1 | FR0 | φ | Iφ | Eφ | Pφ | CKIO |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ×1 | ×1 | ×1 | ×1 | Eφ |
| 0 | 1 | 0 | 1 | ×2 | ×2 | ×2 | ×1 | Eφ |
| 0 | 1 | 1 | 0 | ×2 | ×2 | ×2 | ×2 | Eφ |
| 1 | 1 | 0 | 0 | ×4 | ×4 | ×4 | ×1 | Eφ |
| 1 | 1 | 1 | 0 | ×4 | ×4 | ×4 | ×2 | Eφ |

Note: Do not use combinations other than those shown above.

- Modes 0 and 1
  PLL circuit 1 operating, PLL circuit 2 halted
  EXTAL input or crystal resonator used

| FR3 | FR2 | FR1 | FR0 | φ | Iφ | Eφ | Pφ | CKIO |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ×4 | ×1 | ×1 | ×1 | Eφ |
| 0 | 1 | 0 | 0 | ×4 | ×2 | ×1 | ×1 | Eφ |
| 1 | 0 | 0 | 0 | ×4 | ×4 | ×1 | ×1 | Eφ |

Note: Do not use combinations other than those shown above.

- Modes 4 and 5
  PLL circuit 1 operating, PLL circuit 2 halted
  CKIO input

| FR3 | FR2 | FR1 | FR0 | φ | Iφ | Eφ | Pφ | CKIO |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | ×2 | ×1 | ×1 | ×1/2 | Eφ |
| 0 | 1 | 1 | 0 | ×2 | ×1 | ×1 | ×1 | Eφ |
| 1 | 0 | 0 | 1 | ×2 | ×2 | ×1 | ×1/2 | Eφ |
| 1 | 0 | 1 | 0 | ×2 | ×2 | ×1 | ×1 | Eφ |

Note: Do not use combinations other than those shown above.

- Modes 0 to 6

  PLL circuits 1 and 2 halted

  EXTAL input or crystal resonator used (modes 0 to 3)

  CKIO input (modes 4 to 6)

| FR3 | FR2 | FR1 | FR0 | ϕ | Iϕ | Eϕ | Pϕ | CKIO |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ×1 | ×1/4 | ×1/4 | ×1/4 | ×1 |
| 0 | 1 | 0 | 0 | ×1 | ×1/2 | ×1/4 | ×1/4 | ×1 |
| 0 | 1 | 0 | 1 | ×1 | ×1/2 | ×1/2 | ×1/4 | ×1 |
| 0 | 1 | 1 | 0 | ×1 | ×1/2 | ×1/2 | ×1/2 | ×1 |
| 1 | 0 | 0 | 0 | ×1 | ×1 | ×1/4 | ×1/4 | ×1 |
| 1 | 0 | 0 | 1 | ×1 | ×1 | ×1/2 | ×1/4 | ×1 |
| 1 | 0 | 1 | 0 | ×1 | ×1 | ×1/2 | ×1/2 | ×1 |
| 1 | 1 | 0 | 0 | ×1 | ×1 | ×1 | ×1/4 | ×1 |
| 1 | 1 | 1 | 0 | ×1 | ×1 | ×1 | ×1/2 | ×1 |
| 1 | 1 | 1 | 1 | ×1 | ×1 | ×1 | ×1 | ×1 |

Note: Do not use combinations other than those shown above.

**Frequency Change:** When PLL circuit 1 or PLL circuit 2 becomes operational after modifying the frequency modification register (including modification the frequency modification register in the operating state), access the frequency modification register using the following procedure, and noting the cautions listed below.

Frequency change procedure

- Set the on-chip watchdog timer (WDT) overflow time to secure the PLL circuit oscillation settling time (CKS[2–0] bits in WTCSR).
- Clear the WT/$\overline{IT}$ and TME bit to 0 in WTCSR.
- Perform a read anywhere in an external memory area 0–4 cache-through area.
- Change the frequency modification register to the target frequency, or change the operating/halted state of the PLL circuits 1 and 2 (the entire chip will stop for a maximum of 5 peripheral module clock (Pø) cycles).
- The oscillation circuits operate, and the clock is supplied to the WDT. This clock increments the WDT.
- On WDT overflow, supply of a clock with the frequency set in frequency setting bits FR3–FR0 begins. In this case, the OVF bit in WTSCR and the WOVF bit in RSTCSR are not set, an interval timer interrupt (ITI) is not requested, and the $\overline{WDTOVF}$ signal is not asserted.

- The read from the external memory space 0–4 cache-through area and the write to the frequency modification register should be performed in on-chip X/Y memory or cache memory. After reading from the external memory space 0–4 cache-through area, do not perform any write operations in external memory spaces 0–4 until the write to the frequency modification register.
- When performing frequency modification in on-chip X/Y memory, place the instruction that rewrites the frequency modification register at address 4n, and read the frequency modification register immediately afterwards.
- When the program that performs frequency modification is executed in on-chip cache memory, the following procedure should be used.
  - — Writes to the frequency modification register should be carried out when the cache is disabled.
  - — Place the frequency modification program in the on-chip cache memory and execute it using data array forced access space. The frequency modification flowchart is shown in figure 1.
  - — When the frequency modification program is run, execute an associative purge or forced purge for entries in the data array used.
  - — The instruction that writes to the frequency modification register must be immediately followed by at least eight consecutive NOP instructions.

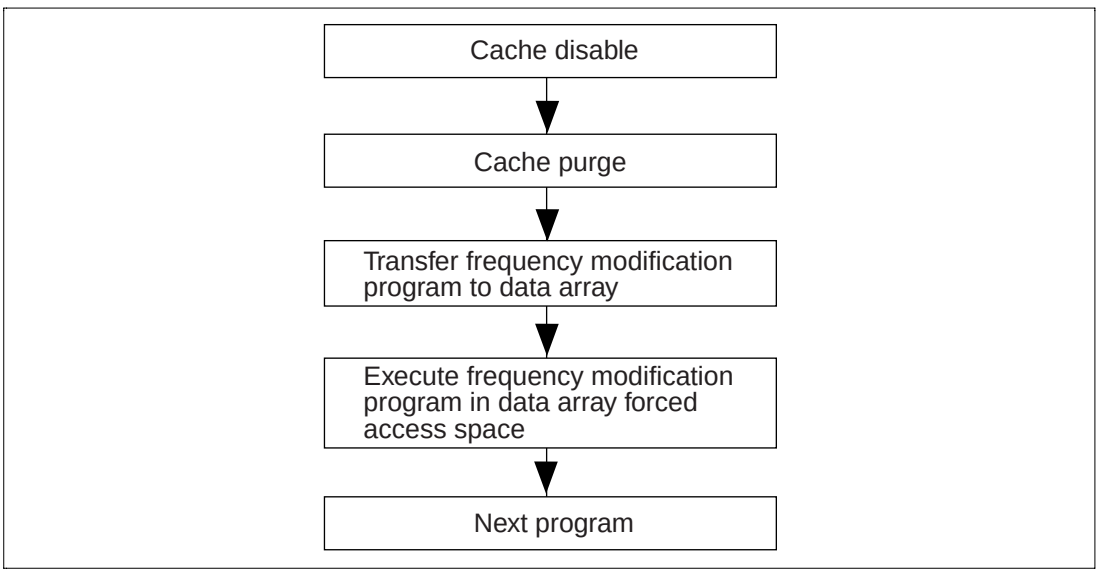



**Figure 3.5 Frequency Modification Flowchart**

- When the write access to the frequency modification register is executed, the WDT starts automatically.

- Do not turn off the CKIO output when PLL circuit 1 is in the operating state.
- The CKIO output will be unstable until the PLL circuit stabilizes.

If PLL circuit 1 or PLL circuit 2 does not become operational after modifying the frequency modification register (including modification in the operating state), it means that the above procedure or cautions have not been properly observed. In this case, the WDT will not operate even though the frequency modification register is modified.

Frequency modification register setting program (sample)

Executed in on-chip XRAM

```
; Define constant
DMAOR       .EQU    H'FFFFFFB0
WTCSR       .EQU    H'FFFFFE80
FMR         .EQU    H'FFFFFE90
XRAM        .EQU    H'1000E000

; ============================================================

    .ORG    H'00000450
PROG:

; ------------------------------------------------------------
;     Transfer frequency modification program to on-chip XRAM
; ------------------------------------------------------------

    MOV.L   #FMR_START,R0  ;
    MOV.L   #XRAM,R1       ;
    MOV.L   #FMR_END,R3    ;

PROG_TRNS:
    MOV.L   @R0+,R2        ; Read from main memory
    MOV.L   R2,@R1         ; Write to XRAM
    ADD #4,R1              ; Increment pointer

    CMP/GT  R3,R0          ; Determine end of loop
    BF  PROG_TRNS          ;
```

```
; ------------------------------------------------------------
;   Branch to on-chip XRAM area
; ------------------------------------------------------------

    MOV.L   #XRAM,R0;
    JMP @R0;
    NOP;
    .CONST

; ============================================================
;   Program transferred to on-chip RAM
; ============================================================

FMR_START:

; ------------------------------------------------------------
;   Halt DMA
; ------------------------------------------------------------

    MOV.L   #DMAOR,R0;
    MOV #0,R1;
    MOV.W   R1,@R0;

; ------------------------------------------------------------
;   WDT setting
; ------------------------------------------------------------

    MOV.L   #WTCSR,R0       ;
    MOV.W   #H'A51B,R1      ; Set ≥ PLL/crystal resonator oscillation stabilization time
    MOV.W   R1,@R0          ;

; ------------------------------------------------------------
;   Frequency modification register setting
; ------------------------------------------------------------

    MOV.L   #H'20000000,R0  ;
    MOV.L   @R0,R0          ; Dummy read from cache-through area
```

```
    MOV.L   #FMR,R0          ;
    MOV #H'0A,R1             ; Set Iø:Eø:Pø = 4:2:2
    BRA FMR_W                ; Branch to write to FMR located at 4n boundary
    NOP                      ;

; Place FMR write at 4n, perform FMR read from 4n+2
; Use .ORG specification to specify address

    .ORG    H'00000550
FMR_W:  MOV.B   R1,@R0       ; Write to FMR
    MOV.B   @R0,R1           ; Read from FMR

; ------------------------------------------------------------
;   Branch to next program
; ------------------------------------------------------------

    MOV.L   #NEXT_PROG,R0;
    JMP @R0;
    NOP;
    .CONST
FMR_END:

; ============================================================
;   Next program
; ============================================================

NEXT_PROG:
        :
        :
        :
```

Frequency modification register setting program (sample)

Executed in on-chip cache memory

```
; Define constant
DMAOR       .EQU    H'FFFFFFB0
WTCSR       .EQU    H'FFFFFE80
FMR         .EQU    H'FFFFFE90
DIRECT_RW   .EQU    H'C0000000
CCR         .EQU    H'FFFFFE92

; ============================================================

    .ORG    H'00000450
PROG:

; ------------------------------------------------------------
;   CCR save, cache disable, forced purge
; ------------------------------------------------------------

    MOV.L   #CCR,R0         ; CCR save
    MOV.B   @R0,R14         ;
    MOV #0,R1               ; Disable setting
    MOV.B   R1,@R0          ;
    MOV #H'10,R1            ; Forced purge setting
    MOV.B   R1,@R0          ;

; ------------------------------------------------------------
;   Transfer frequency modification program to on-chip XRAM
; ------------------------------------------------------------

    MOV.L   #FMR_START,R0   ;
    MOV.L   #DIRECT_RW,R1   ;
    MOV.L   #FMR_END,R3     ;

;------------------------------------------------------------

PROG_TRNS:
    MOV.L   @R0+,R2         ; Read from main memory
```

```
    MOV.L   R2,@R1          ; Write to data array
    ADD #4,R1               ; Increment pointer

    CMP/GT  R3,R0           ; Determine end of loop
    BF  PROG_TRNS           ;

; ------------------------------------------------------------
;   Branch to data array forced access space
; ------------------------------------------------------------

    MOV.L   #DIRECT_RW,R0;
    JMP @R0;
    NOP;
    .CONST

; ============================================================
;   Program transferred to on-chip cache memory
; ============================================================

FMR_START:

; ------------------------------------------------------------
;   Halt DMA
; ------------------------------------------------------------

    MOV.L   #DMAOR,R0;
    MOV #0,R1;
    MOV.W   R1,@R0;

; ------------------------------------------------------------
;   WDT setting
; ------------------------------------------------------------

    MOV.L   #WTCSR,R0       ;
    MOV.W   #H'A51B,R1      ; Set ≥ PLL/crystal resonator oscillation stabilization time
    MOV.W   R1,@R0          ;
```

```
; ------------------------------------------------------------
;    Frequency modification register setting
; ------------------------------------------------------------

    MOV.L   #H'20000000,R0 ;
    MOV.L   @R0,R0          ; Dummy read from cache-through area

    MOV.L   #FMR,R0         ;
    MOV #H'0A,R1            ; Set Iø:Eø:Pø = 4:2:2
FMR_W:  MOV.B   R1,@R0      ; Write to FMR

    NOP ;
    NOP ;
    NOP ;
    NOP ;
    NOP ;
    NOP ;
    NOP ;
    NOP ;

; ------------------------------------------------------------
;    Branch to next program
; ------------------------------------------------------------

    MOV.L   #NEXT_PROG,R0 ;
    JMP @R0 ;
    NOP ;
    .CONST
FMR_END:

; ============================================================
;    Next program
; ============================================================

NEXT_PROG:

    MOV.L   #CCR,R0         ; CCR load
```

```
MOV.B   R14,@R0         ;
        :
        :
        :
```

### 3.2.6 Clock Modes and Frequency Ranges

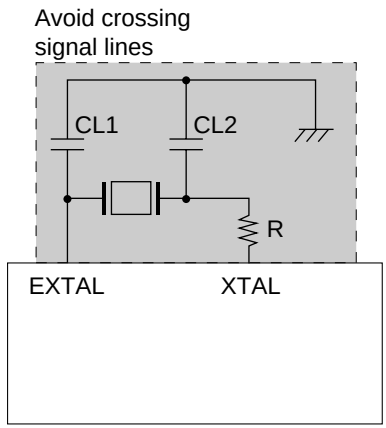
The following table shows the operating modes and the associated frequency ranges for input clocks.

| | Clock Input | | PLL Circuit | | Internal Clock*2,*3 | | | |
|---|---|---|---|---|---|---|---|---|
| Mode | Pin | Input Frequency Range (MHz) | PLL1 | PLL2 | Iϕ (MHz) | Eϕ (MHz) | Pϕ (MHz) | CKIO Output (MHz) |
| 0, 1 | EXTAL or crystal resonator*1 | 8–15 | On | On | 8–60 | 8–60 | 8–30 | 8–60 |
| | | | Off | On | 8–60 | 8–60 | 8–30 | 8–60 |
| | | | On | Off | 8–60 | 8–15 | 8–15 | 8–15 |
| | | 1–30 | Off | Off | 1–30 | 1–30 | 1–30 | 1–30 |
| 2 | | 8–15 | Off | On | 8–60 | 8–60 | 8–30 | 8–60 |
| | | 1–30 | | Off | 1–30 | 1–30 | 1–30 | 1–30 |
| 3 | | 8–15 | | On | 8–60 | 8–60 | 8–30 | — |
| | | 1–30 | | Off | 1–30 | 1–30 | 1–30 | |
| 4, 5 | CKIO | 16–30 | On | Off | 16–60 | 16–30 | 16–30 | — |
| | | 1–30 | Off | | 1–30 | 1–30 | 1–30 | |
| 6 | | 1–30 | Off | | 1–30 | 1–30 | 1–30 | |

Notes: 1. When a crystal resonator is used, set the frequency in the range of 8 to 15 MHz.
2. Set the frequency modification register so that the frequency of all internal clocks is 1 MHz or higher.
3. Use internal clock frequencies such that $I\phi \geq E\phi \geq P\phi$.

### 3.2.7 Notes on Board Design

**When Using an External Crystal Oscillator:** Place the crystal resonator, capacitors CL1 and CL2, and damping resistor R close to the EXTAL and XTAL pins. To prevent induction from interfering with correct oscillation, use a common grounding point for the capacitors connected to the resonator, and do not locate a wiring pattern near these components.




Note: The values for CL1, CL2, and the damping resistance should be determined after consultation with the crystal resonator manufacturer.

**Figure 3.6 Points for Attention when Using Crystal Resonator**

**Bypass Capacitors:** As far as possible, insert a laminated ceramic capacitor of 0.01 to 0.1 μF as a bypass capacitor for each $V_{SS}/V_{CC}$ pair. Mount the bypass capacitors as close as possible to the LSI power supply pins, and use components with a frequency characteristic suitable for the LSI operating frequency, as well as a suitable capacitance value.

Digital system $V_{SS}/V_{CC}$ pairs: 13-11, 28-26, 36-34, 44-42, 52-50, 60-58, 75-73, 88-86, 101-99, 109-107, 119-117, 127-125, 140-138, 153-151, 168-166, 21-19, 113-111, 157-155, 4,5-3*

PLL system $V_{SS}/V_{CC}$ pair: 8-6

Note: * Do not connect a bypass capacitor to these pins when performing debugging with an E10 emulator (series type E10).

**When Using a PLL Oscillator Circuit:** Keep the wiring short from the PLL $V_{CC}$ and $V_{SS}$ connection pattern to the power supply pins, and make the pattern width large, to minimize the inductance component. Ground the oscillation stabilization capacitors C1 and C2 to $V_{SS}$ (PLL1) and $V_{SS}$ (PLL2), respectively. Place C1 and C2 close to the CAP1 and CAP2 pins and do not locate a wiring pattern in the vicinity.

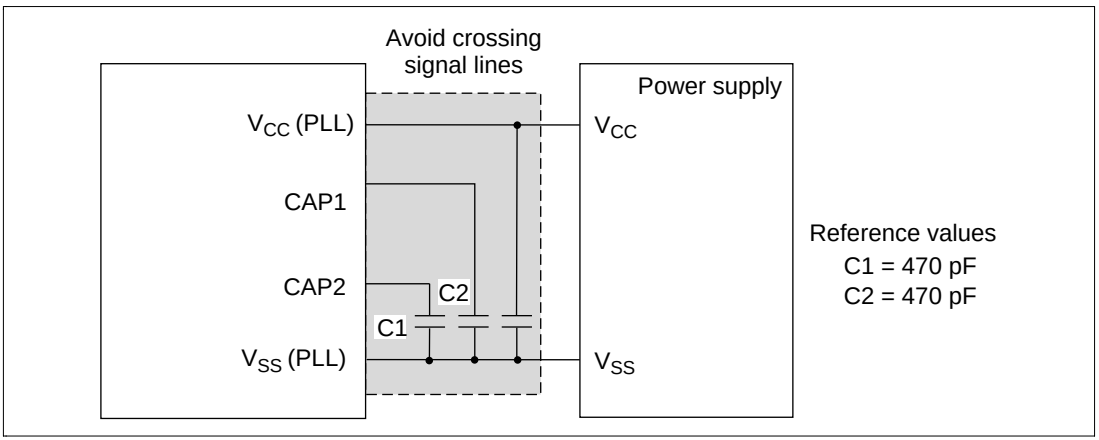



**Figure 3.7 Points for Attention when Using PLL Oscillator Circuit**

## 3.3 Bus Width of the CS0 Area

Pins MD3 and MD4 are used to specify the bus width of the CS0 area. The pin combination and functions are listed in table 3.6. Do not switch the MD4 and MD3 pins while they are operating. Switching them will cause operating errors.

**Table 3.6 Bus Width of the CS0 Area**

| Pin | | |
|---|---|---|
| **MD4** | **MD3** | **Function** |
| 0 | 0 | 8-bit bus width selected |
| 0 | 1 | 16-bit bus width selected |
| 1 | 0 | 32-bit bus width selected |
| 1 | 1 | Setting prohibited |

# Section 4 Exception Handling

## 4.1 Overview

### 4.1.1 Types of Exception Handling and Priority Order

Exception handling is initiated by four sources: resets, address errors, interrupts, and instructions (table 4.1). When several exception sources occur simultaneously, they are accepted and processed according to the priority order shown in table 4.1.

**Table 4.1 Types of Exception Handling and Priority Order**

| Exception | Source | | Priority |
|---|---|---|---|
| Reset | Power-on reset | | High |
| | Manual reset | | ↑ |
| Address error | CPU address error | | |
| | DMA address error | | |
| Interrupt | NMI | | |
| | User break | | |
| | Hitachi user debug interface (H-UDI) | | |
| | External interrupts (IRL1–IRL15, IRQ0–IRQ3 (set with $\overline{\text{IRL3}}$, $\overline{\text{IRL2}}$, $\overline{\text{IRL1}}$, $\overline{\text{IRL0}}$ pins)) | | |
| | On-chip peripheral modules | Division unit (DIVU) | |
| | | Direct memory access controller (DMAC) | |
| | | Watchdog timer (WDT) | |
| | | Compare match interrupt (part of the bus state controller) | |
| | | Serial communication interface (SCI) | |
| | | 16-bit free-running timer (FRT) | |
| | | Serial communication interface with FIFO (SCIF) | |
| | | 16-bit timer pulse unit (TPU) | |
| | | Serial I/O (SIO) | |
| Instructions | Trap instruction (TRAPA) | | |
| | General illegal instructions (undefined code) | | ↓ |
| | Illegal slot instructions (undefined code placed directly following a delayed branch instruction*1 or instructions that rewrite the PC*2) | | Low |

Notes: 1. Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF

2. Instructions that rewrite the PC: JMP, JSR, BRA, BSR, RTS, RTE, BT, BF, TRAPA, BF/S, BT/S, BSRF, BRAF

### 4.1.2 Exception Handling Operations

Exception handling sources are detected, and exception handling started, according to the timing shown in table 4.2.

**Table 4.2 Timing of Exception Source Detection and Start of Exception Handling**

| Exception Source | | Timing of Source Detection and Start of Handling |
|---|---|---|
| Reset | Power-on reset | Starts when the NMI pin is high and the $\overline{\text{RES}}$ pin changes from low to high |
| | Manual reset | Starts when the NMI pin is low and the $\overline{\text{RES}}$ pin changes from low to high |
| Address error | | Detected when instruction is decoded and starts when the previous executing instruction finishes executing |
| Interrupts | | Detected when instruction is decoded and starts when the previous executing instruction finishes executing |
| Instructions | Trap instruction | Starts from the execution of a TRAPA instruction |
| | General illegal instructions | Starts from the decoding of undefined code anytime except after a delayed branch instruction (delay slot) |
| | Illegal slot instructions | Starts from the decoding of undefined code placed directly following a delayed branch instruction (delay slot) or of an instruction that rewrites the PC |

When exception handling starts, the CPU operates as follows:

1. Exception handling triggered by reset

   The initial values of the program counter (PC) and stack pointer (SP) are fetched from the exception vector table (PC and SP are respectively addresses H'00000000 and H'00000004 for a power-on reset and addresses H'00000008 and H'0000000C addresses for a manual reset). See section 4.1.3, Exception Vector Table, for more information. 0 is then written to the vector base register (VBR) and 1111 is written to the interrupt mask bits (I3–I0) of the status register (SR). The program begins running from the PC address fetched from the exception vector table.
2. Exception handling triggered by address errors, interrupts, and instructions

   SR and PC are saved to the stack address indicated by R15. For interrupt exception handling, the interrupt priority level is written to the SR's interrupt mask bits (I3–I0). For address error and instruction exception handling, the I3–I0 bits are not affected. The start address is then fetched from the exception vector table and the program begins running from that address.

### 4.1.3 Exception Vector Table

Before exception handling begins, the exception vector table must be written in memory. The exception vector table stores the start addresses of exception service routines. (The reset exception table holds the initial values of PC and SP.)

All exception sources are given different vector numbers and vector table address offsets, from which the vector table addresses are calculated. In exception handling, the start address of the exception service routine is fetched from the exception vector table as indicated by the vector table address.

Table 4.3 lists the vector numbers and vector table address offsets. Table 4.4 shows vector table address calculations.

**Table 4.3 (a) Exception Vector Table**

| Exception Source | | Vector Number | Vector Table Address Offset | Vector Address |
|---|---|---|---|---|
| Power-on reset | PC | 0 | H'00000000–H'00000003 | Vector number × 4 |
| | SP | 1 | H'00000004–H'00000007 | |
| Manual reset | PC | 2 | H'00000008–H'0000000B | |
| | SP | 3 | H'0000000C–H'0000000F | |
| General illegal instruction | | 4 | H'00000010–H'00000013 | VBR + (vector number × 4) |
| (Reserved by system) | | 5 | H'00000014–H'00000017 | |
| Slot illegal instruction | | 6 | H'00000018–H'0000001B | |
| (Reserved by system) | | 7 | H'0000001C–H'0000001F | |
| | | 8 | H'00000020–H'00000023 | |
| CPU address error | | 9 | H'00000024–H'00000027 | |
| DMA address error | | 10 | H'00000028–H'0000002B | |
| Interrupt | NMI | 11 | H'0000002C–H'0000002F | |
| | User break | 12 | H'00000030–H'00000033 | |
| | H-UDI | 13 | H'00000034–H'00000037 | |
| (Reserved by system) | | 14 | H'00000038–H'0000003B | |
| | | : | : | |
| | | 31 | H'0000007C–H'0000007F | |
| Trap instruction (user vector) | | 32 | H'00000080–H'00000083 | |
| | | : | : | |
| | | 63 | H'000000FC–H'000000FF | |

**Table 4.3 (b) Exception Processing Vector Table (IRQ Mode)**

| Exception Source | | Vector Number | Vector Table Address Offset | Vector Addresses |
|---|---|---|---|---|
| Interrupt | IRQ0 | 64*2 | H'00000100–H'00000103 | |
| | IRQ1 | 65*2 | H'00000104–H'00000107 | |
| | IRQ2 | 66*2 | H'00000108–H'0000010B | |
| | IRQ3 | 67*2 | H'0000010C–H'0000010F | |
| | On-chip peripheral module | 0 | H'00000000–H'00000003 | |
| | | : | : | |
| | | 255*4 | H'000003FC–H'000003FF | |

**Table 4.3 (c) Exception Processing Vector Table (IRL Mode)**

| Exception Source | | Vector Number | Vector Table Address Offset | Vector Addresses |
|---|---|---|---|---|
| Interrupt | IRL1*1 | 64*2 | H'00000100–H'00000103 | VBR + (vector number × 4) |
| | IRL2*1 | 65*2 | H'00000104–H'00000107 | |
| | IRL3*1 | | | |
| | IRL4*1 | 66*2 | H'00000108–H'0000010B | |
| | IRL5*1 | | | |
| | IRL6*1 | 67*2 | H'0000010C–H'0000010F | |
| | IRL7*1 | | | |
| | IRL8*1 | 68*2 | H'00000110–H'00000113 | |
| | IRL9*1 | | | |
| | IRL10*1 | 69*2 | H'00000114–H'00000117 | |
| | IRL11*1 | | | |
| | IRL12*1 | 70*2 | H'00000118–H'0000011B | |
| | IRL13*1 | | | |
| | IRL14*1 | 71*2 | H'0000011C–H'0000011F | |
| | IRL15*1 | | | |
| | On-chip peripheral module*3 | 0*4 | H'00000000–H'00000003 | |
| | | : | : | |
| | | 255*4 | H'000003FC–H'000003FF | |

Notes: 1. When 1110 is input to the $\overline{\text{IRL3}}$, $\overline{\text{IRL2}}$, $\overline{\text{IRL1}}$, and $\overline{\text{IRL0}}$ pins, an IRL1 interrupt results. When 0000 is input, an IRL15 interrupt results.

2. External vector number fetches can be performed without using the auto-vector numbers in this table.

3. The vector numbers and vector table address offsets for each on-chip peripheral module interrupt are given table 5.4, Interrupt Exception Vectors and Priorities, in section 5, Interrupt Controller.
4. Vector numbers are set in the on-chip vector number register. See section 5.3, Register Descriptions, in section 5, Interrupt Controller, section 9, Direct Memory Access Controller, and section 10, Division Unit, for more information.

**Table 4.4 Calculating Exception Vector Table Addresses**

| Exception Source | Vector Table Address Calculation | |
|---|---|---|
| Power-on reset<br>Manual reset | (Vector table address) | = (vector table address offset)<br>= (vector number) × 4 |
| Other exception handling | (Vector table address) | = VBR + (vector table address offset)<br>= VBR + (vector number) × 4 |

Note: VBR: Vector base register
Vector table address offset: See table 4.3.
Vector number: See table 4.3.

## 4.2 Resets

### 4.2.1 Types of Resets

Resets have the highest priority of any exception source. There are two types of resets: manual resets and power-on resets. As table 4.5 shows, both types of resets initialize the internal status of the CPU. In power-on resets, all registers of the on-chip peripheral modules are initialized; in manual resets, registers of all on-chip peripheral modules except the bus state controller (BSC), user break controller (UBC), pin function controller (PFC), and frequency modification register (FMR) are initialized. (Use the power-on reset when turning the power on.)

**Table 4.5 Types of Resets**

| | Conditions for Transition to Reset Status | | Internal Status | |
|---|---|---|---|---|
| **Type** | **NMI Pin** | **$\overline{\text{RES}}$ Pin** | **CPU** | **On-Chip Peripheral Modules** |
| Power-on reset | High | Low | Initialized | Initialized |
| Manual reset | Low | Low | Initialized | Initialized except for BSC, UBC, PFC, and frequency modification register (FMR) |

### 4.2.2 Power-On Reset

When the NMI pin is high and the $\overline{\text{RES}}$ pin is driven low, the device performs a power-on reset. For a reliable reset, the $\overline{\text{RES}}$ pin should be kept low for at least the duration of the oscillation settling time (when the PLL circuit is halted) or for $20tP_{cyc}$ (when the PLL circuit is running). During a power-on reset, the CPU's internal state and all on-chip peripheral module registers are initialized. See appendix B, Pin States, for the state of individual pins in the power-on reset state.

In a power-on reset, power-on reset exception handling starts when the NMI pin is kept high and the $\overline{\text{RES}}$ pin is first driven low for a set period of time and then returned to high. The CPU will then operate as follows:

1. The initial value (execution start address) of the program counter (PC) is fetched from the exception vector table.
2. The initial value of the stack pointer (SP) is fetched from the exception vector table.
3. The vector base register (VBR) is cleared to H'00000000 and the interrupt mask bits (I3–I0) of the status register (SR) are set to H'F (1111).
4. The values fetched from the exception vector table are set in the PC and SP, and the program begins executing.

### 4.2.3 Manual Reset

When the NMI pin is low and the $\overline{\text{RES}}$ pin is driven low, the device executes a manual reset. For a reliable reset, the $\overline{\text{RES}}$ pin should be kept low for at least 20 clock cycles. During a manual reset, the CPU's internal state is initialized. All on-chip peripheral module registers are initialized, except for the bus state controller (BSC), user break controller (UBC), and pin function controller (PFC) registers, and the frequency modification register (FMR). When the chip enters the manual reset state in the middle of a bus cycle, manual reset exception handling does not start until the bus cycle has ended. Thus, manual resets do not abort bus cycles. See appendix B, Pin States, for the state of individual pins in the manual reset state.

In a manual reset, manual reset exception handling starts when the NMI pin is kept low and the $\overline{\text{RES}}$ pin is first kept low for a set period of time and then returned to high. The CPU will then operate in the same way as for a power-on reset.

## 4.3 Address Errors

### 4.3.1 Sources of Address Errors

Address errors occur when instructions are fetched or data read or written, as shown in table 4.6.

**Table 4.6 Bus Cycles and Address Errors**

| Bus Cycle Type | Bus Master | Bus Cycle Description | Address Errors |
|---|---|---|---|
| Instruction fetch | CPU | Instruction fetched from even address*[1] | None (normal) |
| | | Instruction fetched from odd address*[1] | Address error occurs |
| | | Instruction fetched from other than on-chip peripheral module space*[1] | None (normal) |
| | | Instruction fetched from on-chip peripheral module space*[1] | Address error occurs |
| Data read/write | CPU or DMAC | Word data accessed from even address*[1] | None (normal) |
| | | Word data accessed from odd address*[1] | Address error occurs |
| | | Longword data accessed from a longword boundary*[1] | None (normal) |
| | | Longword data accessed from other than a longword boundary*[1] | Address error occurs |
| | | Access of cache purge space, address array read/write space, on-chip peripheral module space, or synchronous DRAM mode setting space by PC-relative addressing | Address error occurs |
| | | Access of cache purge space, address array read/write space, data array read/write space, on-chip peripheral module space, or synchronous DRAM mode setting space by a TAS.B instruction | Address error occurs |
| | | Byte, word, or longword data accessed in on-chip peripheral module space at addresses H'FFFFFC00 to H'FFFFFCFF | None (normal) |
| | | Longword data accessed in on-chip peripheral module space at addresses H'FFFFFE00 to H'FFFFFEFF | Address error occurs |
| | | Word or byte data accessed in on-chip peripheral module space at addresses H'FFFFFE00 to H'FFFFFEFF | None (normal) |
| | | Byte data accessed in on-chip peripheral module space at addresses H'FFFF0000 to H'FFFFF0FF or H'FFFFFF00 to H'FFFFFFFF | Address error occurs |
| | | Word or longword data accessed in on-chip peripheral module space at addresses H'FFFF0000 to H'FFFFF0FF or H'FFFFFF00 to H'FFFFFFFF | None (normal) |

Notes: 1. Address errors do not occur during the synchronous DRAM mode register write cycle.
2. 16-byte DMAC transfers use longword accesses.

### 4.3.2 Address Error Exception Handling

When an address error occurs, address error exception handling begins after the end of the bus cycle in which the error occurred and completion of the executing instruction. The CPU operates as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the start address of the instruction to be executed after the last instruction executed .
3. The exception service routine start address is fetched from the exception vector table entry that corresponds to the address error that occurred, and the program starts executing from that address. The jump that occurs is not a delayed branch.

## 4.4 Interrupts

### 4.4.1 Interrupt Sources

Table 4.7 shows the sources that initiate interrupt exception handling. These are divided into NMI, user breaks, H-UDI, IRL, IRQ, and on-chip peripheral modules.

**Table 4.7 Types of Interrupt Sources**

| Type | Request Source | Number of Sources |
|---|---|---|
| NMI | NMI pin (external input) | 1 |
| User break | User break controller | 1 |
| H-UDI | Hitachi user debug interface (H-UDI) | 1 |
| IRL | IRL1–IRL15 (external input) | 15 |
| IRQ | IRQ0–IRQ3 (external input) | 4 |
| On-chip peripheral module | Direct memory access controller (DMAC) | 2 |
| | Division unit (DIVU) | 1 |
| | Serial communication interface (SCI) | 4 |
| | Free-running timer (FRT) | 3 |
| | Watchdog timer (WDT) | 1 |
| | Bus state controller (BSC) | 1 |
| | Serial I/O (SIO) | 4 |
| | Serial communication interface with FIFO (SCIF) | 4 |
| | 16-bit timer pulse unit (TPU) | 13 |

Each interrupt source is allocated a different vector number and vector table address offset. See table 5.4, Interrupt Exception Vectors and Priority Order, in section 5, Interrupt Controller, for more information.

### 4.4.2 Interrupt Priority Levels

The interrupt priority order is predetermined. When multiple interrupts occur simultaneously, the interrupt controller (INTC) determines their relative priorities and begins exception handling accordingly.

The priority order of interrupts is expressed as priority levels 0–16, with priority 0 the lowest and priority 16 the highest. The NMI interrupt has priority 16 and cannot be masked, so it is always accepted. The user break interrupt priority level is 15 and IRL interrupts have priorities of 1–15. On-chip peripheral module interrupt priority levels can be set freely using the INTC's interrupt priority level setting registers A–E (IPRA–IPRE) as shown in table 4.8. The priority levels that can be set are 0–15. Level 16 cannot be set. For more information on IPRA–IPRE, see sections 5.3.1, Interrupt Priority Level Setting Register A (IPRA), to 5.3.5, Interrupt Priority Level Setting Register E (IPRE).

**Table 4.8 Interrupt Priority Order**

| Type | Priority Level | Comment |
|---|---|---|
| NMI | 16 | Fixed priority level. Cannot be masked |
| User break | 15 | Fixed priority level |
| H-UDI | 15 | Fixed priority level |
| IRL | 1–15 | Set with $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ pins |
| IRQ | 0–15 | Set with interrupt priority level setting register C (IPRC) |
| On-chip peripheral module | 0–15 | Set with interrupt priority level setting registers A, B, D, and E (IPRA, IPRB, IPRD, IPRE) |

### 4.4.3 Interrupt Exception Handling

When an interrupt occurs, its priority level is ascertained by the interrupt controller (INTC). NMI is always accepted, but other interrupts are only accepted if they have a priority level higher than the priority level set in the interrupt mask bits (I3–I0) of the status register (SR).

When an interrupt is accepted, exception handling begins. In interrupt exception handling, the CPU saves SR and the program counter (PC) to the stack. The priority level value of the accepted interrupt is written to SR bits I3–I0. For NMI, however, the priority level is 16, but the value set in I3–I0 is H'F (level 15). Next, the start address of the exception service routine is fetched from the exception vector table for the accepted interrupt, that address is jumped to and execution begins. For more information about interrupt exception handling, see section 5.4, Interrupt Operation.

## 4.5 Exceptions Triggered by Instructions

### 4.5.1 Instruction-Triggered Exception Types

Exception handling can be triggered by a trap instruction, general illegal instruction or illegal slot instruction, as shown in table 4.9.

**Table 4.9 Types of Exceptions Triggered by Instructions**

| Type | Source Instruction | Comment |
|---|---|---|
| Trap instruction | TRAPA | — |
| Illegal slot instruction | Undefined code placed immediately after a delayed branch instruction (delay slot) and instructions that rewrite the PC | Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF<br>Instructions that rewrite the PC: JMP, JSR, BRA, BSR, RTS, RTE, BT, BF, TRAPA, BF/S, BT/S, BSRF, BRAF |
| General illegal instruction | Undefined code anywhere besides in a delay slot | — |

### 4.5.2 Trap Instructions

When a TRAPA instruction is executed, trap instruction exception handling starts. The CPU operates as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the start address of the instruction to be executed after the TRAPA instruction.
3. The exception service routine start address is fetched from the exception vector table entry that corresponds to the vector number specified by the TRAPA instruction. That address is jumped to and the program starts executing. The jump that occurs is not a delayed branch.

### 4.5.3 Illegal Slot Instructions

An instruction placed immediately after a delayed branch instruction is said to be placed in a delay slot. If the instruction placed in the delay slot is undefined code, illegal slot exception handling begins when the undefined code is decoded. Illegal slot exception handling is also started when an instruction that rewrites the program counter (PC) is placed in a delay slot. The exception handling starts when the instruction is decoded. The CPU handles an illegal slot instruction as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the jump address of the delayed branch instruction immediately before the undefined code or the instruction that rewrites the PC.
3. The exception service routine start address is fetched from the exception vector table entry that corresponds to the exception that occurred. That address is jumped to and the program starts executing. The jump that occurs is not a delayed branch.

### 4.5.4 General Illegal Instructions

When undefined code placed anywhere other than immediately after a delayed branch instruction (i.e., in a delay slot) is decoded, general illegal instruction exception handling starts. The CPU handles general illegal instructions in the same way as illegal slot instructions. Unlike processing of illegal slot instructions, however, the program counter value saved is the start address of the undefined code.

## 4.6 When Exception Sources Are Not Accepted

When an address error or interrupt is generated after a delayed branch instruction or interrupt-disabled instruction, it is sometimes not immediately accepted but is stored instead, as described in table 4.10. When this happens, it will be accepted when an instruction for which exception acceptance is possible is decoded.

**Table 4.10 Exception Source Generation Immediately after a Delayed Branch Instruction or Interrupt-Disabled Instruction**

| | Exception Source | |
|---|---|---|
| **Point of Occurrence** | **Address Error** | **Interrupt** |
| Immediately after a delayed branch instruction*1 | Not accepted | Not accepted |
| Immediately after an interrupt-disabled instruction*2 | Accepted | Not accepted |
| A repeat loop comprising up to three instructions (instruction fetch cycle not generated)<br>First instruction or last three instructions in a repeat loop containing four or more instructions | Not accepted | Not accepted |
| Fourth from last instruction in a repeat loop containing four or more instructions | Accepted | Not accepted |

Notes: 1. Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF
2. Interrupt-disabled instructions: LDC, LDC.L, STC, STC.L, LDS, LDS.L, STS, STS.L

### 4.6.1 Immediately after a Delayed Branch Instruction

When an instruction placed immediately after a delayed branch instruction (delay slot) is decoded, neither address errors nor interrupts are accepted. The delayed branch instruction and the instruction located immediately after it (delay slot) are always executed consecutively, so no exception handling occurs between the two.

### 4.6.2 Immediately after an Interrupt-Disabled Instruction

When an instruction immediately following an interrupt-disabled instruction is decoded, interrupts are not accepted. Address errors are accepted.

### 4.6.3 Instructions in Repeat Loops

If a repeat loop comprises up to three instructions, neither exceptions nor interrupts are accepted. If a repeat loop contains four or more instructions, neither exceptions nor interrupts are accepted during the execution cycle of the first instruction or the last three instructions. If a repeat loop contains four or more instructions, address errors only are accepted during the execution cycle of the fourth from last instruction. For more information, see the *SH-1/SH-2/SH-DSP Programming Manual*.

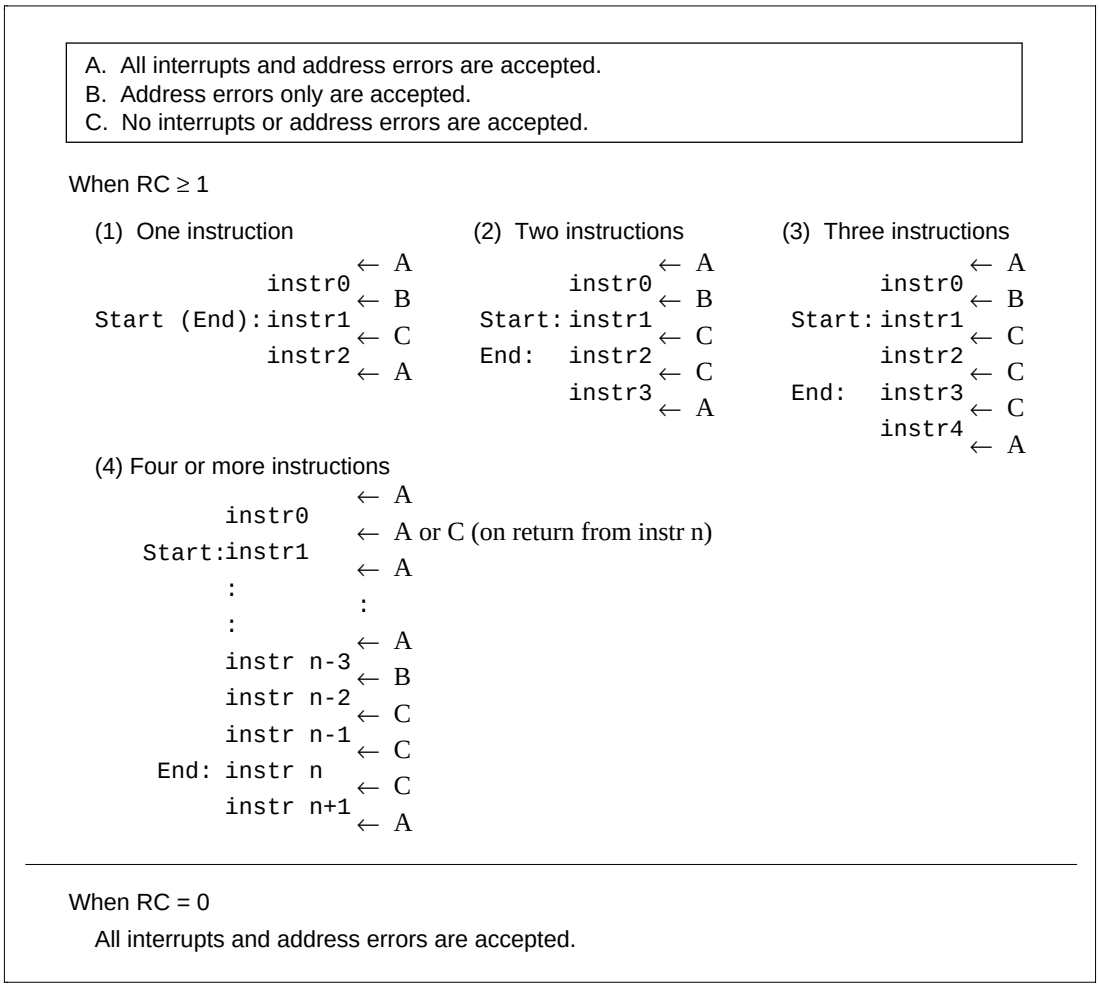



**Figure 4.1 Interrupt Acceptance Restrictions in Repeat Mode**

## 4.7 Stack Status after Exception Handling

The status of the stack after exception handling ends is as shown in table 4.11.

**Table 4.11 Stack Status after Exception Handling**

| Type | Stack Status | | |
|---|---|---|---|
| Address error | SP → | Address of instruction after executed instruction | 32 bits |
| | | SR | 32 bits |
| Trap instruction | SP → | Address of instruction after TRAPA instruction | 32 bits |
| | | SR | 32 bits |
| General illegal instruction | SP → | Start address of illegal instruction | 32 bits |
| | | SR | 32 bits |
| Interrupt | SP → | Address of instruction after executed instruction | 32 bits |
| | | SR | 32 bits |
| Illegal slot instruction | SP → | Jump destination address of delayed branch instruction | 32 bits |
| | | SR | 32 bits |

## 4.8 Usage Notes

### 4.8.1 Value of Stack Pointer (SP)

The value of the stack pointer must always be a multiple of four, otherwise an address error will occur when the stack is accessed during exception handling.

### 4.8.2 Value of Vector Base Register (VBR)

The value of the vector base register must always be a multiple of four, otherwise an address error will occur when the vector table is accessed during exception handling.

### 4.8.3 Address Errors Caused by Stacking of Address Error Exception Handling

If the stack pointer value is not a multiple of four, an address error will occur during stacking of the exception handling (interrupts, etc.). Address error exception handling will begin after the original exception handling ends, but address errors will continue to occur. To ensure that address error exception handling does not go into an endless loop, no address errors are accepted at that point. This allows program control to be shifted to the address error exception service routine and enables error handling to be carried out.

When an address error occurs during exception handling stacking, the stacking bus cycle (write) is executed. In stacking of the status register (SR) and program counter (PC), the SP is decremented by 4 for both, so the value of SP will not be a multiple of four after the stacking either. The address value output during stacking is the SP value, so the address where the error occurred is itself output. This means that the write data stacked will be undefined.

### 4.8.4 Manual Reset during Register Access

Do not initiate a manual reset during access of a bus state controller (BSC), user break controller (UBC), or pin function controller (PFC) register, or the frequency modification register (FMR), otherwise a write error may result.

# Section 5 Interrupt Controller (INTC)

## 5.1 Overview

The interrupt controller (INTC) ascertains the priority order of interrupt sources and controls interrupt requests to the CPU. The INTC has registers for setting the priority of each interrupt which allow the user to set the order of priority in which interrupt requests are handled.

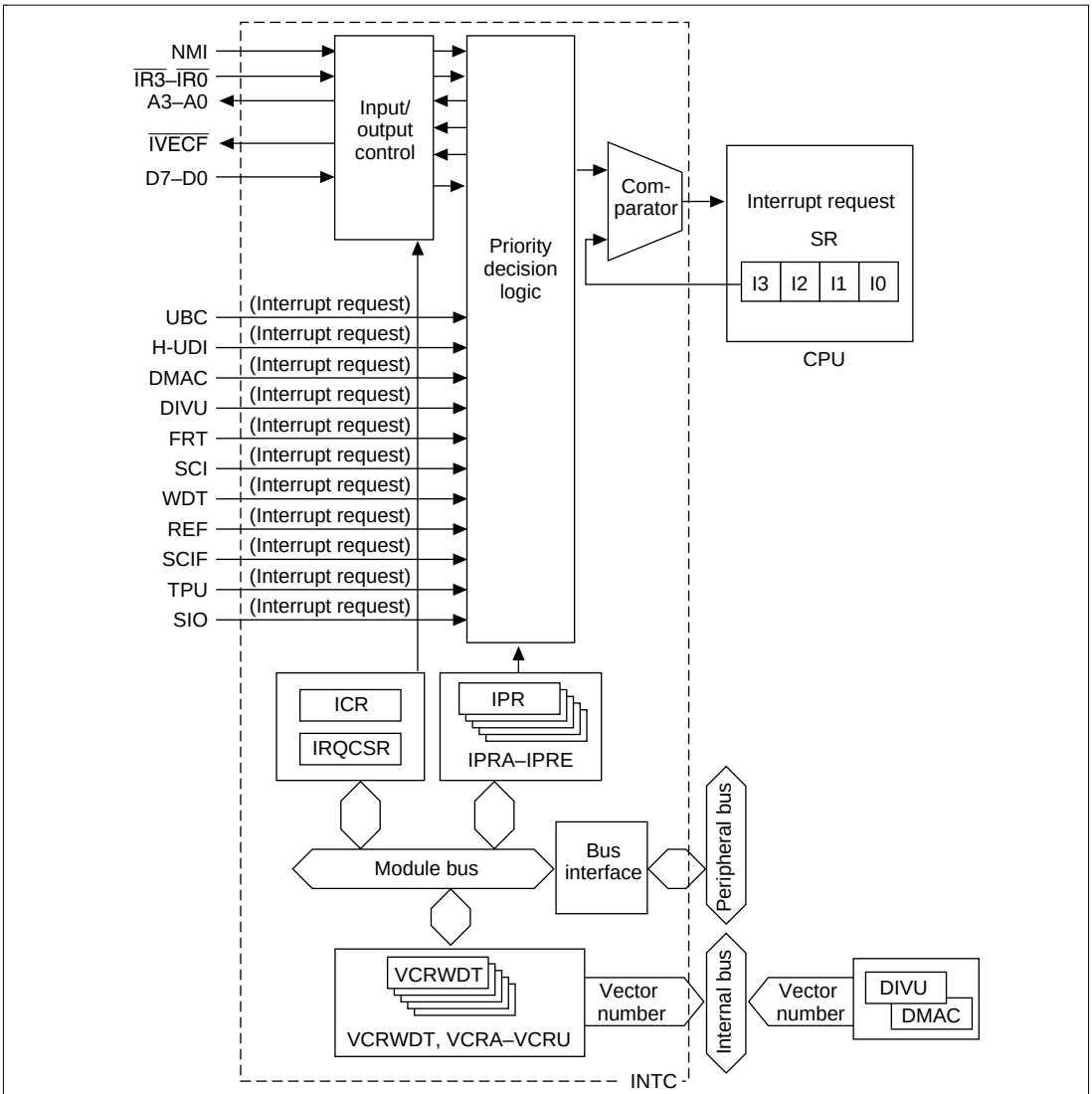
### 5.1.1 Features

The INTC has the following features:

- Sixteen interrupt priority levels can be set
  By setting the five interrupt priority registers, the priorities of on-chip peripheral module interrupts can be selected at 16 levels for different request sources.
- Vector numbers for on-chip peripheral module interrupt can be set
  By setting the 22 vector number setting registers, the vector numbers of on-chip peripheral module interrupts can be set to values from 0 to 127 for different request sources.
- The IRL interrupt vector number setting method can be selected: Either of two modes can be selected by a register setting: auto-vector mode in which vector numbers are determined internally, and external vector mode in which vector numbers are set externally.
- IRQ interrupt settings can be made (low level, rising-, falling-, or both-edge detection)

### 5.1.2 Block Diagram

Figure 5.1 shows a block diagram of the INTC.




UBC: User break controller
H-UDI: Hitachi user debug interface
DMAC: Direct memory access controller
DIVU: Division unit
FRT: Free-running timer
SCI: Serial communication interface
WDT: Watchdog timer
REF: Refresh request within bus state controller
SCIF: Serial communication interface with FIFO
TPU: 16-bit timer pulse unit
SIO: Serial I/O

ICR: Interrupt control register
IRQCSR: IRQ control/status register
IPRA–IPRE: Interrupt priority level setting registers A–E
VCRWDT: Vector number setting register WDT
VCRA–VCRU: Vector number setting registers A–U
SR: Status register

**Figure 5.1 INTC Block Diagram**

### 5.1.3 Pin Configuration

Table 5.1 shows the INTC pin configuration.

**Table 5.1 Pin Configuration**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Nonmaskable interrupt input pin | NMI | I | Input of nonmaskable interrupt request signal |
| Level request interrupt input pins | $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ | I | Input of maskable interrupt request signals |
| Interrupt acceptance level output pins | A3–A0 | O | In external vector mode, output an interrupt level signal when an IRL/IRQ interrupt is accepted |
| External vector fetch pin | $\overline{\text{IVECF}}$ | O | Indicates external vector read cycle |
| External vector number input pins | D7–D0 | I | Input external vector number |

### 5.1.4 Register Configuration

The INTC has the 32 registers shown in table 5.2. These registers perform various INTC functions including setting interrupt priority, and controlling external interrupt input signal detection.

**Table 5.2 Register Configuration**

| Name | Abbr. | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Interrupt priority register setting register A | IPRA | R/W | H'0000 | H'FFFFFEE2 | 8, 16 |
| Interrupt priority register setting register B | IPRB | R/W | H'0000 | H'FFFFFE60 | 8, 16 |
| Interrupt priority register setting register C | IPRC | R/W | H'0000 | H'FFFFFEE6 | 8, 16 |
| Interrupt priority register setting register D | IPRD | R/W | H'0000 | H'FFFFFE40 | 8, 16 |
| Interrupt priority register setting register E | IPRE | R/W | H'0000 | H'FFFFFEC0 | 8, 16 |
| Vector number setting register A | VCRA | R/W | H'0000 | H'FFFFFE62 | 8, 16 |
| Vector number setting register B | VCRB | R/W | H'0000 | H'FFFFFE64 | 8, 16 |
| Vector number setting register C | VCRC | R/W | H'0000 | H'FFFFFE66 | 8, 16 |
| Vector number setting register D | VCRD | R/W | H'0000 | H'FFFFFE68 | 8, 16 |
| Vector number setting register E | VCRE | R/W | H'0000 | H'FFFFFE42 | 8, 16 |
| Vector number setting register F | VCRF | R/W | H'0000 | H'FFFFFE44 | 8, 16 |
| Vector number setting register G | VCRG | R/W | H'0000 | H'FFFFFE46 | 8, 16 |

**Table 5.2 Register Configuration (cont)**

| Name | Abbr. | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Vector number setting register H | VCRH | R/W | H'0000 | H'FFFFFE48 | 8, 16 |
| Vector number setting register I | VCRI | R/W | H'0000 | H'FFFFFE4A | 8, 16 |
| Vector number setting register J | VCRJ | R/W | H'0000 | H'FFFFFE4C | 8, 16 |
| Vector number setting register K | VCRK | R/W | H'0000 | H'FFFFFE4E | 8, 16 |
| Vector number setting register L | VCRL | R/W | H'0000 | H'FFFFFE 50 | 8, 16 |
| Vector number setting register M | VCRM | R/W | H'0000 | H'FFFFFE52 | 8, 16 |
| Vector number setting register N | VCRN | R/W | H'0000 | H'FFFFFE54 | 8, 16 |
| Vector number setting register O | VCRO | R/W | H'0000 | H'FFFFFE56 | 8, 16 |
| Vector number setting register P | VCRP | R/W | H'0000 | H'FFFFFEC2 | 8, 16 |
| Vector number setting register Q | VCRQ | R/W | H'0000 | H'FFFFFEC4 | 8, 16 |
| Vector number setting register R | VCRR | R/W | H'0000 | H'FFFFFEC6 | 8, 16 |
| Vector number setting register S | VCRS | R/W | H'0000 | H'FFFFFEC8 | 8, 16 |
| Vector number setting register T | VCRT | R/W | H'0000 | H'FFFFFECA | 8, 16 |
| Vector number setting register U | VCRU | R/W | H'0000 | H'FFFFFECC | 8, 16 |
| Vector number setting register WDT | VCRWDT | R/W | H'0000 | H'FFFFFEE4 | 8, 16 |
| Vector number setting register DIV | VCRDIV | R/W | Undefined | H'FFFFFF0C | 32 |
| Vector number setting register DMAC0 | VCRDMA0 | R/W | Undefined | H'FFFFFFA0 | 32 |
| Vector number setting register DMAC1 | VCRDMA1 | R/W | Undefined | H'FFFFFFA8 | 32 |
| Interrupt control register | ICR | R/W | H'8000/ H'0000*[1] | H'FFFFFEE0 | 8, 16 |
| IRQ control/status register | IRQCSR | R/W | *[2] | H'FFFFFEE8 | 8, 16 |

Notes: 1. The value when the NMI pin is high is H'8000; when the NMI pin is low, it is H'0000.
See sections 9, Direct Memory Access Controller, and 10, Division Unit, for more information on VCRDIV, VCRDMA0, and VCRDMA1.

2. When pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ are high, bits 7–4 in IRQCSR are set to 1. When pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ are low, bits 7–4 in IRQCSR are cleared to 0. The initial value of bits other than 7–4 is 0.

## 5.2 Interrupt Sources

There are five types of interrupt sources: NMI, user breaks, H-UDI, IRL/IRQ and on-chip peripheral modules. Each interrupt has a priority expressed as a priority level (0 to 16, with 0 the lowest and 16 the highest). Giving an interrupt a priority level of 0 masks it.

### 5.2.1 NMI Interrupt

The NMI interrupt has priority 16 and is always accepted. Input at the NMI pin is detected by edge. Use the NMI edge select bit (NMIE) in the interrupt control register (ICR) to select either the rising or falling edge. NMI interrupt exception handling sets the interrupt mask level bits (I3–I0) in the status register (SR) to level 15.

### 5.2.2 User Break Interrupt

A user break interrupt has priority level 15 and occurs when the break condition set in the user break controller (UBC) is satisfied. User break interrupt exception handling sets the interrupt mask level bits (I3–I0) in the status register (SR) to level 15. For more information about the user break interrupt, see section 6, User Break Controller.

### 5.2.3 H-UDI Interrupt

The H-UDI interrupt has a priority level of 15, and is generated when an H-UDI interrupt instruction is serially input. H-UDI interrupt exception processing sets the interrupt mask bits (I3–I0) in the status register (SR) to level 15. See section 19, Hitachi User Debug Interface, for details of the H-UDI interrupt.

### 5.2.4 IRL Interrupts

IRL interrupts are requested by input from pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$. Fifteen interrupts, IRL15–IRL1, can be input externally via pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$. The priority levels of interrupts IRL15–IRL0 are 15–1, respectively, and their vector numbers are 71–64. Set the vector numbers with the interrupt vector mode select (VECMD) bit of the interrupt control register (ICR) to enable external input. External input of vector numbers consists of vector numbers 0–127 from the external vector input pins (D7–D0). When an external vector is used, 0 is input to D7. Internal vectors are called auto-vectors and vectors input externally are called external vectors. Table 5.3 lists IRL priority levels and auto vector numbers.

When an IRL interrupt is accepted in external vector mode, the IRL interrupt level is output from the interrupt acceptance level output pins (A3–A0). The external vector fetch pin ($\overline{\text{IVECF}}$) is also asserted. The external vector number is read from pins D7–D0 at this time.

IRL interrupt exception processing sets the interrupt mask level bits (I3 to I0) in the status register (SR) to the priority level value of the IRL interrupt that was accepted.

### 5.2.5 IRQ Interrupts

An IRQ interrupt is requested when the external interrupt vector mode select bit (EXIMD) of the interrupt control register (ICR) is set to 1. An IRQ interrupt corresponds to input at one of pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$. Low-level sensing or rising/falling/both-edge sensing can be selected independently for each pin by the IRQ sense select bits (IRQ00S–IRQ31S) in the IRQ control/status register (IRQCSR), and a priority level of 0 to 15 can be selected independently for each pin by means of interrupt priority register C (IPRC). Set the interrupt vector mode select bit (VECMD) of the interrupt control register (ICR) to enable external input of vector numbers. External vector numbers are 0 to 127, and are input to the external vector input pins (D7–D0) during the interrupt vector fetch bus cycle. When an external vector is used, 0 is input to D7.

When an IRQ interrupt is accepted in external vector mode, the IRQ interrupt priority level is output from the interrupt acceptance level output pins (A3–A0). The external vector fetch signal ($\overline{\text{IVECF}}$) is also asserted. The external vector number is read from signals D7–D0 at this time.

IRQ interrupt exception processing sets the interrupt mask bits (I3–I0) in the status register (SR) to the priority level value of the IRQ interrupt that was accepted.

**Table 5.3 IRL Interrupt Priority Levels and Auto-Vector Numbers**

| Pin | | | | Priority | Vector |
|---|---|---|---|---|---|
| $\overline{\text{IRL3}}$ | $\overline{\text{IRL2}}$ | $\overline{\text{IRL1}}$ | $\overline{\text{IRL0}}$ | Level | Number |
| 0 | 0 | 0 | 0 | 15 | 71 |
| | | | 1 | 14 | |
| | | 1 | 0 | 13 | 70 |
| | | | 1 | 12 | |
| | 1 | 0 | 0 | 11 | 69 |
| | | | 1 | 10 | |
| | | 1 | 0 | 9 | 68 |
| | | | 1 | 8 | |
| 1 | 0 | 0 | 0 | 7 | 67 |
| | | | 1 | 6 | |
| | | 1 | 0 | 5 | 66 |
| | | | 1 | 4 | |
| | 1 | 0 | 0 | 3 | 65 |
| | | | 1 | 2 | |
| | | 1 | 0 | 1 | 64 |

An example of connections for external vector mode interrupts is shown in figure 5.2, and an example of connections for auto-vector mode interrupts in figure 5.3.

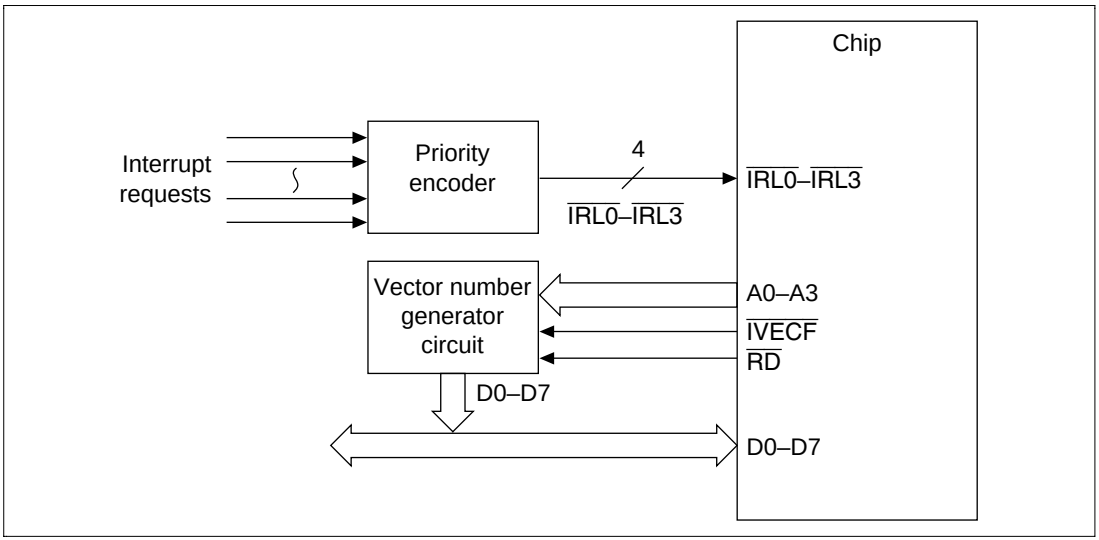



**Figure 5.2 Example of Connections for External Vector Mode Interrupts**

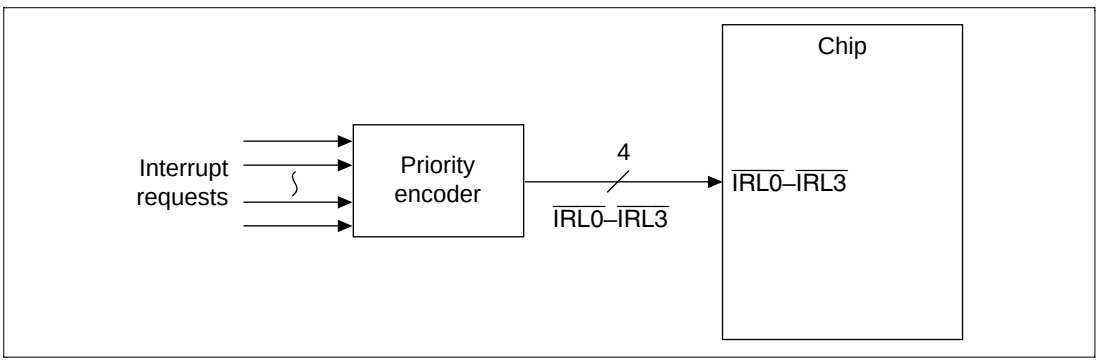



**Figure 5.3 Example of Connections for Auto-Vector Mode Interrupts**

Figures 5.4 to 5.7 show the interrupt vector fetch cycle for the external vector mode. During this cycle, $\overline{CS0}$–$\overline{CS4}$ stay high. A24–A4 output undefined values. The $\overline{WAIT}$ pin is sampled, but programmable waits are not valid.

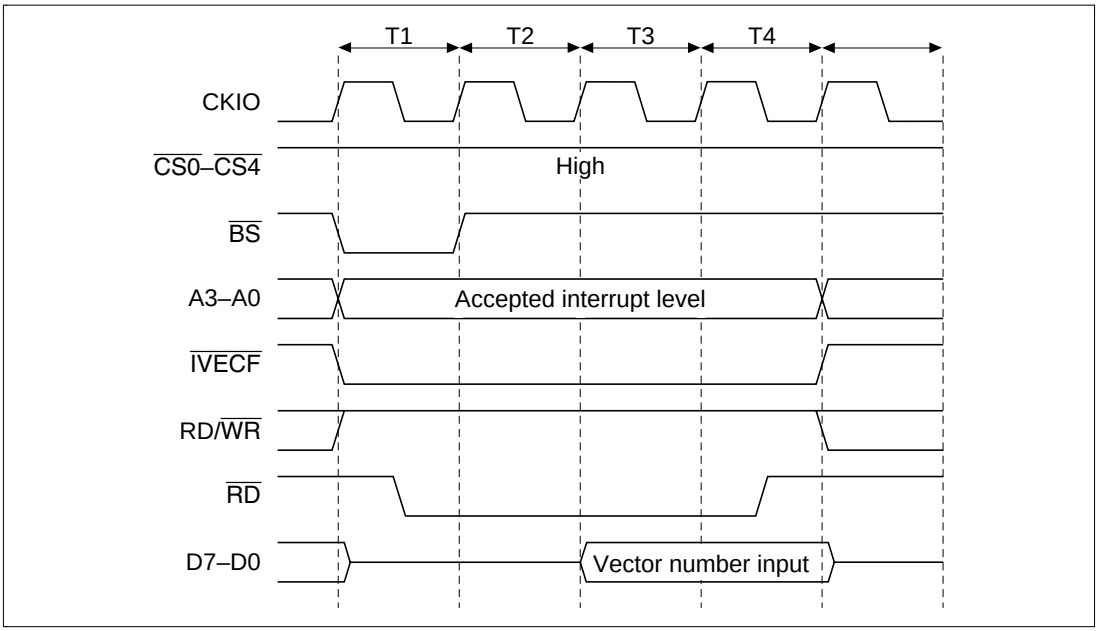



**Figure 5.4 External Vector Fetch (Iϕ : Eϕ = 1 : 1)**

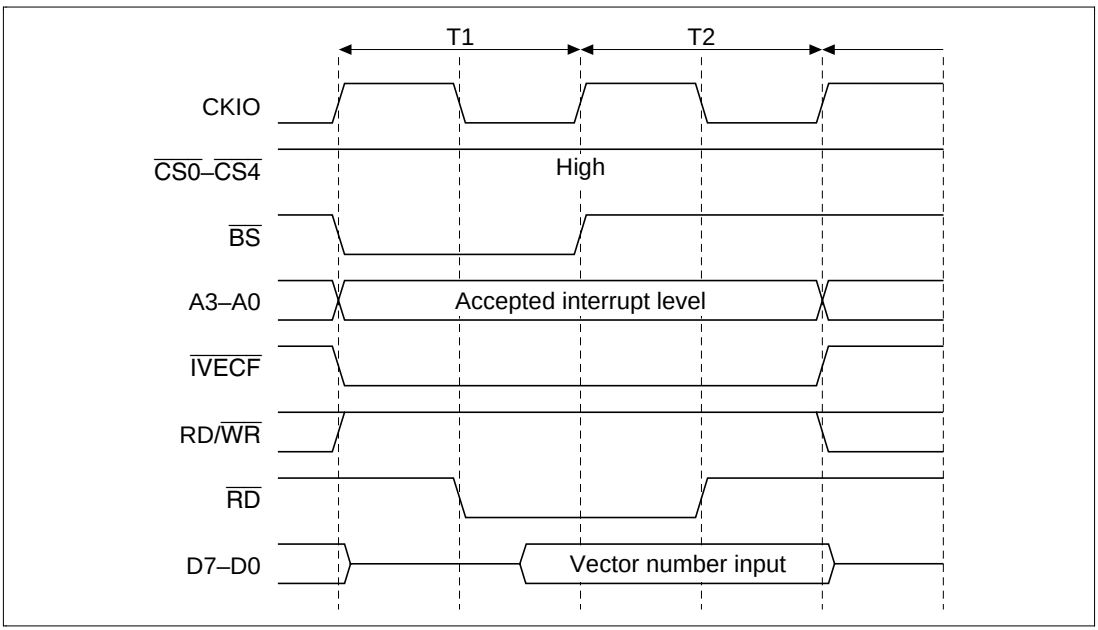



**Figure 5.5 External Vector Fetch (Iϕ : Eϕ ≠ 1 : 1)**

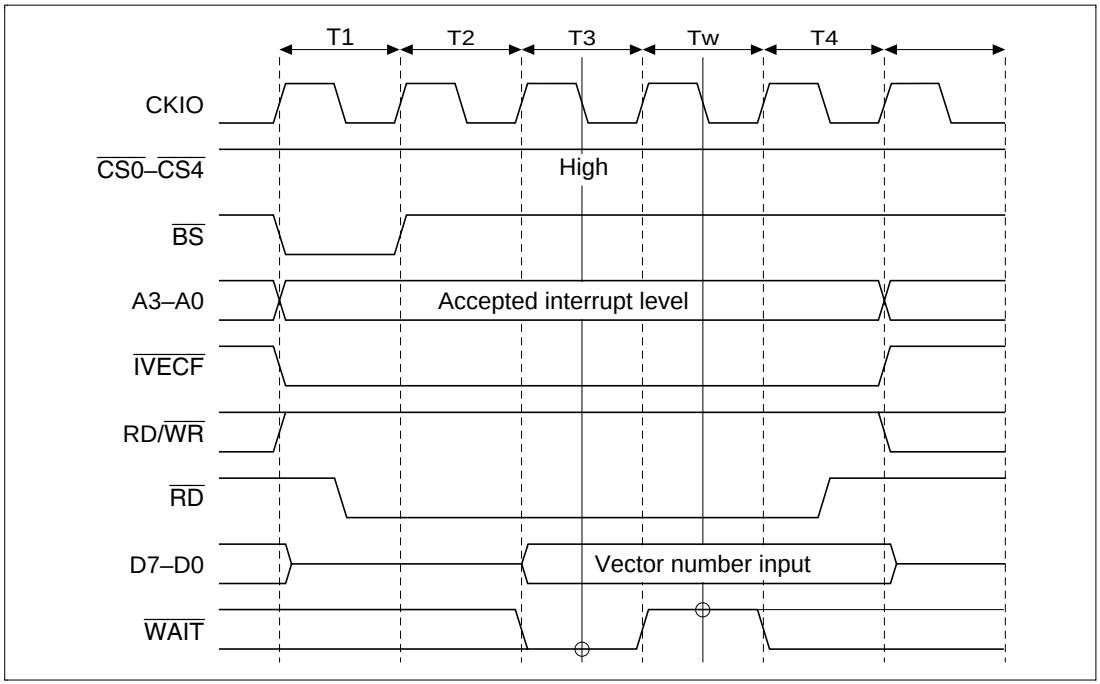



**Figure 5.6 External Vector Fetch (Iϕ : Eϕ = 1 : 1 (WAIT Input))**

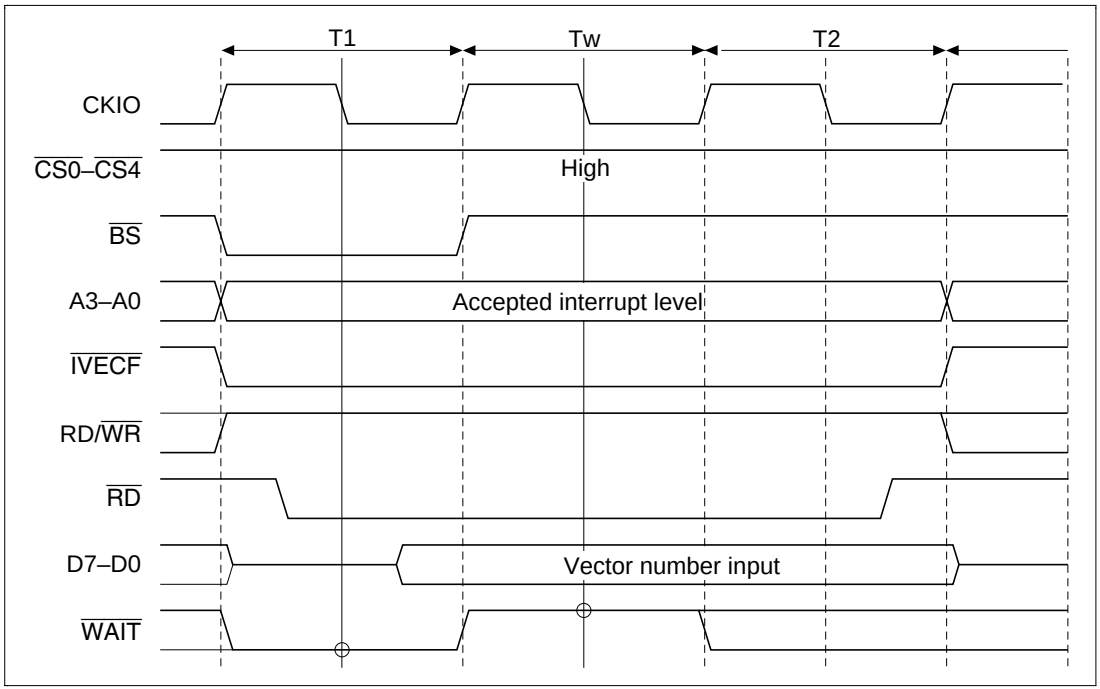



**Figure 5.7 External Vector Fetch (Iϕ : Eϕ ≠ 1 : 1 (WAIT Input))**

### 5.2.6 On-chip Peripheral Module Interrupts

On-chip peripheral module interrupts are interrupts generated by the following 9 on-chip peripheral modules:

- Division unit (DIVU)
- Direct memory access controller (DMAC)
- Serial communication interface (SCI)
- Bus state controller (BSC)
- Watchdog timer (WDT)
- Free-running timer (FRT)
- 16-bit timer pulse unit (TPU)
- Serial communication interface with FIFO (SCIF)
- Serial I/O (SIO)

A different interrupt vector is assigned to each interrupt source, so the exception service routine does not have to decide which interrupt has occurred. Priority levels between 0 and 15 can be assigned to individual on-chip peripheral modules in interrupt priority registers A–E (IPRA–IPRE). On-chip peripheral module interrupt exception handling sets the interrupt mask level bits (I3–I0) in the status register (SR) to the priority level value of the on-chip peripheral module interrupt that was accepted.

### 5.2.7 Interrupt Exception Vectors and Priority Order

Table 5.4 lists interrupt sources and their vector numbers, vector table address offsets and interrupt priorities.

Each interrupt source is allocated a different vector number and vector table address offset. Vector table addresses are calculated from vector numbers and vector table address offsets. In interrupt exception handling, the exception service routine start address is fetched from the vector table entry indicated by the vector table address. See table 4.4, Calculating Exception Vector Table Addresses, in section 4, Exception Handling, for more information on this calculation.

IRL interrupts IRL15–IRL1 have interrupt priority levels of 15–1, respectively. IRQ interrupt and on-chip peripheral module interrupt priorities can be set freely between 0 and 15 for each module by setting interrupt priority registers A–E (IPRA–IPRE). The ranking of interrupt sources for IPRA–IPRE, however, must be the order listed under Priority within IPR Setting Unit in table 5.4 and cannot be changed. A reset assigns priority level 0 to on-chip peripheral module interrupts. If the same priority level is assigned to two or more interrupt sources and interrupts from those sources occur simultaneously, their priority order is the default priority order indicated at the right in table 5.4.

**Table 5.4 (a) Interrupt Exception Vectors and Priority Order (IRL Mode)**

| Interrupt Source | | Vectors: Vector No. | Vectors: Vector Table Address | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| NMI | | 11 | VBR + (vector No. × 4) | 16 | — | — | — | High |
| User break | | 12 | | 15 | — | — | — | ↑ |
| H-UDI | | 13 | | 15 | — | — | — | |
| IRL15*4 | | 71*1 | | 15 | — | — | — | |
| IRL14*4 | | | | 14 | — | — | — | |
| IRL13*4 | | 70*1 | | 13 | — | — | — | |
| IRL12*4 | | | | 12 | — | — | — | |
| IRL11*4 | | 69*1 | | 11 | — | — | — | |
| IRL10*4 | | | | 10 | — | — | — | |
| IRL9*4 | | 68*1 | | 9 | — | — | — | |
| IRL8*4 | | | | 8 | — | — | — | |
| IRL7*4 | | 67*1 | | 7 | — | — | — | |
| IRL6*4 | | | | 6 | — | — | — | |
| IRL5*4 | | 66*1 | | 5 | — | — | — | |
| IRL4*4 | | | | 4 | — | — | — | |
| IRL3*4 | | 65*1 | | 3 | — | — | — | |
| IRL2*4 | | | | 2 | — | — | — | |
| IRL1*4 | | 64*1 | | 1 | — | — | — | |
| DIVU | OVFI | 0–127*2 | | 15–0 (0) | IPRA (15–12) | High ↑ ↓ Low | VCRDIV (6–0) | |
| DMAC0 | Transfer end | 0–127*2 | | 15–0 (0) | IPRA (11–8) | High ↑ ↓ Low | VCRDMA0 (6–0) | |
| DMAC1 | Transfer end | 0–127*2 | | 15–0 (0) | | High ↑ ↓ Low | VCRDMA1 (6–0) | ↓ Low |

**Table 5.4 (a) Interrupt Exception Vectors and Priority Order (IRL Mode) (cont)**

| Interrupt Source | | Vectors | | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| | | Vector No. | Vector Table Address | | | | | |
| WDT | ITI | 0–127*2 | VBR + (vector No. × 4) | 15–0 (0) | IPRA (7–4) | High<br>↑<br>↓<br>Low | VCRWDT (14–8) | High<br>↑<br>↓<br>Low |
| REF*3 | CMI | 0–127*2 | | 15–0 (0) | | High<br>↑<br>↓<br>Low | VCRWDT (6–0) | |
| SCI | ERI | 0–127*2 | | 15–0 (0) | IPRB (15–12) | High<br>↑<br>↓<br>Low | VCRA (14–8) | |
| | RXI | 0–127*2 | | | | | VCRA (6–0) | |
| | TXI | 0–127*2 | | | | | VCRB (14–8) | |
| | TEI | 0–127*2 | | | | | VCRB (6–0) | |
| FRT | ICI | 0–127*2 | | 15–0 (0) | IPRB (11–8) | High<br>↑<br>↓<br>Low | VCRC (14–8) | |
| | OCI | 0–127*2 | | | | | VCRC (6–0) | |
| | OVI | 0–127*2 | | | | | VCRD (14–8) | |
| TPU0 | TGI0A | 0–127*2 | | 15–0 (0) | IPRD (15–12) | High<br>↑<br>↓<br>Low | VCRE (14–8) | |
| | TGI0B | 0–127*2 | | | | | VCRE (6–0) | |
| | TGI0C | 0–127*2 | | | | | VCRF (14–8) | |
| | TGI0D | 0–127*2 | | | | | VCRF (6–0) | |
| | TCI0V | 0–127*2 | | | | | VCRG (14–8) | |
| TPU1 | TGI1A | 0–127*2 | | 15–0 (0) | IPRD (11–8) | High<br>↑<br>↓<br>Low | VCRH (14–8) | |
| | TGI1B | 0–127*2 | | | | | VCRH (6–0) | |
| | TCI1V | 0–127*2 | | | | | VCRI (14–8) | |
| | TCI1U | 0–127*2 | | | | | VCRI (6–0) | |
| TPU2 | TGI2A | 0–127*2 | | 15–0 (0) | IPRD (7–4) | High<br>↑<br>↓<br>Low | VCRJ (14–8) | |
| | TGI2B | 0–127*2 | | | | | VCRJ (6–0) | |
| | TCI2V | 0–127*2 | | | | | VCRK (14–8) | ↓ |
| | TCI2U | 0–127*2 | | | | | VCRK (6–0) | Low |

**Table 5.4 (a) Interrupt Exception Vectors and Priority Order (IRL Mode) (cont)**

| Interrupt Source | | Vectors: Vector No. | Vectors: Vector Table Address | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| SCIF1 | ERI1 | 0–127*[2] | VBR + (vector No. × 4) | 15–0 (0) | IPRD (3–0) | High | VCRL (14–8) | High |
| | RXI1 | 0–127*[2] | | | | ↑ | VCRL (6–0) | ↑ |
| | BRI1 | 0–127*[2] | | | | ↓ | VCRM (14–8) | |
| | TXI1 | 0–127*[2] | | | | Low | VCRM (6–0) | |
| SCIF2 | ERI2 | 0–127*[2] | | 15–0 (0) | IPRE (15–12) | High | VCRN (14–8) | |
| | RXI2 | 0–127*[2] | | | | ↑ | VCRN (6–0) | |
| | BRI2 | 0–127*[2] | | | | ↓ | VCRO (14–8) | |
| | TXI2 | 0–127*[2] | | | | Low | VCRO (6–0) | |
| SIO0 | RERI0 | 0–127*[2] | | 15–0 (0) | IPRE (11–8) | High | VCRP (14–8) | |
| | TERI0 | 0–127*[2] | | | | ↑ | VCRP (6–0) | |
| | RDFI0 | 0–127*[2] | | | | ↓ | VCRQ (14–8) | |
| | TDEI0 | 0–127*[2] | | | | Low | VCRQ (6–0) | |
| SIO1 | RERI1 | 0–127*[2] | | 15–0 (0) | IPRE (7–4) | High | VCRR (14–8) | |
| | TERI1 | 0–127*[2] | | | | ↑ | VCRR (6–0) | |
| | RDFI1 | 0–127*[2] | | | | ↓ | VCRS (14–8) | |
| | TDEI1 | 0–127*[2] | | | | Low | VCRS (6–0) | |
| SIO2 | RERI2 | 0–127*[2] | | 15–0 (0) | IPRE (3–0) | High | VCRT (14–8) | |
| | TERI2 | 0–127*[2] | | | | ↑ | VCRT (6–0) | |
| | RDFI2 | 0–127*[2] | | | | ↓ | VCRU (14–8) | |
| | TDEI2 | 0–127*[2] | | | | Low | VCRU (6–0) | ↓ |
| Reserved | | 128–255 | — | — | — | — | — | Low |

Notes: 1. An external vector number fetch can be performed without using the auto-vector numbers shown in this table. The external vector numbers are 0–127.
2. Vector numbers are set in the on-chip vector number register.
3. REF is the refresh control unit within the bus state controller.
4. Set to IRL1–IRL15 or IRQ0–IRQ3 by the EXIMD bit in ICR.

**Table 5.4 (b) Interrupt Exception Vectors and Priority Order (IRQ Mode)**

| Interrupt Source | | Vectors: Vector No. | Vectors: Vector Table Address | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| NMI | | 11 | VBR + (vector No. × 4) | 16 | — | — | — | High |
| User break | | 12 | | 15 | — | — | — | ↑ |
| H-UDI | | 13 | | 15 | — | — | — | |
| IRQ0*4 | | 64*1 | | 15–0 (0) | IPRC (15–12) | — | — | |
| IRQ1*4 | | 65*1 | | 15–0 (0) | IPRC (11–8) | — | — | |
| IRQ2*4 | | 66*1 | | 15–0 (0) | IPRC (7–4) | — | — | |
| IRQ3*4 | | 67*1 | | 15–0 (0) | IPRC (3–0) | — | — | |
| DIVU | OVFI | 0–127*2 | | 15–0 (0) | IPRA (15–12) | High ↑ ↓ Low | VCRDIV (6–0) | |
| DMAC0 | Transfer end | 0–127*2 | | 15–0 (0) | IPRA (11–8) | High ↑ ↓ Low | VCRDMA0 (6–0) | |
| DMAC1 | Transfer end | 0–127*2 | | 15–0 (0) | | High ↑ ↓ Low | VCRDMA1 (6–0) | |
| WDT | ITI | 0–127*2 | | 15–0 (0) | IPRA (7–4) | High ↑ ↓ Low | VCRWDT (14–8) | |
| REF*3 | CMI | 0–127*2 | | 15–0 (0) | | High ↑ ↓ Low | VCRWDT (6–0) | ↓ Low |

**Table 5.4 (b) Interrupt Exception Vectors and Priority Order (IRQ Mode) (cont)**

| Interrupt Source | | Vectors | | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| | | Vector No. | Vector Table Address | | | | | |
| SCI | ERI | 0–127*[2] | VBR + (vector No. × 4) | 15–0 (0) | IPRB (15–12) | High | VCRA (14–8) | High |
| | RXI | 0–127*[2] | | | | ↑ | VCRA (6–0) | ↑ |
| | TXI | 0–127*[2] | | | | ↓ | VCRB (14–8) | |
| | TEI | 0–127*[2] | | | | Low | VCRB (6–0) | |
| FRT | ICI | 0–127*[2] | | 15–0 (0) | IPRB (11–8) | High | VCRC (14–8) | |
| | OCIA/B | 0–127*[2] | | | | ↑ | VCRC (6–0) | |
| | OVI | 0–127*[2] | | | | ↓ | VCRD (14–8) | |
| | | | | | | Low | | |
| TPU0 | TGI0A | 0–127*[2] | | 15–0 (0) | IPRD (15–12) | High | VCRE (14–8) | |
| | TGI0B | 0–127*[2] | | | | ↑ | VCRE (6–0) | |
| | TGI0C | 0–127*[2] | | | | | VCRF (14–8) | |
| | TGI0D | 0–127*[2] | | | | ↓ | VCRF (6–0) | |
| | TCI0V | 0–127*[2] | | | | Low | VCRG (14–8) | |
| TPU1 | TGI1A | 0–127*[2] | | 15–0 (0) | IPRD (11–8) | High | VCRH (14–8) | |
| | TGI1B | 0–127*[2] | | | | ↑ | VCRH (6–0) | |
| | TCI1V | 0–127*[2] | | | | ↓ | VCRI (14–8) | |
| | TCI1U | 0–127*[2] | | | | Low | VCRI (6–0) | |
| TPU2 | TGI2A | 0–127*[2] | | 15–0 (0) | IPRD (7–4) | High | VCRJ (14–8) | |
| | TGI2B | 0–127*[2] | | | | ↑ | VCRJ (6–0) | |
| | TCI2V | 0–127*[2] | | | | ↓ | VCRK (14–8) | |
| | TCI2U | 0–127*[2] | | | | Low | VCRK (6–0) | |
| SCIF1 | ERI1 | 0–127*[2] | | 15–0 (0) | IPRD (3–0) | High | VCRL (14–8) | |
| | RXI1 | 0–127*[2] | | | | ↑ | VCRL (6–0) | |
| | BRI1 | 0–127*[2] | | | | ↓ | VCRM (14–8) | |
| | TXI1 | 0–127*[2] | | | | Low | VCRM (6–0) | |
| SCIF2 | ERI2 | 0–127*[2] | | 15–0 (0) | IPRE (15–12) | High | VCRN (14–8) | |
| | RXI2 | 0–127*[2] | | | | ↑ | VCRN (6–0) | |
| | BRI2 | 0–127*[2] | | | | ↓ | VCRO (14–8) | ↓ |
| | TXI2 | 0–127*[2] | | | | Low | VCRO (6–0) | Low |

**Table 5.4 (b) Interrupt Exception Vectors and Priority Order (IRQ Mode) (cont)**

| Interrupt Source | | Vectors: Vector No. | Vectors: Vector Table Address | Interrupt Priority Order (Initial Value) | IPR (Bit Numbers) | Priority within IPR Setting Unit | VCR (Bit Numbers) | Default Priority |
|---|---|---|---|---|---|---|---|---|
| SIO0 | RERI0 | 0–127*[2] | VBR + (vector No. × 4) | 15–0 (0) | IPRE (11–8) | High | VCRP (14–8) | High |
| | TERI0 | 0–127*[2] | | | | ↑ | VCRP (6–0) | ↑ |
| | RDFI0 | 0–127*[2] | | | | ↓ | VCRQ (14–8) | |
| | TDEI0 | 0–127*[2] | | | | Low | VCRQ (6–0) | |
| SIO1 | RERI1 | 0–127*[2] | | 15–0 (0) | IPRE (7–4) | High | VCRR (14–8) | |
| | TERI1 | 0–127*[2] | | | | ↑ | VCRR (6–0) | |
| | RDFI1 | 0–127*[2] | | | | ↓ | VCRS (14–8) | |
| | TDEI1 | 0–127*[2] | | | | Low | VCRS (6–0) | |
| SIO2 | RERI2 | 0–127*[2] | | 15–0 (0) | IPRE (3–0) | High | VCRT (14–8) | |
| | TERI2 | 0–127*[2] | | | | ↑ | VCRT (6–0) | |
| | RDFI2 | 0–127*[2] | | | | ↓ | VCRU (14–8) | |
| | TDEI2 | 0–127*[2] | | | | Low | VCRU (6–0) | ↓ |
| Reserved | | 128–255 | — | — | — | — | — | Low |

Notes: 1. An external vector number fetch can be performed without using the auto-vector numbers shown in this table. The external vector numbers are 0–127.
2. Vector numbers are set in the on-chip vector number register.
3. REF is the refresh control unit within the bus state controller.
4. Set to IRL1–IRL15 or IRQ0–IRQ3 by the EXIMD bit in ICR.

## 5.3 Register Descriptions

### 5.3.1 Interrupt Priority Level Setting Register A (IPRA)

Interrupt priority level setting register A (IPRA) is a 16-bit read/write register that assigns priority levels from 0 to 15 to on-chip peripheral module interrupts. IPRA is initialized to H'0000 by a reset. It is not initialized in standby mode. Unless otherwise specified, ‘reset’ refers to both power-on and manual resets throughout this manual.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | DIVU IP3 | DIVU IP2 | DIVU IP1 | DIVU IP0 | DMAC IP3 | DMAC IP2 | DMAC IP1 | DMAC IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | WDT IP3 | WDT IP2 | WDT IP1 | WDT IP0 | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R |

Bits 15 to 12—Division Unit (DIVU) Interrupt Priority Level 3 to 0 (DIVUIP3–DIVUIP0): These bits set the division unit (DIVU) interrupt priority level. There are four bits, so levels 0–15 can be set.

Bits 11 to 8—DMA Controller Interrupt Priority Level 3 to 0 (DMACIP3–DMACIP0): These bits set the DMA controller (DMAC) interrupt priority level. There are four bits, so levels 0–15 can be set. The same level is set for both two DMAC channels. When interrupts occur simultaneously, channel 0 has priority.

Bits 7 to 4—Watchdog Timer (WDT) Interrupt Priority Level 3 to 0 (WDTIP3–WDTIP0): These bits set the watchdog timer (WDT) interrupt priority level and bus state controller (BSC) interrupt priority level. There are four bits, so levels 0–15 can be set. When WDT and BSC interrupts occur simultaneously, the WDT interrupt has priority.

Bits 3 to 0—Reserved: These bits always read 0. The write value should always be 0.

### 5.3.2 Interrupt Priority Level Setting Register B (IPRB)

Interrupt priority level setting register B (IPRB) is a 16-bit read/write register that assigns priority levels from 0 to 15 to on-chip peripheral module interrupts. IPRB is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | SCIIP3 | SCIIP2 | SCIIP1 | SCIIP0 | FRTIP3 | FRTIP2 | FRTIP1 | FRTIP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

Bits 15 to 12—Serial Communication Interface (SCI) Interrupt Priority Level 3 to 0 (SCIIP3–SCIIP0): These bits set the serial communication interface (SCI) interrupt priority level. There are four bits, so levels 0–15 can be set.

Bits 11 to 8—Free-Running Timer (FRT) Interrupt Priority Level 3 to 0 (FRTIP3–FRTIP0): These bits set the free-running timer (FRT) interrupt priority level. There are four bits, so levels 0–15 can be set.

Bits 7 to 0—Reserved: These bits always read 0. The write value should always be 0.

### 5.3.3 Interrupt Priority Level Setting Register C (IPRC)

Interrupt priority level setting register C (IPRC) is a 16-bit read/write register that sets the priority levels (0–15) of IRQ0–IRQ3 interrupts. IPRC is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IRQ0IP3 | IRQ0IP2 | IRQ0IP1 | IRQ0IP0 | IRQ1IP3 | IRQ1IP2 | IRQ1IP1 | IRQ1IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IRQ2IP3 | IRQ2IP2 | IRQ2IP1 | IRQ2IP0 | IRQ3IP3 | IRQ3IP2 | IRQ3IP1 | IRQ3IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 to 0–IRQ0 to IRQ3 Priority Level 3 to 0 (IRQnIP3–IRQnIP0, n = 0–3): These bits set the IRQ0–IRQ3 priority levels. There are four bits for each interrupt, so the value can be set between 0 and 15.

### 5.3.4 Interrupt Priority Level Setting Register D (IPRD)

Interrupt priority level setting register D (IPRD) is a 16-bit read/write register that sets the priority levels (0–15) of on-chip peripheral module interrupts. IPRD is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TPU0IP3 | TPU0IP2 | TPU0IP1 | TPU0IP0 | TPU1IP3 | TPU1IP2 | TPU1IP1 | TPU1IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TPU2IP3 | TPU2IP2 | TPU2IP1 | TPU2IP0 | SCF1IP3 | SCF1IP2 | SCF1IP1 | SCF1IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 to 4—Timer Pulse Unit 0 to 2 (TPU0–TPU2) Interrupt Priority Level 3 to 0 (TPUnIP3–TPUnIP0, n = 0–2): These bits set the timer pulse unit 0 to 2 (TPU0–TPU2) interrupt priority levels. There are four bits for each interrupt, so the value can be set between 0 and 15.

Bits 3 to 0—Serial Communication Interface with FIFO (SCIF1) Interrupt Priority Level 3 to 0 (SCF1IP3–SCF1IP0): These bits set the serial communication interface with FIFO (SCIF1) interrupt priority level. There are four bits, so the value can be set between 0 and 15.

### 5.3.5 Interrupt Priority Level Setting Register E (IPRE)

Interrupt priority level setting register E (IPRE) is a 16-bit read/write register that sets the priority levels (0–15) of on-chip peripheral module interrupts. IPRE is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | SCF2IP3 | SCF2IP2 | SCF2IP1 | SCF2IP0 | SIO0IP3 | SIO0IP2 | SIO0IP1 | SIO0IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | SIO1IP3 | SIO1IP2 | SIO1IP1 | SIO1IP0 | SIO2IP3 | SIO2IP2 | SIO2IP1 | SIO2IP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 to 12—Serial Communication Interface with FIFO (SCIF2) Interrupt Priority Level 3 to 0 (SCF2IP3–SCF2IP0): These bits set the serial communication interface with FIFO (SCIF2) interrupt priority level. There are four bits, so the value can be set between 0 and 15.

Bits 11 to 0—Serial I/O 0 to 2 (SIO0–SIO2) Interrupt Priority Level 3 to 0 (SIOnIP3–SIOnIP0, n = 0–2): These bits set the serial I/O 0 to 2 (SIO0–SIO2) interrupt priority levels. There are four bits for each interrupt, so the value can be set between 0 and 15.

Table 5.5 shows the relationship between on-chip peripheral module interrupts and interrupt priority level setting registers.

**Table 5.5 Interrupt Request Sources and IPRA–IPRE**

| Register | Bits 15 to 12 | Bits 11 to 8 | Bits 7 to 4 | Bits 3 to 0 |
|---|---|---|---|---|
| Interrupt priority level setting register A | DIVU | DMAC0, DMAC1 | WDT, REF | Reserved |
| Interrupt priority level setting register B | SCI | FRT | Reserved | Reserved |
| Interrupt priority level setting register C | IRQ0 | IRQ1 | IRQ2 | IRQ3 |
| Interrupt priority level setting register D | TPU0 | TPU1 | TPU2 | SCIF1 |
| Interrupt priority level setting register E | SCIF2 | SIO0 | SIO1 | SIO2 |

As table 5.5 shows, between two and four on-chip peripheral modules are assigned to each interrupt priority level setting register. Set the priority levels by setting the corresponding 4-bit groups with values in the range of H'0 (0000) to H'F (1111). H'0 is interrupt priority level 0 (the lowest); H'F is level 15 (the highest). When two on-chip peripheral modules are assigned to the same bits (DMAC0 and DMAC1, or WDT and DRAM refresh control unit), those two modules have the same priority. A reset initializes IPRA–IPRE to H'0000. They are not initialized in standby mode.

### 5.3.6 Vector Number Setting Register WDT (VCRWDT)

Vector number setting register WDT (VCRWDT) is a 16-bit read/write register that sets the WDT interval interrupt and BSC compare match interrupt vector numbers (0–127). VCRWDT is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | WITV6 | WITV5 | WITV4 | WITV3 | WITV2 | WITV1 | WITV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | BCMV6 | BCMV5 | BCMV4 | BCMV3 | BCMV2 | BCMV1 | BCMV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Watchdog Timer (WDT) Interval Interrupt Vector Number 6 to 0 (WITV6–WITV0): These bits set the vector number for the interval interrupt (ITI) of the watchdog timer (WDT). There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Bus State Controller (BSC) Compare Match Interrupt Vector Number 6 to 0 (BCMV6–BCMV0): These bits set the vector number for the compare match interrupt (CMI) of the bus state controller (BSC). There are seven bits, so the value can be set between 0 and 127.

### 5.3.7 Vector Number Setting Register A (VCRA)

Vector number setting register A (VCRA) is a 16-bit read/write register that sets the SCI receive-error interrupt and receive-data-full interrupt vector numbers (0–127). VCRA is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SERV6 | SERV5 | SERV4 | SERV3 | SERV2 | SERV1 | SERV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SRXV6 | SRXV5 | SRXV4 | SRXV3 | SRXV2 | SRXV1 | SRXV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface (SCI) Receive-Error Interrupt Vector Number 6 to 0 (SERV6–SERV0): These bits set the vector number for the serial communication interface (SCI) receive-error interrupt (ERI). There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface (SCI) Receive-Data-Full Interrupt Vector Number 6 to 0 (SRXV6–SRXV0): These bits set the vector number for the serial communication interface (SCI) receive-data-full interrupt (RXI). There are seven bits, so the value can be set between 0 and 127.

### 5.3.8 Vector Number Setting Register B (VCRB)

Vector number setting register B (VCRB) is a 16-bit read/write register that sets the SCI transmit-data-empty interrupt and transmit-end interrupt vector numbers (0–127). VCRB is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | STXV6 | STXV5 | STXV4 | STXV3 | STXV2 | STXV1 | STXV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | STEV6 | STEV5 | STEV4 | STEV3 | STEV2 | STEV1 | STEV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface (SCI) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (STXV6–STXV0): These bits set the vector number for the serial communication interface (SCI) transmit-data-empty interrupt (TXI). There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface (SCI) Transmit-End Interrupt Vector Number 6 to 0 (STEV6–STEV0): These bits set the vector number for the serial communication interface (SCI) transmit-end interrupt (TEI). There are seven bits, so the value can be set between 0 and 127.

### 5.3.9 Vector Number Setting Register C (VCRC)

Vector number setting register C (VCRC) is a 16-bit read/write register that sets the free-funning timer (FRT) input-capture interrupt and output-compare interrupt vector numbers (0–127). VCRC is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | FICV6 | FICV5 | FICV4 | FICV3 | FICV2 | FICV1 | FICV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | FOCV6 | FOCV5 | FOCV4 | FOCV3 | FOCV2 | FOCV1 | FOCV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Free-Running Timer (FRT) Input-Capture Interrupt Vector Number 6 to 0 (FICV6–FICV0): These bits set the vector number for the free-running timer (FRT) input-capture interrupt (ICI). There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Free-Running Timer (FRT) Output-Compare Interrupt Vector Number 6 to 0 (FOCV6–FOCV0): These bits set the vector number for the free-running timer (FRT) output-compare interrupt (OCI). There are seven bits, so the value can be set between 0 and 127.

### 5.3.10 Vector Number Setting Register D (VCRD)

Vector number setting register D (VCRD) is a 16-bit read/write register that sets the free-running timer (FRT) overflow interrupt vector number (0–127). VCRD is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | FOVV6 | FOVV5 | FOVV4 | FOVV3 | FOVV2 | FOVV1 | FOVV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

Bits 15 and 7 to 0—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Free-Running Timer (FRT) Overflow Interrupt Vector Number 6 to 0 (FOVV6–FOVV0): These bits set the vector number for the free-running timer (FRT) overflow interrupt (OVI). There are seven bits, so the value can be set between 0 and 127.

### 5.3.11 Vector Number Setting Register E (VCRE)

Vector number setting register E (VCRE) is a 16-bit read/write register that sets the timer pulse unit 0 (TPU0) TGR0A and TGR0B input capture/compare match interrupt vector numbers (0–127).

VCRE is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG0AV6 | TG0AV5 | TG0AV4 | TG0AV3 | TG0AV2 | TG0AV1 | TG0AV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG0BV6 | TG0BV5 | TG0BV4 | TG0BV3 | TG0BV2 | TG0BV1 | TG0BV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved. These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 0 (TPU0) TGR0A Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG0AV6–TG0AV0): These bits set the vector number for the timer pulse unit 0 (TPU0) TGR0A input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 0 (TPU0) TGR0B Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG0BV6–TG0BV0): These bits set the vector number for the timer pulse unit 0 (TPU0) TGR0B input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.12 Vector Number Setting Register F (VCRF)

Vector number setting register F (VCRF) is a 16-bit read/write register that sets the timer pulse unit 0 (TPU0) TGR0C and TGR0D input capture/compare match interrupt vector numbers (0–127).

VCRF is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG0CV6 | TG0CV5 | TG0CV4 | TG0CV3 | TG0CV2 | TG0CV1 | TG0CV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG0DV6 | TG0DV5 | TG0DV4 | TG0DV3 | TG0DV2 | TG0DV1 | TG0DV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 0 (TPU0) TGR0C Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG0CV6–TG0CV0): These bits set the vector number for the timer pulse unit 0 (TPU0) TGR0C input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 0 (TPU0) TGR0D Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG0DV6–TG0DV0): These bits set the vector number for the timer pulse unit 0 (TPU0) TGR0D input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.13 Vector Number Setting Register G (VCRG)

Vector number setting register G (VCRG) is a 16-bit read/write register that sets the timer pulse unit 0 (TPU0) TCNT0 overflow interrupt vector number (0–127).

VCRG is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TC0VV6 | TC0VV5 | TC0VV4 | TC0VV3 | TC0VV2 | TC0VV1 | TC0VV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

Bits 15 and 7 to 0—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 0 (TPU0) TCNT0 Overflow Interrupt Vector Number 6 to 0 (TC0VV6–TV0VV0): These bits set the vector number for the timer pulse unit 0 (TPU0) TCNT0 overflow interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.14 Vector Number Setting Register H (VCRH)

Vector number setting register H (VCRH) is a 16-bit read/write register that sets the timer pulse unit 1 (TPU1) TGR1A and TGR1B input capture/compare match interrupt vector numbers (0–127).

VCRH is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG1AV6 | TG1AV5 | TG1AV4 | TG1AV3 | TG1AV2 | TG1AV1 | TG1AV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG1BV6 | TG1BV5 | TG1BV4 | TG1BV3 | TG1BV2 | TG1BV1 | TG1BV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 1 (TPU1) TGR1A Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG1AV6–TG1AV0): These bits set the vector number for the timer pulse unit 1 (TPU1) TGR1A input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 1 (TPU1) TGR1B Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG1BV6–TG1BV0): These bits set the vector number for the timer pulse unit 1 (TPU1) TGR1B input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.15 Vector Number Setting Register I (VCRI)

Vector number setting register I (VCRI) is a 16-bit read/write register that sets the timer pulse unit 1 (TPU1) TCNT1 overflow/underflow interrupt vector numbers (0–127).

VCRI is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TC1VV6 | TC1VV5 | TC1VV4 | TC1VV3 | TC1VV2 | TC1VV1 | TC1VV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TC1UV6 | TC1UV5 | TC1UV4 | TC1UV3 | TC1UV2 | TC1UV1 | TC1UV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 1 (TPU1) TCNT1 Overflow Interrupt Vector Number 6 to 0 (TC1VV6–TC1VV0): These bits set the vector number for the timer pulse unit 1 (TPU1) TCNT1 overflow interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 1 (TPU1) TCNT1 Underflow Interrupt Vector Number 6 to 0 (TC1UV6–TC1UV0): These bits set the vector number for the timer pulse unit 1 (TPU1) TCNT1 underflow interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.16 Vector Number Setting Register J (VCRJ)

Vector number setting register J (VCRJ) is a 16-bit read/write register that sets the timer pulse unit 2 (TPU2) TGR2A and TGR2B input capture/compare match interrupt vector numbers (0–127).

VCRJ is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG2AV6 | TG2AV5 | TG2AV4 | TG2AV3 | TG2AV2 | TG2AV1 | TG2AV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TG2BV6 | TG2BV5 | TG2BV4 | TG2BV3 | TG2BV2 | TG2BV1 | TG2BV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 2 (TPU2) TGR2A Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG2AV6–TG2AV0): These bits set the vector number for the timer pulse unit 2 (TPU2) TGR2A input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 2 (TPU2) TGR2B Input Capture/Compare Match Interrupt Vector Number 6 to 0 (TG2BV6–TG2BV0): These bits set the vector number for the timer pulse unit 2 (TPU2) TGR2B input capture/compare match interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.17 Vector Number Setting Register K (VCRK)

Vector number setting register K (VCRK) is a 16-bit read/write register that sets the timer pulse unit 2 (TPU2) TCNT2 overflow/underflow interrupt vector numbers (0–127).

VCRK is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TC2VV6 | TC2VV5 | TC2VV4 | TC2VV3 | TC2VV2 | TC2VV1 | TC2VV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TC2UV6 | TC2UV5 | TC2UV4 | TC2UV3 | TC2UV2 | TC2UV1 | TC2UV0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Timer pulse unit 2 (TPU2) TCNT2 Overflow Interrupt Vector Number 6 to 0 (TC2VV6–TC2VV0): These bits set the vector number for the timer pulse unit 2 (TPU2) TCNT2 overflow interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Timer pulse unit 2 (TPU2) TCNT2 Underflow Interrupt Vector Number 6 to 0 (TC2UV6–TC2UV0): These bits set the vector number for the timer pulse unit 2 (TPU2) TCNT2 underflow interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.18 Vector Number Setting Register L (VCRL)

Vector number setting register L (VCRL) is a 16-bit read/write register that sets the serial communication interface with FIFO 1 (SCIF1) receive-error interrupt and receive-data-full/data-ready interrupt vector numbers (0–127).

VCRL is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SER1V6 | SER1V5 | SER1V4 | SER1V3 | SER1V2 | SER1V1 | SER1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SRX1V6 | SRX1V5 | SRX1V4 | SRX1V3 | SRX1V2 | SRX1V1 | SRX1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface with FIFO 1 (SCIF1) Receive-Error Interrupt Vector Number 6 to 0 (SER1V6–SER1V0): These bits set the vector number for the serial communication interface with FIFO 1 (SCIF1) receive-error interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface with FIFO 1 (SCIF1) Receive-Data-Full/Data-Ready Interrupt Vector Number 6 to 0 (SRX1V6–SRX1V0): These bits set the vector number for the serial communication interface with FIFO 1 (SCIF1) receive-data-full/data-ready interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.19 Vector Number Setting Register M (VCRM)

Vector number setting register M (VCRM) is a 16-bit read/write register that sets the serial communication interface with FIFO 1 (SCIF1) break interrupt and transmit-data-empty interrupt vector numbers (0–127).

VCRM is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SBR1V6 | SBR1V5 | SBR1V4 | SBR1V3 | SBR1V2 | SBR1V1 | SBR1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | STX1V6 | STX1V5 | STX1V4 | STX1V3 | STX1V2 | STX1V1 | STX1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface with FIFO 1 (SCIF1) Break Interrupt Vector Number 6 to 0 (SBR1V6–SBR1V0): These bits set the vector number for the serial communication interface with FIFO 1 (SCIF1) break interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface with FIFO 1 (SCIF1) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (STE1V6–STE1V0): These bits set the vector number for the serial communication interface with FIFO 1 (SCIF1) transmit-data-empty interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.20 Vector Number Setting Register N (VCRN)

Vector number setting register N (VCRN) is a 16-bit read/write register that sets the serial communication interface with FIFO 2 (SCIF2) receive-error interrupt and receive-data-full/data-ready interrupt vector numbers (0–127).

VCRN is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SER2V6 | SER2V5 | SER2V4 | SER2V3 | SER2V2 | SER2V1 | SER2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SRX2V6 | SRX2V5 | SRX2V4 | SRX2V3 | SRX2V2 | SRX2V1 | SRX2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface with FIFO 2 (SCIF2) Receive-Error Interrupt Vector Number 6 to 0 (SER2V6–SER2V0): These bits set the vector number for the serial communication interface with FIFO 2 (SCIF2) receive-error interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface with FIFO 2 (SCIF2) Receive-Data-Full/Data-Ready Interrupt Vector Number 6 to 0 (SRX2V6–SRX2V0): These bits set the vector number for the serial communication interface with FIFO 2 (SCIF2) receive-data-full/data-ready interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.21 Vector Number Setting Register O (VCRO)

Vector number setting register O (VCRO) is a 16-bit read/write register that sets the serial communication interface with FIFO 2 (SCIF2) break interrupt and transmit-data-empty interrupt vector numbers (0–127).

VCRO is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | SBR2V6 | SBR2V5 | SBR2V4 | SBR2V3 | SBR2V2 | SBR2V1 | SBR2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | STX2V6 | STX2V5 | STX2V4 | STX2V3 | STX2V2 | STX2V1 | STX2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial Communication Interface with FIFO 2 (SCIF2) Break Interrupt Vector Number 6 to 0 (SBR2V6–SBR2V0): These bits set the vector number for the serial communication interface with FIFO 2 (SCIF2) break interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial Communication Interface with FIFO 2 (SCIF2) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (STE2V6–STE2V0): These bits set the vector number for the serial communication interface with FIFO 2 (SCIF2) transmit-data-empty interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.22 Vector Number Setting Register P (VCRP)

Vector number setting register P (VCRP) is a 16-bit read/write register that sets the serial I/O 0 (SIO0) receive overrun error interrupt and transmit overrun error interrupt vector numbers (0–127).

VCRP is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RER0V6 | RER0V5 | RER0V4 | RER0V3 | RER0V2 | RER0V1 | RER0V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TER0V6 | TER0V5 | TER0V4 | TER0V3 | TER0V2 | TER0V1 | TER0V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 0 (SIO0) Receive Overrun Error Interrupt Vector Number 6 to 0 (RER0V6–RER0V0): These bits set the vector number for the serial I/O 0 (SIO0) receive overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 0 (SIO0) Transmit Overrun Error Interrupt Vector Number 6 to 0 (TER0V6–TER0V0): These bits set the vector number for the serial I/O 0 (SIO0) transmit overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.23 Vector Number Setting Register Q (VCRQ)

Vector number setting register Q (VCRQ) is a 16-bit read/write register that sets the serial I/O 0 (SIO0) receive-data-full interrupt and transmit-data-empty interrupt vector numbers (0–127).

VCRQ is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RDF0V6 | RDF0V5 | RDF0V4 | RDF0V3 | RDF0V2 | RDF0V1 | RDF0V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TDE0V6 | TDE0V5 | TDE0V4 | TDE0V3 | TDE0V2 | TDE0V1 | TDE0V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 0 (SIO0) Receive-Data-Full Interrupt Vector Number 6 to 0 (RDF0V6–RDF0V0): These bits set the vector number for the serial I/O 0 (SIO0) receive-data-full interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 0 (SIO0) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (TDE0V6–TDE0V0): These bits set the vector number for the serial I/O 0 (SIO0) transmit-data-empty interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.24 Vector Number Setting Register R (VCRR)

Vector number setting register R (VCRR) is a 16-bit read/write register that sets the serial I/O 1 (SIO1) receive overrun error interrupt and transmit overrun error interrupt vector numbers (0–127).

VCRR is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RER1V6 | RER1V5 | RER1V4 | RER1V3 | RER1V2 | RER1V1 | RER1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TER1V6 | TER1V5 | TER1V4 | TER1V3 | TER1V2 | TER1V1 | TER1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 1 (SIO1) Receive Overrun Error Interrupt Vector Number 6 to 0 (RER1V6–RER1V0): These bits set the vector number for the serial I/O 1 (SIO1) receive overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 1 (SIO1) Transmit Overrun Error Interrupt Vector Number 6 to 0 (TER1V6–TER1V0): These bits set the vector number for the serial I/O 1 (SIO1) transmit overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.25 Vector Number Setting Register S (VCRS)

Vector number setting register S (VCRS) is a 16-bit read/write register that sets the serial I/O 1 (SIO1) receive-data-full interrupt and transmit-data-empty interrupt vector numbers (0–127).

VCRS is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RDF1V6 | RDF1V5 | RDF1V4 | RDF1V3 | RDF1V2 | RDF1V1 | RDF1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TDE1V6 | TDE1V5 | TDE1V4 | TDE1V3 | TDE1V2 | TDE1V1 | TDE1V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 1 (SIO1) Receive-Data-Full Interrupt Vector Number 6 to 0 (RDF1V6–RDF1V0): These bits set the vector number for the serial I/O 1 (SIO1) receive-data-full interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 1 (SIO1) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (TDE1V6–TDE1V0): These bits set the vector number for the serial I/O 1 (SIO1) transmit-data-empty interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.26 Vector Number Setting Register T (VCRT)

Vector number setting register T (VCRT) is a 16-bit read/write register that sets the serial I/O 2 (SIO2) receive overrun error interrupt and transmit overrun error interrupt vector numbers (0–127).

VCRT is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RER2V6 | RER2V5 | RER2V4 | RER2V3 | RER2V2 | RER2V1 | RER2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TER2V6 | TER2V5 | TER2V4 | TER2V3 | TER2V2 | TER2V1 | TER2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved: These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 2 (SIO2) Receive Overrun Error Interrupt Vector Number 6 to 0 (RER2V6–RER2V0): These bits set the vector number for the serial I/O 2 (SIO2) receive overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 2 (SIO2) Transmit Overrun Error Interrupt Vector Number 6 to 0 (TER2V6–TER2V0): These bits set the vector number for the serial I/O 2 (SIO2) transmit overrun error interrupt. There are seven bits, so the value can be set between 0 and 127.

### 5.3.27 Vector Number Setting Register U (VCRU)

Vector number setting register U (VCRU) is a 16-bit read/write register that sets the serial I/O 2 (SIO2) receive-data-full interrupt and transmit-data-empty interrupt vector numbers (0–127).

VCRU is initialized to H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | RDF2V6 | RDF2V5 | RDF2V4 | RDF2V3 | RDF2V2 | RDF2V1 | RDF2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TDE2V6 | TDE2V5 | TDE2V4 | TDE2V3 | TDE2V2 | TDE2V1 | TDE2V0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 7—Reserved. These bits always read 0. The write value should always be 0.

Bits 14 to 8—Serial I/O 2 (SIO2) Receive-Data-Full Interrupt Vector Number 6 to 0 (RDF2V6–RDF2V0): These bits set the vector number for the serial I/O 2 (SIO2) receive-data-full interrupt. There are seven bits, so the value can be set between 0 and 127.

Bits 6 to 0—Serial I/O 2 (SIO2) Transmit-Data-Empty Interrupt Vector Number 6 to 0 (TDE2V6–TDE2V0): These bits set the vector number for the serial I/O 2 (SIO2) transmit-data-empty interrupt. There are seven bits, so the value can be set between 0 and 127.

Tables 5.6 and 5.7 show the relationship between on-chip peripheral module interrupts and interrupt vector number setting registers.

**Table 5.6 Interrupt Request Sources and Vector Number Setting Registers (1)**

| | Bits | |
|---|---|---|
| **Register** | **14–8** | **6–0** |
| Vector number setting register WDT | Interval interrupt (WDT) | Compare-match interrupt (BSC) |
| Vector number setting register A | Receive-error interrupt (SCI) | Receive-data-full interrupt (SCI) |
| Vector number setting register B | Transmit-data-empty interrupt (SCI) | Transmit-end interrupt (SCI) |
| Vector number setting register C | Input-capture interrupt (FRT) | Output-compare interrupt (FRT) |
| Vector number setting register D | Overflow interrupt (FRT) | Reserved |
| Vector number setting register E | Input capture/compare match interrupt (TPU0/TGR0A) | Input capture/compare match interrupt (TPU0/TGR0B) |
| Vector number setting register F | Input capture/compare match interrupt (TPU0/TGR0C) | Input capture/compare match interrupt (TPU0/TGR0D) |
| Vector number setting register G | Overflow interrupt (TPU0/TCNT0) | Reserved |
| Vector number setting register H | Input capture/compare match interrupt (TPU1/TGR1A) | Input capture/compare match interrupt (TPU1/TGR1B) |
| Vector number setting register I | Overflow interrupt (TPU1/TCNT1) | Underflow interrupt (TPU1/TCNT1) |
| Vector number setting register J | Input capture/compare match interrupt (TPU2/TGR2A) | Input capture/compare match interrupt (TPU2/TGR2B) |
| Vector number setting register K | Overflow interrupt (TPU2/TCNT2) | Underflow interrupt (TPU2/TCNT2) |
| Vector number setting register L | Receive-error interrupt (SCIF1) | Receive-data-full/data-ready interrupt (SCIF1) |
| Vector number setting register M | Break interrupt (SCIF1) | Transmit-data-empty interrupt (SCIF1) |
| Vector number setting register N | Receive-error interrupt (SCIF2) | Receive-data-full/data-ready interrupt (SCIF2) |
| Vector number setting register O | Break interrupt (SCIF2) | Transmit-data-empty interrupt (SCIF2) |
| Vector number setting register P | Receive overrun error interrupt (SIO0) | Transmit overrun error interrupt (SIO0) |
| Vector number setting register Q | Receive-data-full interrupt (SIO0) | Transmit-data-empty interrupt (SIO0) |

**Table 5.6 Interrupt Request Sources and Vector Number Setting Registers (1) (cont)**

| | Bits | |
|---|---|---|
| **Register** | **14–8** | **6–0** |
| Vector number setting register R | Receive overrun error interrupt (SIO1) | Transmit overrun error interrupt (SIO1) |
| Vector number setting register S | Receive-data-full interrupt (SIO1) | Transmit-data-empty interrupt (SIO1) |
| Vector number setting register T | Receive overrun error interrupt (SIO2) | Transmit overrun error interrupt (SIO2) |
| Vector number setting register U | Receive-data-full interrupt (SIO2) | Transmit-data-empty interrupt (SIO2) |

As table 5.6 shows, two on-chip peripheral module interrupts are assigned to each register. Set the vector numbers by setting the corresponding 7-bit groups (bits 14 to 8 and bits 6 to 0) with values in the range of H'00 (0000000) to H'7F (1111111). H'00 is vector number 0 (the lowest); H'7F is vector number 127 (the highest). The vector table address is calculated by the following equation.

$$\text{Vector table address} = \text{VBR} + (\text{vector number} \times 4)$$

A reset initializes a vector number setting register to H'0000. They are not initialized in standby mode.

**Table 5.7 Interrupt Request Sources and Vector Number Setting Registers (2)**

| **Register** | **Setting Function** |
|---|---|
| Vector number setting register DIV (VCRDIV) | Overflow interrupts for division unit |
| Vector number setting register DMAC0 (VCRDMA0) | Channel 0 transfer end interrupt for DMAC |
| Vector number setting register DMAC1 (VCRDMA1) | Channel 1 transfer end interrupt for DMAC |

As shown in table 5.7, the vector number for divider overflow interrupts is set in VCRDIV, and the vector numbers for direct memory access controller transfer-end interrupts are set in VCRDMA0 and VCRDMA1. See sections 9, Direct Memory Access Controller, and 10, Division Unit, for more details.

### 5.3.28 Interrupt Control Register (ICR)

ICR is a 16-bit register that sets the input signal detection mode of external interrupt input pin NMI and indicates the input signal level at the NMI pin. It can also specify IRQ or IRL mode by means of the External Interrupt Vector Mode Select bit. The IRQ/IRL interrupt vector number can be selected for setting in accordance with either auto vector mode or external vector mode by means of the Interrupt Vector Mode Select bit. ICR is initialized to H'8000 or H'0000 by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | NMIL | — | — | — | — | — | — | NMIE |
| Initial value: | 0/1* | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | EXIMD | VECMD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W |

Note: When NMI input is high: 1; when NMI input is low: 0

Bit 15—NMI Input Level (NMIL): Sets the level of the signal input at the NMI pin. This bit can be read to determine the NMI pin level. This bit cannot be modified.

| Bit 15: NMIL | Description |
|---|---|
| 0 | NMI input level is low |
| 1 | NMI input level is high |

Bits 14 to 9—Reserved: These bits always read 0. The write value should always be 0.

Bit 8—NMI Edge Select (NMIE): Selects whether the falling or rising edge of the interrupt request signal to the NMI pin is detected.

| Bit 8: NMIE | Description |
|---|---|
| 0 | Interrupt request is detected on falling edge of NMI input (Initial value) |
| 1 | Interrupt request is detected on rising edge of NMI input |

Bits 7 to 2—Reserved: These bits always read 0. The write value should always be 0.

Bit 1—External Interrupt Vector Mode Select (EXIMD): This bit selects IRQ mode or IRL mode. In IRQ mode, each of signals $\overline{IRL3}$ to $\overline{IRL0}$ functions as a separate interrupt source. In IRL mode, these signals can specify interrupt priority levels 1 to 15.

| Bit 1: EXIMD | Description | |
|---|---|---|
| 0 | IRL mode | (Initial value) |
| 1 | IRQ mode | |

Bit 0—Interrupt Vector Mode Select (VECMD): This bit selects auto-vector mode or external vector mode for IRL/IRQ interrupt vector number setting. In auto-vector mode, an internally determined vector number is set. The IRL15 and IRL14 interrupt vector numbers are set to 71 and the IRL1 vector number is set to 64. In external vector mode, a value between 0 and 127 can be input as the vector number from the external vector number input pins (D7–D0).

| Bit 0: VECMD | Description |
|---|---|
| 0 | Auto vector mode, vector number automatically set internally (Initial value) |
| 1 | External vector mode, vector number set by external input |

### 5.3.29 IRQ Control/Status Register (IRQCSR)

The IRQ control/status register (IRQCSR) is a 16-bit register that sets the $\overline{IRL0}$–$\overline{IRL3}$ input signal detection mode, indicates the input signal levels at pins $\overline{IRL0}$–$\overline{IRL3}$, and also indicates the IRQ interrupt status. IRQCSR is initialized by a reset. It is not initialized in standby mode.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IRQ31S | IRQ30S | IRQ21S | IRQ20S | IRQ11S | IRQ10S | IRQ01S | IRQ00S |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IRL3PS | IRL2PS | IRL1PS | IRL0PS | IRQ3F | IRQ2F | IRQ1F | IRQ0F |
| Initial value: | 0/1 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/(W) | R/(W) | R/(W) | R/(W) |

Bits 15 to 8—IRQ Sense Select Bits (IRQ31S–IRQ00S): These bits set the IRQ detection mode for $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$.

| Bits 15–8: IRQn1S | Bits 15–8: IRQn0S | Description | |
|---|---|---|---|
| 0 | 0 | Low-level detection | (Initial value) |
| | 1 | Rising-edge detection | |
| 1 | 0 | Falling-edge detection | |
| | 1 | Both-edge detection | |

Note: n = 0 to 3

Bits 7 to 4—IRL Pin Status Bits (IRL3PS–IRL0PS): These bits indicate the $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ pin status. The $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ pin levels can be ascertained by reading these bits. These bits cannot be modified.

| Bits 7–4: IRLnPS | Description |
|---|---|
| 0 | Low level is being input to pin $\overline{\text{IRLn}}$ |
| 1 | High level is being input to pin $\overline{\text{IRLn}}$ |

Note: n = 0 to 3

Bits 3 to 0—IRQ3 to IRQ0 Flags (IRQ3F–IRQ0F): These bits indicate the IRQ3–IRQ0 interrupt request status.

| Bits 3–0: IRQ3F–IRQ0F | Detection Setting | Description |
|---|---|---|
| 0 | Level detection | There is no IRQn interrupt request (Initial value)<br>[Clearing condition]<br>When $\overline{\text{IRLn}}$ input is high |
| | Edge detection | An IRQn interrupt request has not been detected (Initial value)<br>[Clearing conditions]<br>• When 0 is written to IRQnF after reading IRQnF = 1<br>• When an IRQn interrupt is accepted |
| 1 | Level detection | There is an IRQn interrupt request<br>[Setting condition]<br>When $\overline{\text{IRLn}}$ input is low |
| | Edge detection | An IRQn interrupt request has been detected<br>[Setting condition]<br>When an $\overline{\text{IRLn}}$ input edge is detected |

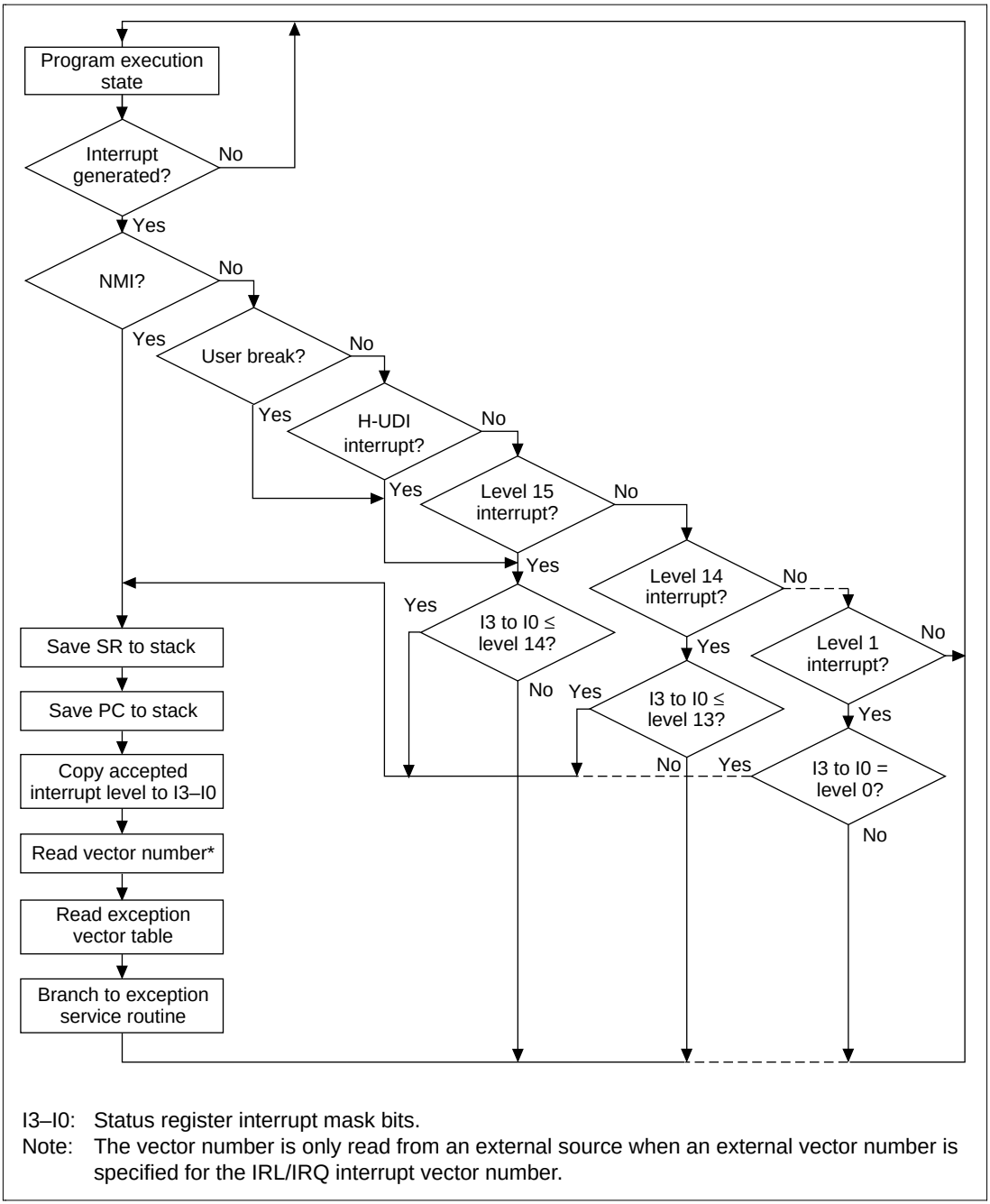
Note: n = 0 to 3

## 5.4 Interrupt Operation

### 5.4.1 Interrupt Sequence

The sequence of operations in interrupt generation is described below and illustrated in figure 5.5.

1. The interrupt request sources send interrupt request signals to the interrupt controller.
2. The interrupt controller selects the highest-priority interrupt among the interrupt requests sent, according to the priority levels set in interrupt priority level setting registers A to E (IPRA–IPRE). Lower-priority interrupts are held pending. If two or more of these interrupts have the same priority level or if multiple interrupts occur within a single module, the interrupt with the highest default priority or the highest priority within its IPR setting unit (as indicated in table 5.4) is selected.
3. The interrupt controller compares the priority level of the selected interrupt request with the interrupt mask bits (I3–I0) in the CPU's status register (SR). If the request priority level is equal to or less than the level set in I3–I0, the request is held pending. If the request priority level is higher than the level in bits I3–I0, the interrupt controller accepts the interrupt and sends an interrupt request signal to the CPU.
4. The CPU detects the interrupt request sent from the interrupt controller when it decodes the next instruction to be executed. Instead of executing the decoded instruction, the CPU starts interrupt exception handling.
5. Status register (SR) and program counter (PC) are saved onto the stack.
6. The priority level of the accepted interrupt is copied to the interrupt mask level bits (I3 to I0) in the status register (SR).
7. When external vector mode is specified for the IRL/IRQ interrupt, the vector number is read from the external vector number input pins (D7–D0).
8. The CPU reads the start address of the exception service routine from the exception vector table entry for the accepted interrupt, jumps to that address, and starts executing the program there. This jump is not a delayed branch.

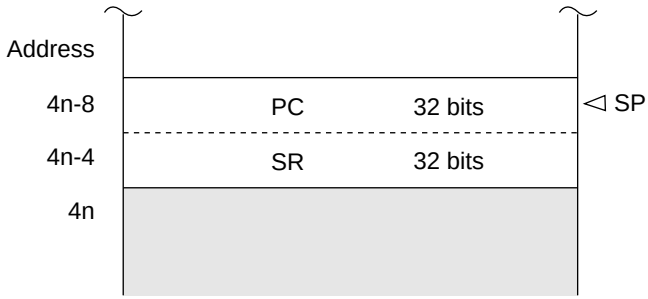



I3–I0: Status register interrupt mask bits.
Note: The vector number is only read from an external source when an external vector number is specified for the IRL/IRQ interrupt vector number.

**Figure 5.8 Interrupt Sequence Flowchart**

### 5.4.2 Stack State after Interrupt Exception Handling

The state of the stack after interrupt exception handling is completed is shown in figure 5.9.




PC: Start address of return destination instruction (instruction after executing instruction)

**Figure 5.9 Stack State after Interrupt Exception Handling**

## 5.5 Interrupt Response Time

Table 5.8 shows the interrupt response time, which is the time from the occurrence of an interrupt request until interrupt exception handling starts and fetching of the first instruction of the interrupt service routine begins.

The response time is shown here as the number of states. The actual response time can be calculated based on the clock mode and clock frequency used.

**Table 5.8 Interrupt Response Time**

| | | Number of States | | | | |
|---|---|---|---|---|---|---|
| | | | | Peripheral Module | | |
| **Item** | | **NMI** | **IRL/IRQ** | **A** | **B** | **Notes** |
| Compare identified interrupt priority with SR mask level | | 2.0 × Icyc | 0.5 × Icyc + 1.0 × Ecyc + 1.5 × Pcyc | 0.5 × Icyc + 1.0 × Pcyc | 1.0 × Pcyc | |
| Wait for completion of sequence currently being executed by CPU | | X (≥ 0) | X (≥ 0) | X (≥ 0) | X (≥ 0) | The longest sequence is for interrupt or address-error exception handling (X = 4.0 × Icyc + m1 + m2 + m3 + m4). If an interrupt-making instruction follows, however, the time may be even longer during repeat instruction execution |
| Time from interrupt exception handling (SR and PC saves and vector address fetch) until fetch of first instruction of exception service routine starts | | 5.0 × Icyc + m1 + m2 + m3 | 5.0 × Icyc + m1 + m2 + m3 | 5.0 × Icyc + m1 + m2 + m3 | 5.0 × Icyc + m1 + m2 + m3 | |
| Response time | Total: | X + 7.0 × Icyc + m1 + m2 + m3 | X + 5.5 × Icyc + 1.0 × Ecyc + 1.5 × Pcyc + m1 + m2 + m3 | X + 5.5 × Icyc + 1.0 × Pcyc + m1 + m2 + m3 | X + 5.0 × Icyc + 1.0 × Pcyc + m1 + m2 + m3 | |
| | Minimum: | 10 | 11 | 9.5 | 9 | Iϕ:Eϕ:Pϕ = 1:1:1 |
| | Maximum: | 11 + 2 (m1 + m2 + m3) + m4 | 19.5 + 2 (m1 + m2 + m3) + m4 | 13.5 + 2 (m1 + m2 + m3) + m4 | 13.0 + 2 (m1 + m2 + m3) + m4 | Iϕ:Eϕ:Pϕ = 1:1/4:1/4 |

Note: m1–m4 are the number of states needed for the following memory accesses
m1: SR save (longword write)
m2: PC save (longword write)

m3: Vector address read (longword write)
m4: Fetch of first instruction of interrupt service routine
Icyc: Iϕ cycle time
Ecyc: Eϕ cycle time
Pcyc: Pϕ cycle time
Peripheral modules A: DIVU, DMAC, REF (BSC)
Peripheral modules B: WDT, SCI, FRT, TPU, SCIF, SIO

## 5.6 Sampling of Pins $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$

Signals on interrupt pins $\overline{\text{IRL3}}$ to $\overline{\text{IRL0}}$ pass through the noise canceler before being sent by the interrupt controller to the CPU as interrupt requests, as shown in figure 5.10. The noise canceler cancels noise that changes in short cycles. The CPU samples the interrupt requests between executing instructions. During this period, the noise canceler output changes according to the noise-eliminated pin level, so the pin level must be held until the CPU samples it. This means that interrupt sources generally must not be cleared inside interrupt routines.

When an external vector is fetched, the interrupt source can also be cleared when the external vector fetch cycle is detected.

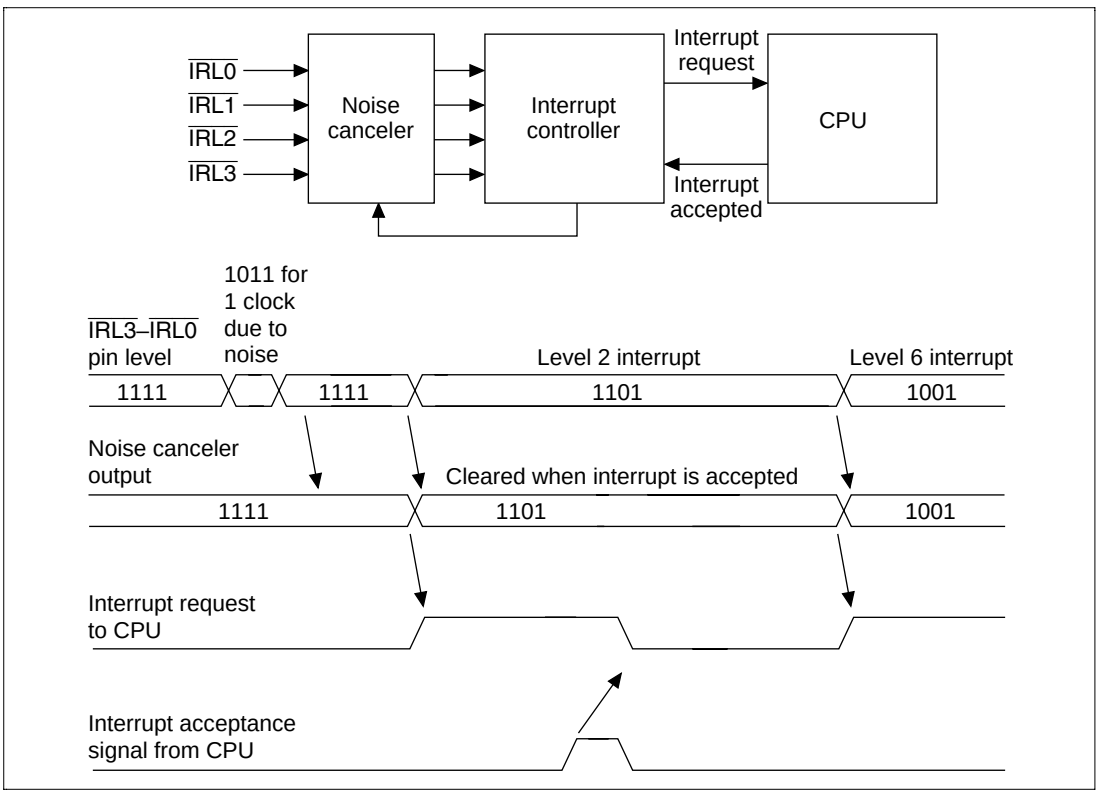



**Figure 5.10 $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ Pin Sampling**

## 5.7 Usage Notes

1. Note on module standby execution

   Do not execute module standby for modules that have the module standby function when the possibility remains that an interrupt request may be output.

2. Notes on interrupt source clearing

   A. When clearing external interrupt source

   If interrupt source clearing is performed by writing to an IO address (external), the next instruction will be executed before completion of the write operation because of the write buffer. To ensure that the write operation is completed before the next instruction is executed, synchronization is achieved when a read is performed from the same address following the write.

   a. When returning from interrupt handling by means of RTE instruction

   When the RTE instruction is used to return from interrupt handling, as shown in figure 5.11, consider the cycles to be inserted between the read instruction for synchronization and the RTE instruction, according to the set clock ratio (Iϕ : Eϕ : Pϕ) and external bus cycle.

   IRL3—IRL0 should be negated at least 0.5 Icyc + 1.0 Ecyc + 1.5 Pcyc before interrupt acceptance becomes possible.

   For example, if clock ratio Iϕ : Eϕ : Pϕ is 4 : 2 : 2, at least 5.5 Icyc should be inserted.

   b. When changing level during interrupt handling

   When the SR value is changed by means of an LDC instruction and multiple implementation of other interrupts is enabled, also, consider the cycles to be inserted between the synchronization instruction and the LDC instruction as shown in figure 5.12, according to the set clock ratio (Iϕ : Eϕ : Pϕ) and external bus cycle.

   IRL3–IRL0 should be negated at least 0.5 Icyc + 1.0 Ecyc + 1.5 Pcyc before interrupt acceptance becomes possible.

   For example, if clock ratio Iϕ : Eϕ : Pϕ is 4 : 2 : 2, at least 5.5 Icyc should be inserted.

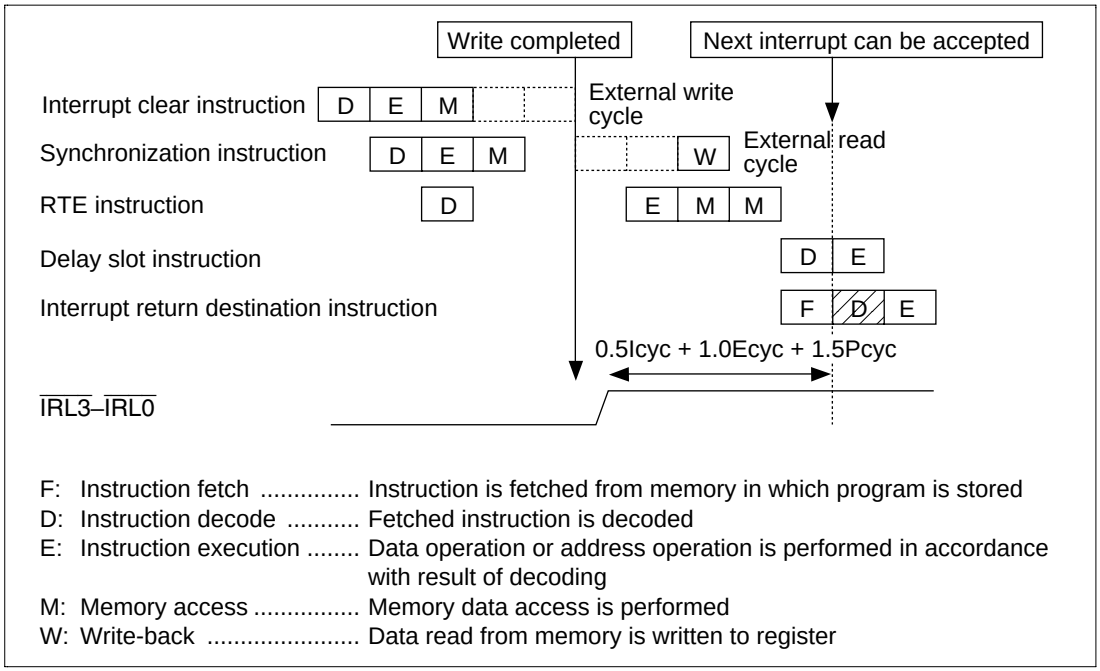



**Figure 5.11 Pipeline Operation when Returning by Means of RTE Instruction**

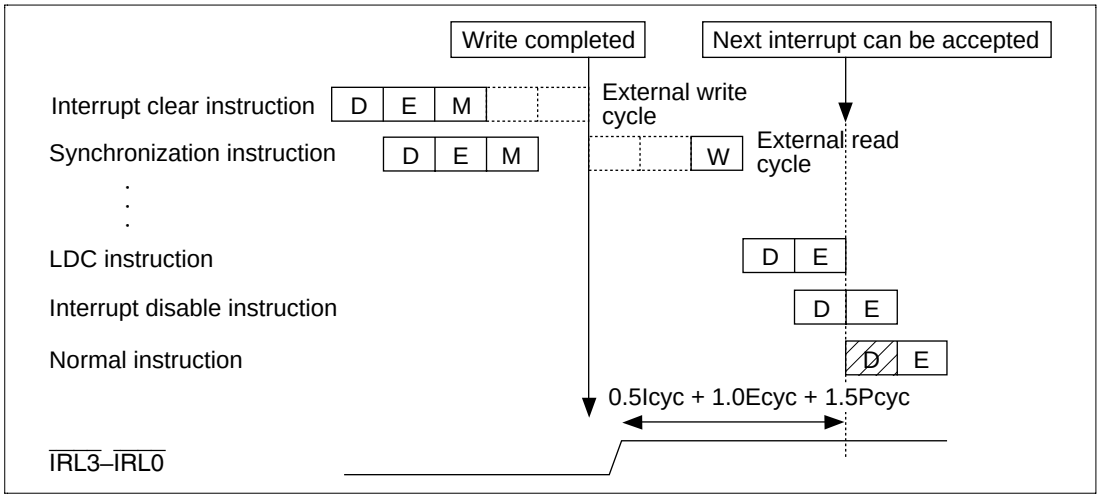



**Figure 5.12 Pipeline Operation when Interrupts are Enabled by Means of SR Modification**

When an interrupt source is cleared by the program, pipeline operation must be considered to ensure that the same interrupt is not implemented again.

B. When clearing on-chip interrupt source

When an interrupt source is from an on-chip peripheral module, also, pipeline operation must be considered to ensure that the same interrupt is not implemented again. An interval of 0.5 Icyc + 1.0 Pcyc is required until an on-chip peripheral module interrupt is identified by the CPU. Similarly, an interval of 0.5 Icyc + 1.0 Pcyc is also necessary to report the fact that an interrupt request is no longer present.

a. When returning from interrupt handling by means of RTE instruction

When the RTE instruction is used to return from interrupt handling, as shown in figure 5.13, consider the cycles to be inserted between the read instruction for synchronization and the RTE instruction, according to the set clock ratio (Iϕ : Eϕ : Pϕ).

The on-chip peripheral interrupt signal should be negated at least 0.5 Icyc + 1.0 Pcyc before interrupt acceptance becomes possible.

For example, if clock ratio Iϕ : Eϕ : Pϕ is 4 : 2 : 2, at least 2.5 Icyc should be inserted.

b. When changing level during interrupt handling

When the SR value is changed by means of an LDC instruction and multiple implementation of other interrupts is enabled, consider the cycles to be inserted between the synchronization instruction and the LDC instruction as shown in figure 5.14, according to the set clock ratio (Iϕ : Eϕ : Pϕ).

The on-chip peripheral interrupt signal should be negated at least 0.5 Icyc + 1.0 Pcyc before interrupt acceptance becomes possible.

For example, if clock ratio Iϕ : Eϕ : Pϕ is 4 : 2 : 2, at least 2.5 Icyc should be inserted.

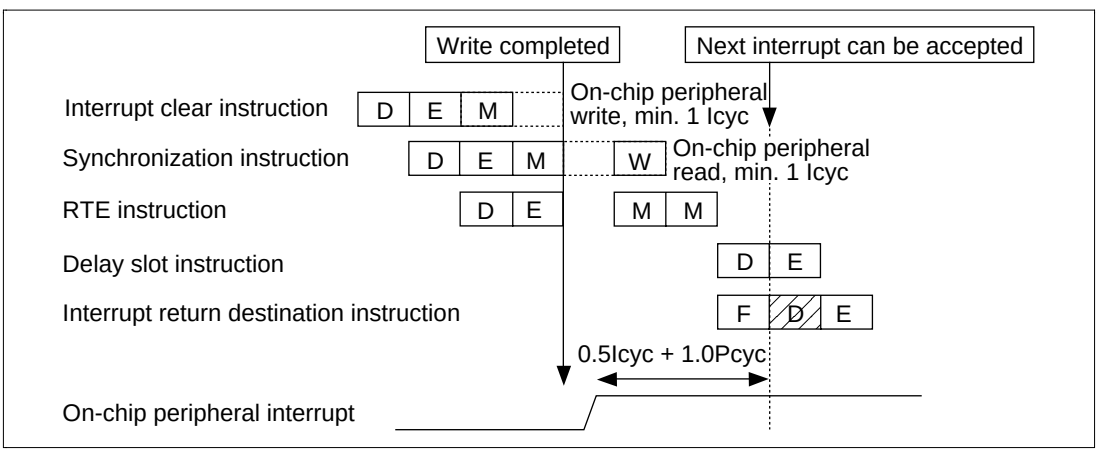



**Figure 5.13 Pipeline Operation when Returning by Means of RTE Instruction**

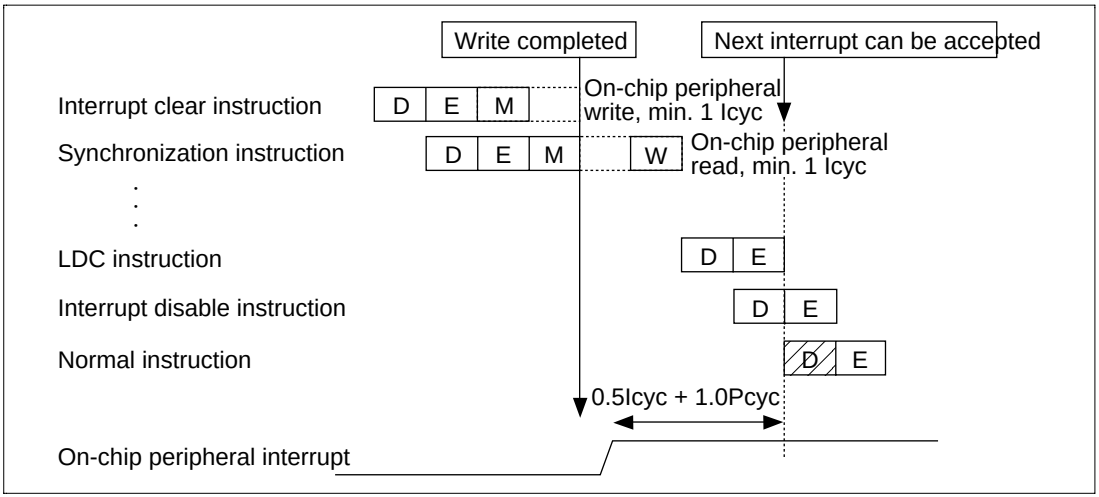



**Figure 5.14 Pipeline Operation when Interrupts are Enabled by Means of SR Modification**

In the above figure, the stage in which the instruction fetch occurs cannot be specified because of the mix of DSP instructions in this chip, so instruction fetch F is omitted in most cases during pipeline operation.

# Section 6 User Break Controller (UBC)

## 6.1 Overview

The user break controller (UBC) provides functions that simplify program debugging. Break conditions are set in the UBC and a user break interrupt is generated according to the conditions of the bus cycle generated by the CPU and on-chip DMAC.

This function makes it easy to design an effective self-monitoring debugger, enabling the chip to debug programs without using an in-circuit emulator.

### 6.1.1 Features

The UBC has the following features:

- The following break conditions can be set
  - Two break channels (channel A, channel B)
  - User break interrupts can be generated on independent conditions for channels A, and B, or on sequential conditions (sequential break setting: channel A → Channel B)
  - Address: 32-bit maskable, individual address setting possible (cache bus (CPU), internal bus (DMAC), X/Y bus)
  - Data (channel B only): 32-bit maskable, individual address setting possible (cache bus (CPU), internal bus (DMAC), X/Y bus)
  - Bus master: CPU cycle/DMAC cycle
  - Bus cycle: instruction fetch/data access
  - Read/write
  - Operand size: byte/word/longword
- User break interrupt generation on occurrence of break condition

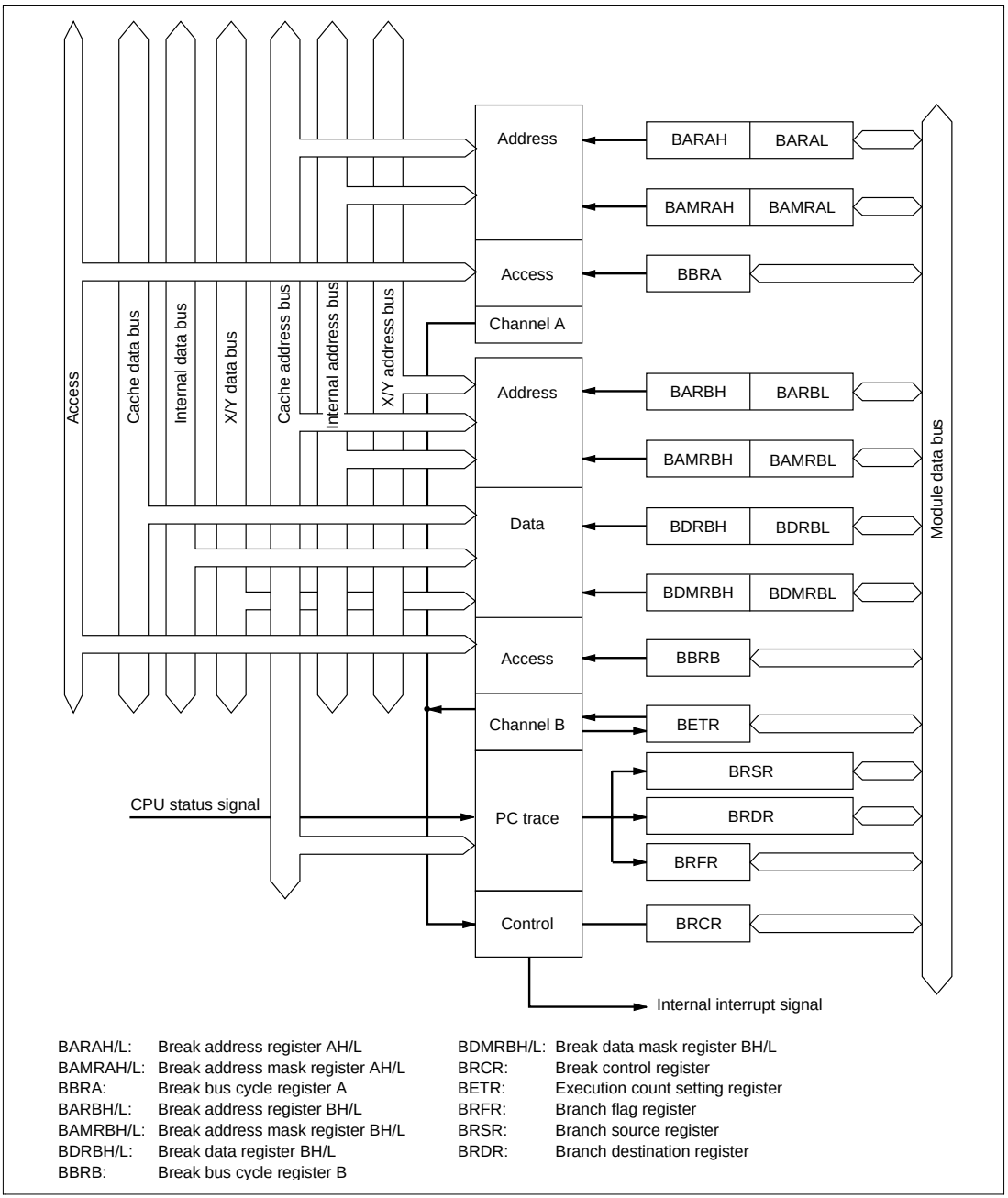
  A user-written user break interrupt handling routine can be executed.
- When an instruction fetch cycle is set as a condition, it is possible to select interrupt generation before instruction execution or interrupt handling generation after instruction execution.
- Multiple-execution specification break (channel B only)

  Settable number of executions: Max. $2^{12} - 1$ (4095)
- PC trace function

  The branch source/branch destination can be traced when a branch instruction is executed (max. 4 pairs, 8 addresses)

### 6.1.2 Block Diagram

Figure 6.1 shows a block diagram of the UBC.




BARAH/L: Break address register AH/L
BAMRAH/L: Break address mask register AH/L
BBRA: Break bus cycle register A
BARBH/L: Break address register BH/L
BAMRBH/L: Break address mask register BH/L
BDRBH/L: Break data register BH/L
BBRB: Break bus cycle register B
BDMRBH/L: Break data mask register BH/L
BRCR: Break control register
BETR: Execution count setting register
BRFR: Branch flag register
BRSR: Branch source register
BRDR: Branch destination register

**Figure 6.1 User Break Controller Block Diagram**

### 6.1.3 Register Configuration

Table 6.1 shows the UBC register configuration.

**Table 6.1 Register Configuration**

| Name | Abbr. | R/W | Initial Value*1 | Address | Access Size*2 | |
|---|---|---|---|---|---|---|
| Break address register AH | BARAH | R/W | H'0000 | H'FFFFFF40 | 16 | 32 |
| Break address register AL | BARAL | R/W | H'0000 | H'FFFFFF42 | 16 | |
| Break address mask register AH | BAMRAH | R/W | H'0000 | H'FFFFFF44 | 16 | 32 |
| Break address mask register AL | BAMRAL | R/W | H'0000 | H'FFFFFF46 | 16 | |
| Break bus cycle register A | BBRA | R/W | H'0000 | H'FFFFFF48 | 16, 32 | |
| Break address register BH | BARBH | R/W | H'0000 | H'FFFFFF60 | 16 | 32 |
| Break address register BL | BARBL | R/W | H'0000 | H'FFFFFF62 | 16 | |
| Break address mask register BH | BAMRBH | R/W | H'0000 | H'FFFFFF64 | 16 | 32 |
| Break address mask register BL | BAMRBL | R/W | H'0000 | H'FFFFFF66 | 16 | |
| Break data register BH | BDRBH | R/W | H'0000 | H'FFFFFF70 | 16 | 32 |
| Break data register BL | BDRBL | R/W | H'0000 | H'FFFFFF72 | 16 | |
| Break data mask register BH | BDMRBH | R/W | H'0000 | H'FFFFFF74 | 16 | 32 |
| Break data mask register BL | BDMRBL | R/W | H'0000 | H'FFFFFF76 | 16 | |
| Break bus cycle register B | BBRB | R/W | H'0000 | H'FFFFFF68 | 16, 32 | |
| Break control register | BRCR | R/W | H'0000 | H'FFFFFF78 | 16, 32 | |
| Execution count setting register | BETR | R/W | H'0000 | H'FFFFFF4C | 16, 32 | |
| Branch flag register | BRFR | R | *3 | H'FFFFFF50 | 16, 32 | |
| Branch source register | BRSR | R | Unde-fined | H'FFFFFF54 | 16, 32 | |
| Branch destination register | BRDR | R | Unde-fined | H'FFFFFF58 | 16, 32 | |

Notes: 1. Initialized by a power-on reset. Values held in standby mode. Value undefined after a manual reset.
2. Byte access not permitted.
3. Bits SVF and DVF in BRFR are initialized by a power-on reset, but the other bits in BRFR are not.

## 6.2 Register Descriptions

### 6.2.1 Break Address Register A (BARA)

**BARAH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAA31 | BAA30 | BAA29 | BAA28 | BAA27 | BAA26 | BAA25 | BAA24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAA23 | BAA22 | BAA21 | BAA20 | BAA19 | BAA18 | BAA17 | BAA16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BARAL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAA15 | BAA14 | BAA13 | BAA12 | BAA11 | BA10 | BAA9 | BAA8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAA7 | BAA6 | BAA5 | BAA4 | BAA3 | BAA2 | BAA1 | BAA0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break address registers A—break address register AH (BARAH) and break address register AL (BARAL)—together form a single group. Both are 16-bit read/write register. BARAH specifies the upper bits (bits 31 to 16) of the address of the channel A break condition, while BARAL specifies the lower bits (bits 15 to 0). A power-on reset initializes both BARAH and BARAL to H'0000. Their values are undefined after a manual reset.

BARAH Bits 15 to 0—Break Address A 31 to 16 (BAA31 to BAA16): These bits store the upper bit values (bits 31 to 16) of the address of the channel A break condition.

BARAL Bits 15 to 0—Break Address A 15 to 0 (BAA15 to BAA0): These bits store the lower bit values (bits 15 to 0) of the address of the channel A break condition.

### 6.2.2 Break Address Mask Register A (BAMRA)

**BAMRAH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMA31 | BAMA30 | BAMA29 | BAMA28 | BAMA27 | BAMA26 | BAMA25 | BAMA24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMA23 | BAMA22 | BAMA21 | BAMA20 | BAMA19 | BAMA18 | BAMA17 | BAMA16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BAMRAL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMA15 | BAMA14 | BAMA13 | BAMA12 | BAMA11 | BAMA10 | BAMA9 | BAMA8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMA7 | BAMA6 | BAMA5 | BAMA4 | BAMA3 | BAMA2 | BAMA1 | BAMA0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break address mask registers A (BAMRA)—break address mask register AH (BAMRAH) and break address mask register AL (BAMRAL)—together form a single group. Both are 16-bit read/write registers. BAMRAH determines which of the bits in the break address set in BARAH are masked. BAMRAL determines which of the bits in the break address set in BARAL are masked. A power-on reset initializes BAMRAH and BAMRAL to H'0000. Their values are undefined after a manual reset.

BAMRAH Bits 15 to 0—Break Address Mask A 31 to 16 (BAMA31 to BAMA16): These bits specify whether bits 31 to 16 (BAA31 to BAA16) of the channel A break address set in BARAH are masked.

BAMRAL Bits 15 to 0—Break Address Mask A 15 to 0 (BAMA15 to BAMA0): These bits specify whether bits 15 to 0 (BAA15 to BAA0) of the channel A break address set in BARAL are masked.

| Bits 31–0: BAMAn | Description |
|---|---|
| 0 | Channel A break address BAAn is included in the break conditions (Initial value) |
| 1 | Channel A break address BAAn is masked and therefore not included in the break conditions |

Note: n = 31 to 0

### 6.2.3 Break Bus Cycle Register A (BBRA)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CPA1 | CPA0 | IDA1 | IDA0 | RWA1 | RWA0 | SZA1 | SZA0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The break bus cycle register A (BBRA) is a 16-bit read/write register that selects the following four channel A break conditions:

1. CPU cycle/DMAC cycle
2. Instruction fetch/data access
3. Read/write
4. Operand size

A power-on reset initializes BBRA to H'0000. Its value is undefined after a manual reset.

Bits 15 to 8—Reserved: These bits always read 0. The write value should always be 0.

Bits 7 and 6—CPU Cycle/DMAC Cycle Select A (CPA1, CPA0): These bits select whether to break channel A on a CPU and/or DMAC bus cycle. When the DMAC cycle setting is made, on-chip DMAC cycles are always included in the break conditions.

| Bit 7: CPA1 | Bit 6: CPA0 | Description | |
|---|---|---|---|
| 0 | 0 | No channel A user break interrupt occurs | (Initial value) |
| | 1 | Break only on CPU cycles | |
| 1 | 0 | Break only on DMAC cycles | |
| | 1 | Break on both CPU and DMAC cycles | |

Bits 5 and 4—Instruction Fetch/Data Access Select A (IDA1, IDA0): These bits select whether to break channel A on instruction fetch and/or data access cycles.

| Bit 5: IDA1 | Bit 4: IDA0 | Description | |
|---|---|---|---|
| 0 | 0 | No channel A user break interrupt occurs | (Initial value) |
| | 1 | Break only on instruction fetch cycles | |
| 1 | 0 | Break only on data access cycles | |
| | 1 | Break on both instruction fetch and data access cycles | |

Bits 3 and 2—Read/Write Select A (RWA1, RWA0): These bits select whether to break channel A on read and/or write cycles.

| Bit 3: RWA1 | Bit 2: RWA0 | Description | |
|---|---|---|---|
| 0 | 0 | No channel A user break interrupt occurs | (Initial value) |
| | 1 | Break only on read cycles | |
| 1 | 0 | Break only on write cycles | |
| | 1 | Break on both read and write cycles | |

Bits 1 and 0—Operand Size Select A (SZA1, SZA0): These bits select bus cycle operand size as a channel A break condition.

| Bit 1: SZA1 | Bit 0: SZA0 | Description | |
|---|---|---|---|
| 0 | 0 | Operand size is not a break condition | (Initial value) |
| | 1 | Break on byte access | |
| 1 | 0 | Break on word access | |
| | 1 | Break on longword access | |

Note: When breaking on an instruction fetch, set the SZA0 bit to 0. All instructions are considered to be word-size accesses (instruction fetches are always longword). Operand size is word for instructions or determined by the operand size specified for the CPU/DMAC data access. It is not determined by the bus width of the space being accessed.

### 6.2.4 Break Address Register B (BARB)

**BARBH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAB31 | BAB30 | BAB29 | BAB28 | BAB27 | BAB26 | BAB25 | BAB24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAB23 | BAB22 | BAB21 | BAB20 | BAB19 | BAB18 | BAB17 | BAB16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BARBL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAB15 | BAB14 | BAB13 | BAB12 | BAB11 | BAB10 | BAB9 | BAB8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAB7 | BAB6 | BAB5 | BAB4 | BAB3 | BAB2 | BAB1 | BAB0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break address registers B (BARB)—break address register BH (BARBH) and break address register BL (BARBL)—together form a single group. Both are 16-bit read/write registers. BARBH stores the upper bits (bits 31 to 16) of the address of the channel B break condition, while BARBL stores the lower bits (bits 15 to 0). In addition, the address bus connected to X/Y memory can also be specified as a break condition by means of XYE bit/XYS bit settings in break bus cycle register B (BBRB). When XYE = 0, BAB31–BAB0 specify the address; when XYE = 1, the upper 16 bits of BARB (bits BAB31–BAB16) specify the X address bus, and the lower 16 bits (bits BAB15–BAB0) specify the Y address bus. A power-on reset initializes both BARBH and BARBL to H'0000. Their values are undefined after a manual reset.

- BARB configuration

| | | Upper 16 bits (BAB31–BAB16) | Lower 16 bits (BAB15–BAB0) |
|---|---|---|---|
| XYE = 0 | Address | Upper 16 bits of address bus | Lower 16 bits of address bus |
| XYE = 1 | X address (when XYS = 0) | X address (XAB15–XAB1)* | — |
| | Y address (when XYS = 1) | — | Y address (YAB15–YAB1)* |

Note: * As X/Y bus accesses are always word accesses, the values of XAB0 and YAB0 are not included in the break conditions.

### 6.2.5 Break Address Mask Register B (BAMRB)

**BAMRBH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMB31 | BAMB30 | BAMB29 | BAMB28 | BAMB27 | BAMB26 | BAMB25 | BAMB24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMB23 | BAMB22 | BAMB21 | BAMB20 | BAMB19 | BAMB18 | BAMB17 | BAMB16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BAMRBL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMB15 | BAMB14 | BAMB13 | BAMB12 | BAMB11 | BAMB10 | BAMB9 | BAMB8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BAMB7 | BAMB6 | BAMB5 | BAMB4 | BAMB3 | BAMB2 | BAMB1 | BAMB0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break address mask registers B (BAMRB)—break address mask register BH (BAMRBH) and break address mask register BL (BAMRBL)—together form a single group. Both are 16-bit read/write registers. BAMRBH determines which of the bits in the break address set in BARBH are masked. BAMRBL determines which of the bits in the break address set in BARBL are masked. The function of these registers is affected by bits XYE and XYS in break bus cycle register B (BBRB) as shown below.

- BAMRB configuration

| | | **Upper 16 bits (BAMB31–BAMB16)** | **Lower 16 bits (BAMB15–BAMB0)** |
|---|---|---|---|
| XYE = 0 | Address | Upper 16 bits maskable | Lower 16 bits maskable |
| XYE = 1 | X address (when XYS = 0) | Maskable | — |
| | Y address (when XYS = 1) | — | Maskable |

| **Bits 31–0: BAMBn** | **Description** |
|---|---|
| 0 | Channel B break address BABn is included in the break conditions (Initial value) |
| 1 | Channel B break address BABn is masked and therefore not included in the break conditions |

Note: n = 31 to 0

### 6.2.6 Break Data Register B (BDRB)

**BDRBH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDB31 | BDB30 | BDB29 | BDB28 | BDB27 | BDB26 | BDB25 | BDB24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDB23 | BDB22 | BDB21 | BDB20 | BDB19 | BDB18 | BDB17 | BDB16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BDRBL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDB15 | BDB14 | BDB13 | BDB12 | BDB11 | BDB10 | BDB9 | BDB8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDB7 | BDB6 | BDB5 | BDB4 | BDB3 | BDB2 | BDB1 | BDB0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break data registers B (BDRB)—break data register BH (BDRBH) and break data register BL (BDRBL)—together form a single group. Both are 16-bit read/write registers. BDRBH specifies the upper half (bits 31–16) of the data that is the break condition for channel B, while BDRBL specifies the lower half (bits 15–0). In addition, the data bus connected to X/Y memory can also be specified as a break condition by means of XYE bit/XYS bit settings in break bus cycle register B (BBRB). When XYE = 1, the upper 16 bits of BDRB (bits BDB31–BDB16) specify the X data bus, and the lower 16 bits (bits BDB15–BDB0) specify the Y data bus.

- BDRB configuration

| | | Upper 16 bits (BDB31–BDB16) | Lower 16 bits (BDB15–BDB0) |
|---|---|---|---|
| XYE = 0 | Data | Upper 16 bits of data bus | Lower 16 bits of data bus |
| XYE = 1 | X data (when XYS = 0) | X data (XDB15–XDB0) | — |
| | Y data (when XYS = 1) | — | Y data (YDB15–YDB0) |

### 6.2.7 Break Data Mask Register B (BDMRB)

**BDMRBH:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDMB31 | BDMB30 | BDMB29 | BDMB28 | BDMB27 | BDMB26 | BDMB25 | BDMB24 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDMB23 | BDMB22 | BDMB21 | BDMB20 | BDMB19 | BDMB18 | BDMB17 | BDMB16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**BDMRBL:**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDMB15 | BDMB14 | BDMB13 | BDMB12 | BDMB11 | BDMB10 | BDMB9 | BDMB8 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDMB7 | BDMB6 | BDMB5 | BDMB4 | BDMB3 | BDMB2 | BDMB1 | BDMB0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The two break data mask registers B (BDMRB)—break data mask register BH (BDMRBH) and break data mask register BL (BDMRBL)—together form a single group. Both are 16-bit read/write registers. BDMRBH determines which of the bits in the break address set in BDRBH are masked. BDMRBL determines which of the bits in the break address set in BDRBL are masked. The function of these registers is affected by bits XYE and XYS in break bus cycle register B (BBRB) as shown below. A power-on reset initializes BDMRBH and BDMRBL to H'0000. Their values are undefined after a manual reset.

- BDMRB configuration

| | | **Upper 16 bits (BDMB31–BDMB16)** | **Lower 16 bits (BDMB15–BDMB0)** |
|---|---|---|---|
| XYE = 0 | Data | Upper 16 bits maskable | Lower 16 bits maskable |
| XYE = 1 | X data (when XYS = 0) | Maskable | — |
| | Y data (when XYS = 1) | — | Maskable |

| **Bits 31–0: BDMBn** | **Description** |
|---|---|
| 0 | Channel B break address bit BDBn is included in the break condition |
| 1 | Channel B break address bit BDBn is masked and therefore not included in the break condition |

n = 31 to 0

Notes: 1. When the data bus value is included in the break conditions, specify the operand size.
2. When byte-size is specified and odd-address data is used as a break condition, set the value in bits 7 to 0 of BDRB and BDMRB. When even-address data is used as a break condition, set the value in bits 15 to 8. The unused 8 bits do not affect the break conditions.

### 6.2.8 Break Bus Cycle Register B (BBRB)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | XYE | XYS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CPB1 | CPB0 | IDB1 | IDB0 | RWB1 | RWB0 | SZB1 | SZB0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Break bus cycle register B (BBRB) is a 16-bit read/write register that selects the following five channel B break conditions:

1. Internal bus (C bus, I bus)/X memory bus/Y memory bus
2. CPU cycle/DMAC cycle
3. Instruction fetch/data access
4. Read/write
5. Operand size

A power-on reset initializes BBRB to H'0000. Its value is undefined after a manual reset.

Bits 15 to 10—Reserved: These bits always read 0. The write value should always be 0.

Bit 9—X/Y Memory Bus Enable (XYE): Selects whether the X/Y bus is used for channel B break conditions.

| Bit 9: XYE | Description |
|---|---|
| 0 | Cache bus or internal bus is condition for channel B address/data (Initial value) |
| 1 | X/Y bus is condition for channel B address/data |

Bit 8—X Bus/Y Bus Select (XYS): Selects whether the X bus or Y bus is used for channel B break conditions. This bit is only valid when the XYE bit is set to 1.

| Bit 8: XYS | Description |
|---|---|
| 0 | X bus is used for channel B break condition (Initial value) |
| 1 | Y bus is used for channel B break condition |

Bits 7 to 0 have the same configuration as in BBRA.

### 6.2.9 Break Control Register (BRCR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CMFCA | CMFPA | EBBE | UMD | PCTE | PCBA | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CMFCB | CMFPB | ETBE | SEQ | DBEB | PCBB | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

The BRCR register:

1. Determines whether to use channels A and B as two independent channels or as sequential conditions.
2. Selects whether to break before or after instruction execution during the instruction fetch cycle.
3. Selects whether to include the data bus in channel B comparison conditions.
4. Selects whether an execution times break is set.
5. Selects whether a PC trace is executed.

It also has a flag that is set when conditions match. A power-on reset initializes BRCR to H'0000. Its value is undefined after a manual reset.

Bit 15—CPU Condition-Match Flag A (CMFCA): Set to 1 when CPU bus cycle conditions included in the break conditions set for channel A are met. Not cleared to 0 (once set, it must be cleared by a write before it can be used again).

| Bit 15: CMFCA | Description |
|---|---|
| 0 | Channel A CPU cycle conditions do not match, no user break interrupt generated (Initial value) |
| 1 | Channel A CPU cycle conditions have matched, user break interrupt generated |

Bit 14—DMAC Condition-Match Flag A (CMFPA): Set to 1 when DMAC bus cycle conditions included in the break conditions set for channel A are met. Not cleared to 0 (once set, it must be cleared by a write before it can be used again).

| Bit 14: CMFPA | Description |
|---|---|
| 0 | Channel A DMAC cycle conditions do not match, no user break interrupt generated (Initial value) |
| 1 | Channel A DMAC cycle conditions have matched, user break interrupt generated |

Bit 13—External Bus Break Enable (EBBE): In the SH7604, this bit monitors the external bus master's address bus when the bus is released, and includes the external bus master's bus cycle in the bus cycle select conditions (CPA1, CPB1). However, this bit is not supported in this chip.

Bit 12—UBC Mode (UMD): In the SH7604, this bit selects SH7000 series-compatible mode or SH7604 mode as the operating mode. However, this bit is not supported in this chip.

Bit 11—PC Trace Enable (PCTE): Selects whether a PC trace is executed.

| Bit 11: PCTE | Description |
|---|---|
| 0 | PC trace is not executed (Initial value) |
| 1 | PC trace is executed |

Bit 10—PC Break Select A (PCBA): Selects whether to place the channel A break in the instruction fetch cycle before or after instruction execution.

| Bit 10: PCBA | Description |
|---|---|
| 0 | Places the channel A instruction fetch cycle break before instruction execution (Initial value) |
| 1 | Places the channel A instruction fetch cycle break after instruction execution |

Bits 9 and 8—Reserved: These bits always read 0. The write value should always be 0.

Bit 7—CPU Condition-Match Flag B (CMFCB): Set to 1 when CPU bus cycle conditions included in the break conditions set for channel B are met. Not cleared to 0 (once set, it must be cleared by a write before it can be used again).

| Bit 7: CMFCB | Description |
|---|---|
| 0 | Channel B CPU cycle conditions do not match, no user break interrupt generated (Initial value) |
| 1 | Channel B CPU cycle conditions have matched, user break interrupt generated |

Bit 6—DMAC Condition-Match Flag B (CMFPB): Set to 1 when DMAC bus cycle conditions included in the break conditions set for channel B are met. Not cleared to 0 (once set, it must be cleared by a write before it can be used again).

| Bit 6: CMFPB | Description |
|---|---|
| 0 | Channel B DMAC cycle conditions do not match, no user break interrupt generated (Initial value) |
| 1 | Channel B DMAC cycle conditions have matched, user break interrupt generated |

Bit 5—Execution Times Specification Break Enable (ETBE): Enables the execution times break conditions (channel B only). When this bit is set to 1, a user break interrupt is requested when the number of break condition occurrences equals the execution times setting in the execution times break register (BETR).

| Bit 5: ETBE | Description |
|---|---|
| 0 | Channel B execution times break condition is disabled (Initial value) |
| 1 | Channel B execution times break condition is enabled |

Bit 4—Sequence Condition Select (SEQ): Selects whether to handle the channel A and B conditions independently or sequentially.

| Bit 4: SEQ | Description |
|---|---|
| 0 | Channel A and B conditions compared independently (Initial value) |
| 1 | Channel A and B conditions compared sequentially (channel A, then channel B) |

Bit 3—Data Break Enable B (DBEB): Selects whether to include data bus conditions in the channel B break conditions.

| Bit 3: DBEB | Description |
|---|---|
| 0 | Data bus conditions not included in the channel B conditions (Initial value) |
| 1 | Data bus conditions included in the channel B conditions |

Bit 2—Instruction Break Select B (PCBB): Selects whether to place the channel B instruction fetch cycle break before or after instruction execution.

| Bit 2: PCBB | Description |
|---|---|
| 0 | Places the channel B instruction fetch cycle break before instruction execution (Initial value) |
| 1 | Places the channel B instruction fetch cycle break after instruction execution |

Bits 1 and 0—Reserved: These bits always read 0. The write value should always be 0.

### 6.2.10 Execution Times Break Register (BETR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The execution times break register (BETR) is a 16-bit register that specifies the number of occurrences of channel B break conditions before a user break interrupt is requested after channel B execution times break conditions are enabled (by setting bit ETBE in BRCR). The maximum value is $2^{12} - 1$. A power-on reset initializes BETR to H'0000. BETR is decremented by 1 each time a channel B break condition is satisfied. After BETR reaches H'0001, an interrupt is requested when a break condition is satisfied.

Exceptions and interrupts are not accepted for instructions in a repeat loop comprising up to three instructions (see section 4.6). Therefore, BETR is not decremented by a break condition match for an instruction in a repeat loop comprising up to three instructions.

Bits 15 to 12—Reserved: These bits always read 0. The write value should always be 0.

### 6.2.11 Branch Source Register (BRSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BSA31 | BSA30 | BSA29 | BSA28 | BSA27 | BSA26 | BSA25 | BSA24 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BSA23 | BSA22 | BSA21 | BSA20 | BSA19 | BSA18 | BSA17 | BSA16 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BSA15 | BSA14 | BSA13 | BSA12 | BSA11 | BSA10 | BSA9 | BSA8 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BSA7 | BSA6 | BSA5 | BSA4 | BSA3 | BSA2 | BSA1 | BSA0 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

The branch source register (BRSR) comprises a set of four 32-bit read-only registers that store the last address fetched before the previous branch. These values are used to calculate the address of the last instruction executed before a branch when a PC trace is performed. BRSR has a FIFO (first-in-first-out) queue structure for PC trace use. The queue shifts on each branch.

BRSR is not initialized by a reset.

### 6.2.12 Branch Destination Register (BRDR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDA31 | BDA30 | BDA29 | BDA28 | BDA27 | BDA26 | BDA25 | BDA24 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDA23 | BDA22 | BDA21 | BDA20 | BDA19 | BDA18 | BDA17 | BDA16 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDA15 | BDA14 | BDA13 | BDA12 | BDA11 | BDA10 | BDA9 | BDA8 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | BDA7 | BDA6 | BDA5 | BDA4 | BDA3 | BDA2 | BDA1 | BDA0 |
| Initial value: | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R |

The branch destination register (BRDR) comprises a set of four 32-bit read-only registers that store branch destination fetch addresses to be used in a PC trace. BRDR has a FIFO (first-in-first-out) queue structure for PC trace use. The queue shifts on each branch.

BRDR is not initialized by a reset.

### 6.2.13 Branch Flag Register (BRFR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | SVF | PID2 | PID1 | PID0 | — | — | — | — |
| Initial value: | 0 | * | * | * | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | DVF | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

The branch flag register (BRFR) comprises a set of four 16-bit read-only registers that contain flags indicating whether the address is stored in BRSR and BRDR when a branch is made (branch instruction, repeat, interrupt, etc.), and three bit pointers indicating the number of cycles from fetch until execution for the last instruction executed. BRFR has a FIFO (first-in-first-out) queue structure for PC trace use. The queue shifts on each branch.

A reset initializes bits SVF and DVF but does not initialize bits PID2 to PID0.

Bit 15—BRSR Verification Flag (SVF): This flag indicates whether BRSR contains an address and pointer that enable the branch source address to be calculated. SVF is set when a branch destination address instruction is fetched, and reset when BRSR is read.

| **Bit 15: SVF** | **Description** | |
|---|---|---|
| 0 | BRSR value is invalid | (Initial value) |
| 1 | BRSR value is valid | |

Bits 14 to 12—PID2 to PID0: These bits are pointers that indicate the number of the instruction buffer for the instruction executed immediately before the instruction at which a branch occurred.

| **Bits 14–12: PID2–PID0** | **Description** |
|---|---|
| Even | PID indicates instruction buffer number |
| Odd | PID+2 indicates instruction buffer number |

Bit 7—BRDR Verification Flag (DVF): This flag indicates whether BRDR contains the branch destination address. DVF is set when a branch destination address instruction is fetched, and reset when BRDR is read.

| Bit 7: DVF | Description | |
|---|---|---|
| 0 | BRDR value is invalid | (Initial value) |
| 1 | BRDR value is valid | |

See the PC Trace section for the method of executing a PC trace using the branch source register (BRSR), branch destination register (BRDR), and branch flag register (BRFR).

Bits 11 to 8 and 6 to 0—Reserved. These bits always read 0.

## 6.3 Operation

### 6.3.1 Flow of the User Break Operation

The flow from setting of break conditions to user break interrupt exception handling is described below:

1. The break addresses are set in the break address registers (BARA, BARB), the masked addresses are set in the break address mask registers (BAMRA, BAMRB), the break data is set in the break data register (BDRB), and the masked data is set in the break data mask register (BDMRB). The breaking bus conditions are set in the break bus cycle registers (BBRA, BBRB). The three groups of the BBRA and BBRB registers—CPU cycle/DMAC cycle select, instruction fetch/data access select, and read/write select— are each set. No user break interrupt will be generated if even one of these groups is set with 00. The conditions are set in the respective bits of the BRCR register.
2. When the set conditions are satisfied, the UBC sends a user break interrupt request to the interrupt controller (INTC). When conditions match, the CPU condition match flags (CMFCA, CMFCB) and DMAC condition match flags (CMFPA, CMFPB) for the respective channels are set.
3. The interrupt controller checks the user break interrupt's priority level. The user break interrupt has priority level 15, so it is accepted only if the interrupt mask level in bits I3–I0 in the status register (SR) is 14 or lower. When the I3–I0 bit level is 15, the user break interrupt cannot be accepted but it is held pending until user break interrupt exception handling can be carried out. Section 5, Interrupt Controller, describes the handling of priority levels in detail.
4. When the priority is found to permit acceptance of the user break interrupt, the CPU starts user break interrupt exception handling.
5. The appropriate condition match flag (CMFCA, CMFPA, CMFCB, CMFPB) can be used to check if the set conditions match or not. The flags are set by the matching of the conditions, but they are not reset. 0 must first be written to them before they can be used again.

   If an execution times break is specified for channel B, CMFCB or CMFPB is set when the number of executions matches the execution times setting in BETR.

### 6.3.2 Break on Instruction Fetch Cycle

1. When CPU/instruction fetch/read/word is set in the break bus cycle registers (BBRA/BBRB), the break condition becomes the CPU's instruction fetch cycle. Whether it breaks before or after the execution of the instruction can then be selected for the appropriate channel with the PCBA/PCBB bit in the break control register (BRCR).
2. An instruction set for a break before execution breaks when it is confirmed that the instruction has been fetched and will be executed. This means this feature cannot be used on instructions fetched by overrun (instructions fetched at a branch or during an interrupt transition, but not to be executed). When this kind of break is set for the delay slot of a delayed branch instruction or an instruction following an interrupt-disabled instruction, such as LDC, the interrupt is generated prior to execution of the first instruction at which the interrupt is subsequently then accepted.
3. With the post-execution condition, the instruction for which the break condition is set is executed and then the interrupt is generated before execution of the next instruction. As with pre-execution breaks, this cannot be used with overrun fetch instructions. When this kind of break is set for a delayed branch instruction or an interrupt-disabled instruction, such as LDC, the interrupt is generated before execution of the next instruction at which the interrupt can be accepted.
4. When an instruction fetch cycle is set for channel B, break data register B (BDRB) is ignored. There is thus no need to set break data for an instruction fetch cycle break.
5. When an instruction fetch cycle is set, set the address for the break as the first address at which that instruction is located. A break will not occur if any other address is set. A break will not occur if the address of the lower word of a 32-bit instruction is set.

### 6.3.3 Break on Data Access Cycle

1. The bus cycles in which CPU data access breaks occur are: memory cycles from instructions, and stacking and vector reads during exception handling. These breaks cannot be used for vector fetch cycles of external vector interrupts, or for dummy cycles in synchronous DRAM burst write or burst read access.
2. The relationship between the data access cycle address and the comparison condition for operand size are shown in table 6.2.

**Table 6.2 Data Access Cycle Addresses and Operand Size Comparison Conditions**

| Access Size | Address Compared |
|---|---|
| Longword | Break address register bits 31–2 compared with address bus bits 31–2 |
| Word | Break address register bits 31–1 compared with address bus bits 31–1 |
| Byte | Break address register bits 31–0 compared with address bus bits 31–0 |

This means that when address H'00001003 is set without specifying the size condition, for example, the bus cycle in which the break condition is satisfied is as follows (where other conditions are met):

Longword access at address H'00001000

Word access at address H'00001002

Byte access at address H'00001003

3. When the data value is included in the break conditions on channel B:

   When the data value is included in the break conditions, specify either longword, word, or byte as the operand size in the break bus cycle registers (BBRA, BBRB). When data values are included in break conditions, a break interrupt is generated when the address conditions and data conditions both match. To specify byte data for this case, set the same data in the two bytes at bits 15 to 8 and bits 7 to 0 of the break data register B (BDRB) and break data mask register B (BDMRB). When word or byte is set, bits 31 to 16 of BDRB and BDMRB are ignored.

### 6.3.4 Program Counter (PC) Values Saved

1. Break on Instruction Fetch (Before Execution): The program counter (PC) value saved to the stack in user break interrupt exception handling is the address that matches the break condition. The user break interrupt is generated before the fetched instruction is executed. If a break condition is set on an instruction that follows an interrupt-disabled instruction, however, the break occurs before execution of the instruction at which the next interrupt is accepted, so the PC value saved is the address of the break.
2. Break on Instruction Fetch (After Execution): The program counter (PC) value saved to the stack in user break interrupt exception handling is the address executed after the one that matches the break condition. The fetched instruction is executed and the user break interrupt generated before the next instruction is executed. If a break condition is set on an interrupt-disabled instruction, the break occurs before execution of the instruction at which the next interrupt is accepted, so the PC value saved is the address of the break.
3. Break on Data Access (CPU/DMAC): The address saved is the start address of the next instruction after the last instruction executed before user break exception handling started. When data access (CPU/DMAC) is set as a break condition, the place where the break will occur cannot be specified exactly. The break will occur at an instruction fetched close to where the data access to receive the break occurs.

### 6.3.5 X Memory or Y Memory Bus Cycle Breaks

Break conditions for X bus cycles or Y bus cycles can be specified only for channel B. When XYE is set to 1 in BBRB, break addresses and break data on the X memory or Y memory bus can be specified. Either the X memory bus or the Y memory bus must be selected by means of the XYS setting in BBRB. It is not possible to include both X memory and Y memory in the break conditions at the same time. The break conditions are applied to X memory bus cycles or Y memory bus cycles by setting CPU bus master, data access cycle, read or write access, and word operand size or no operand size specification in break bus cycle register B (BBRB).

When an X memory address is selected as a break condition, specify the X memory address in the upper 16 bits of BARB and BAMRB; when a Y memory address is selected, specify the Y memory address in the lower 16 bits of BARB and BAMRB. X memory data or Y memory data specification for BDRB and BDMRB is carried out in a similar way.

### 6.3.6 Sequential Breaks

When the SEQ bit is set to 1 in BRCR, a sequential break occurs when a channel B break condition is satisfied after a channel A break condition has been satisfied. A user break will not be executed if a channel B break condition is satisfied before a channel A break condition, and a sequential break will not be executed if channel A and channel B break conditions occur simultaneously.

However, if the bus cycle conditions for channel A are specified as break before instruction execution (PCBA = 0 in BRCR) and instruction fetch cycles (by BBRA), when channel A and channel B bus cycle conditions are satisfied, a break is generated and the condition match flag is set to 1 in BRCR.

In a sequential break specification, the X bus or Y bus can be selected, and an execution times break condition can also be specified. For example, when an execution times break condition is specified, the break condition is satisfied when the channel B break condition is met when BETR = H'0001 after the channel A break condition is met. Since the BETR value is decremented only by a channel B condition, a user break interrupt is issued after the occurrence of the channel B condition in the last circuit following the occurrence of the channel A condition.

### 6.3.7 PC Trace

1. A PC trace is started by setting the PC trace enable bit (PCTE) to 1 in BRCR. When a branch (branch instruction, repeat, interrupt) occurs, an address that enables the branch source address to be calculated and the branch destination address are stored in the branch source register (BRSR) and branch destination register (BRDR). The branch destination instruction fetch address is stored in BRDR, while the last instruction fetch address before the branch is stored in BRSR. The branch flag register (BRFR) holds a pointer that indicates the relationship to the instruction executed immediately before the branch.

2. The address of the instruction executed immediately before the branch can be calculated from the address stored in BRSR and the pointer stored in BRFR. If the address stored in BRSR is BSA, the pointer stored in BRFR is PID, and the address prior to the branch is IA, then IA = BSA – 2 × PID.

   With this equation, caution is required in the case where an interrupt (branch) is executed before the branch destination instruction is executed. In the example in figure 6.2, the address of instruction “Exec” executed immediately before the branch is calculated using the equation IA = BSA – 2 × PID. However, if branch “branch” has a delay slot and the branch destination is address 4n + 2, branch destination address “Dest” specified by the branch instruction is stored in BRSR. Therefore, the equation IA = BSA – 2 × PID does not apply in this case, and this PID is invalid. In this case only, BSA is at the 4n + 2 boundary, classified as shown in table 6.3.

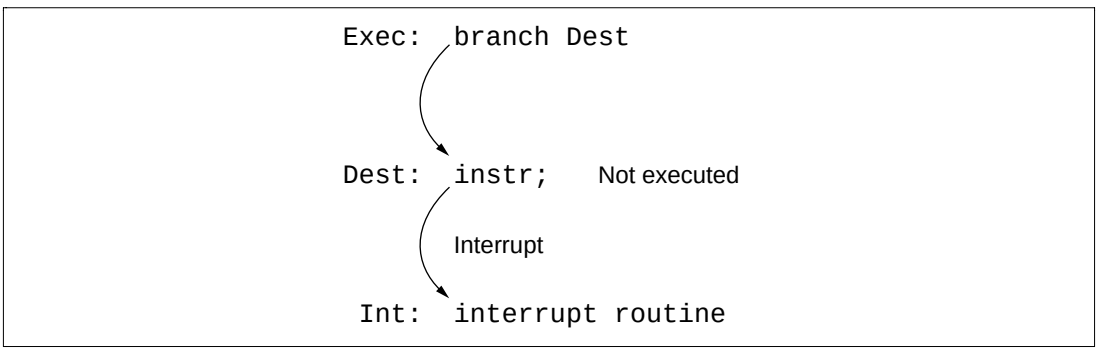



**Figure 6.2 When Interrupt Occurs before Branch Instruction is Executed**

**Table 6.3 BSA Values Stored in Exception Handling before Execution of Branch Destination Instruction**

| Branch | Branch Destination (Dest) | BSA | Branch Source Address Calculable by Means of BRSR and BRFR |
|---|---|---|---|
| Delay | 4n | 4n | Exec = IA = BSA – 2 × PID |
| | 4n + 2 | 4n + 2 | Dest = BSA |
| No delay | 4n or 4n + 2 | 4n | Exec = IA = BSA – 2 × PID |

If PID is an odd number, the value incremented by 2 indicates the instruction buffer, but the equations in the table do not take this into account. Therefore, the calculation can be performed using the values of BSA stored in BRSR and PID stored in BRFR.

3. The location indicated by IA, the address prior to the branch, depends on the type of branch.

   a. Branch instruction: Branch instruction address
   b. Repeat loop: Second-before-last instruction of the repeat loop

```
Repeat_Start: inst (1)    ;----->BRDR
              inst (2)    ;
               :
              inst (n–1) ;----->Address calculated from BRSR and BRFR
Repeat End:   inst (n)    ;
```

   c. Interrupt: Instruction executed immediately before the interrupt
      The start address of the interrupt routine is stored in BRDR.

   In a repeat loop consisting of no more than three instructions, an instruction fetch cycle is not generated. A PC trace is invalid, since the branch destination address is unknown.

4. BRSR, BRDR, and BRFR have a four-queue structure. When reading addresses stored in a PC trace, reads are performed from the head of the queue. BRFR, BRSR, and BRDR are read in that order. After BRDR is read, the queue shifts by one. BRSR and BRDR should be read by longword access.

### 6.3.8 Example of Use

**Break on a CPU Instruction Fetch Bus Cycle:**

A. Register settings: BARA = H'00000404, BAMRA = H'00000000, BBRA = H'0054
BARB = H'00008010, BAMRB = H'00000006, BBRB = H'0054
BDRB = H'00000000, BDMRB = H'00000000
BRCR = H'0400

Conditions set (channel A/channel B independent mode):
Channel A: Address = H'00000404, address mask H'00000000
Bus cycle = CPU, instruction fetch (after execution), read
(operand size not included in conditions)
Channel B: Address = H'00008010, address mask H'00000006
Data H'00000000, data mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read
(operand size not included in conditions)

A user break will occur after the instruction at address H'00000404 is executed, or a user break will be generated before the execution of the instruction at address H'00008010–H'00008016.

B. Register settings: BARA = H'00037226, BAMRA = H'00000000, BBRA = H'0056
BARB = H'0003722E, BAMRB = H'00000000, BBRB = H'0056
BDRB = H'00000000, BDMRB = H'00000000
BRCR = H'0010

Conditions set (channel A → channel B sequential mode):
Channel A: Address = H'00037226, address mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read, word
Channel B: Address = H'0003722E, address mask H'00000000
Data H'00000000, data mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read, word

The instruction at address H'00037226 will be executed and then a user break interrupt will occur before the instruction at address H'0003722E is executed.

C. Register settings: BARA = H'00027128, BAMRA = H'00000000, BBRA = H'005A
BARB = H'00031415, BAMRB = H'00000000, BBRB = H'0054
BDRB = H'00000000, BDMRB = H'00000000
BRCR = H'0000

Conditions set (channel A/channel B independent mode):

Channel A: Address = H'00027128, address mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), write , word

Channel B: Address = H'00031415, address mask H'00000000
Data H'00000000, data mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read
(operand size not included in conditions)

A user break interrupt is not generated for channel A since the instruction fetch is not a write cycle. A user break interrupt is not generated for channel B because the instruction fetch is for an odd address.

D. Register settings: BARA = H'00037226, BAMRA = H'00000000, BBRA = H'005A
BARB = H'0003722E, BAMRB = H'00000000, BBRB = H'0056
BDRB = H'00000000, BDMRB = H'00000000
BRCR = H'0010

Conditions set (channel A → channel B sequential mode):

Channel A: Address = H'00037226, address mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), write, word

Channel B: Address = H'0003722E, address mask H'00000000
Data H'00000000, data mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read, word

The break for channel A is a write cycle, so conditions are not satisfied; since the sequence conditions are not met, no user break interrupt occurs.

E. Register settings: BARA = H'00000500/BAMRA = H'00000000/BBRA = H'0057
BARB = H'00001000/BAMRB = H'00000000/BBRB = H'0057
BDRB = H'00000000/BDMRB = H'00000000
BRCR = H'0020/BETR = H'0005

Conditions set (channel A/channel B independent mode):

Channel A: Address = H'00000500, address mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read, longword

Channel B: Address = H'00001000, address mask H'00000000
Data H'00000000, data mask H'00000000
Bus cycle = CPU, instruction fetch (before execution), read, longword
Execution times setting break enabled (5 times)

For channel A, a user break interrupt is generated before execution of the instruction at address H'00000500. For channel B, a user break interrupt is generated after the instruction at address H'00001000 has been executed 4 times, and before it is executed a fifth time.

**Break on CPU Data Access Cycle:**

A. Register settings: BARA = H'00123456, BAMRA = H'00000000, BBRA = H'0064
BARB = H'000ABCDE, BAMRB = H'000000FF, BBRB = H'006A
BDRB = H'0000A512, BDMRB = H'00000000
BRCR = H'0008

Conditions set (channel A/channel B independent mode):
Channel A: Address = H'00123456, address mask H'00000000
Bus cycle = CPU, data access, read
(operand size not included in conditions)
Channel B: Address = H'000ABCDE, address mask H'000000FF
Data H'0000A512, data mask H'00000000
Bus cycle = CPU, data access, write, word

For channel A, a user break interrupt occurs when it is read as longword at address H'00123454, as word at address H'00123456 or as byte at address H'00123456. For channel B, a user break interrupt occurs when H'A512 is written as word at H'000ABC00–H'000ABCFE.

B. Register settings: BARA = H'01000000/BAMRA = H'00000000/BBRA = H'0066
BARB = H'0000F000/BAMRB = H'FFFF0000/BBRB = H'036A
BDRB = H'00004567/BDMRB = H'00000000
BRCR = H'0008

Conditions set (channel A/channel B independent mode):
Channel A: Address = H'01000000, address mask H'00000000
Bus cycle = CPU, data access, read, word
Channel B: Y address = H'0001F000, address mask H'FFFF0000
Data H'00004567, data mask H'00000000
Bus cycle = CPU, data access, write, word

For channel A, a user break interrupt occurs in a word read at address H'01000000. For channel B, a user break interrupt occurs when H'4567 is written as a word at address H'0001F000 in Y memory space. X/Y memory space addresses differ according to the mode setting. See section 7, Bus State Controller (BSC), for details.

**Break on DMAC Data Access Cycle:**

Register settings: BARA = H'00314156, BAMRA = H'00000000, BBRA = H'0094
BARB = H'00055555, BAMRB = H'00000000, BBRB = H'00A9
BDRB = H'00007878, BDMRB = H'00000F0F
BRCR = H'0008

Conditions set (channel A/channel B independent mode):
Channel A: Address = H'00314156, address mask H'00000000
Bus cycle = DMA, instruction fetch, read
(operand size not included in conditions)
Channel B: Address = H'00055555, address mask H'00000000
Data H'00007878, data mask H'00000F0F
Bus cycle = DMA, data access, write, byte

For channel A, a user break interrupt does not occur, since no instruction fetch occurs in the DMAC cycle. For channel B, a user break interrupt occurs when the DMAC writes H'7* (where * means don't care) as byte at H'00055555.

## 6.4 Usage Notes

1. Note the following regarding sequential break specifications.
   a. When set for a sequential break, conditions match when a match of channel B conditions occurs some time after the bus cycle in which a channel A match occurs. This means that the conditions will not be satisfied when set for a bus cycle in which channel A and channel B occur simultaneously.
   b. Since the CPU uses a pipeline structure, the order of the instruction fetch cycle and memory cycle is fixed, so sequential conditions will be satisfied when the respective channel conditions are met in the order the bus cycles occur.
   c. The following must be noted when the channel A bus cycle conditions are specified as pre-execution break (PCBA = 0 in BRCR) and instruction fetch (by BBRA). When the bus cycle conditions for channel A and channel B match, a break will occur and the BRCR condition match flag will be set to 1.
2. When register settings are changed, the write values usually become valid after three cycles. For on-chip memory, instruction fetches get two instructions simultaneously. If a break condition is set on the fetch of the second of these two instructions but the contents of the UBC registers are changed so as to alter the break condition immediately after the first of the two instructions is fetched, a user break interrupt will still occur before the second instruction. To ensure the timing of the change in the setting, read the register written last as a dummy. The changed settings will be valid thereafter.

3. When a user break interrupt is generated upon a match of the instruction fetch condition and the conditions match again in the UBC while the exception handling service routine is executing, the break will cause exception handling when the interrupt request mask value in SR reaches 14 or lower. When masking addresses, when setting instruction fetch and after-execution as break conditions, and when executing in steps, the UBC's exception service routine should not cause a match of addresses with the UBC.
4. When an instruction during repeat execution, including a repeat instruction, is specified as a break condition, the following points must be noted.

   When an instruction in a repeat loop is specified as a break condition:

   a. A break will not occur during execution of a repeat loop consisting of no more than three instructions.
   b. When an execution times break is set, an instruction fetch from memory will not be performed during execution of a repeat loop consisting of no more than three instructions. Consequently, the value in the execution times register (BETR) will not be decremented.
5. Do not execute a branch instruction immediately after reading a PC trace register (BRFR, BRSR, or BRDR).
6. When an execution times break is set and CPU and DMAC bus cycles are set as break conditions, BETR will only be decremented once if the CPU and DMAC conditions occur simultaneously.
7. When the emulator is used, the UBC and H-UDI are used by the emulator, and therefore operation relating to the UBC and H-UDI may differ between the emulator and the actual chip. If the UBC and H-UDI are not used by the user system, register settings should not be made.

# Section 7 Bus State Controller (BSC)

## 7.1 Overview

The bus state controller (BSC) manages the address spaces and outputs control signals to allow optimum memory accesses to the five spaces. This enables memories like DRAM, and SDRAM, and peripheral chips, to be linked directly.

### 7.1.1 Features

The BSC has the following features:

- Address space is managed as five spaces
  - Maximum linear 32 Mbytes for each of the address spaces CS0 to CS4
  - Memory type (DRAM, synchronous DRAM, burst ROM, etc.) can be specified for each space.
  - Bus width (8, 16, or 32 bits) can be selected for each space.
  - Wait state insertion can be controlled for each space.
  - Control signals are output for each space.
- Cache
  - Cache area and cache-through area can be selected by access address.
  - In cache access, in the event of a cache access miss 16 bytes are read consecutively in 4-byte units to fill the cache. Write-through mode is used for writes.
  - In cache-through access, access is performed according to access size.
- Refresh
  - Supports CAS-before-RAS refresh (auto-refresh) and self-refresh.
  - Refresh interval can be set by the refresh counter and clock selection.
  - Intensive refreshing by means of refresh count setting (1, 2, 4, 6, or 8)
- Direct interface to DRAM
  - Row/column address multiplex output.
  - Burst transfer during reads, fast page mode for consecutive accesses.
  - TP cycle generation to secure RAS precharge time.
  - EDO mode
- Direct interface to synchronous DRAM
  - Row/column address multiplex output.
  - Selection of burst read, single write mode or burst read, burst write mode
  - Bank active mode

- Bus arbitration
  - All resources are shared with the CPU, and use of the bus is granted on reception of a bus release request from off-chip.
- Refresh counter can be used as an interval timer
  - Interrupt request generation on compare match (CMI interrupt request signal).

### 7.1.2 Block Diagram

Figure 7.1 shows a block diagram of the BSC.

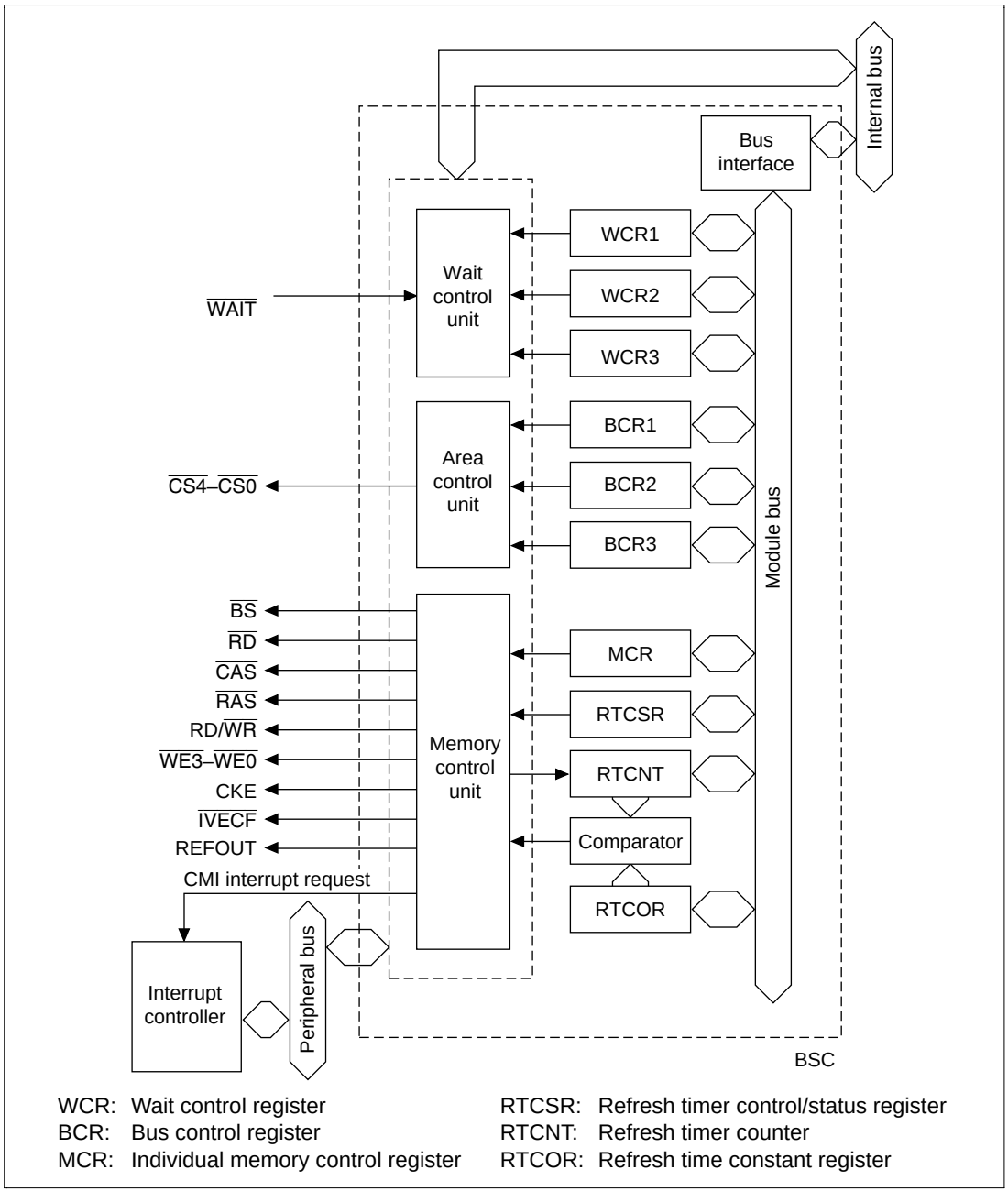



**Figure 7.1 BSC Block Diagram**

### 7.1.3 Pin Configuration

Table 7.1 shows the BSC pin configuration.

**Table 7.1 Pin Configuration**

| Signal | I/O | With Bus Released | Description |
|---|---|---|---|
| A24–A0 | O | Hi-Z | Address bus. 32 Mbytes of memory space can be specified with 25 bits |
| D31–D0 | I/O | Hi-Z | 32-bit data bus. When reading or writing a 16-bit width area, use D15–D0; when reading or writing a 8-bit width area, use D7–D0. With 8-bit accesses that read or write a 32-bit width area, input and output the data via the byte position determined by the lower address bits of the 32-bit bus |
| $\overline{BS}$ | O | Hi-Z | Indicates start of bus cycle or monitor. With the basic interface (device interfaces except for DRAM, synchronous DRAM), signal is asserted for a single clock cycle simultaneous with address output. The start of the bus cycle can be determined by this signal |
| $\overline{CS0}$–$\overline{CS4}$ | O | Hi-Z | Chip select. $\overline{CS3}$ is not asserted when the CS3 space is DRAM space |
| RD/$\overline{WR}$ | O | Hi-Z | Read/write signal. Signal that indicates access cycle direction (read/write). Connected to $\overline{WE}$ pin when DRAM/synchronous DRAM is connected |
| $\overline{RAS}$ | O | Hi-Z | $\overline{RAS}$ pin for DRAM/synchronous DRAM |
| $\overline{CAS}$, $\overline{OE}$ | O | Hi-Z | Open when using DRAM<br>Connected to $\overline{OE}$ pin when using EDO RAM<br>Connected to $\overline{CAS}$ pin when using synchronous DRAM |
| $\overline{RD}$ | O | Hi-Z | Read pulse signal (read data output enable signal). Normally, connected to the device's $\overline{OE}$ pin; when there is an external data buffer, the read cycle data can only be output when this signal is low |
| $\overline{WAIT}$ | I | Ignore | Hardware wait input |
| $\overline{BRLS}$ | I | I | Bus release request input |
| $\overline{BGR}$ | O | O | Bus grant output |
| CKE | O | O | Synchronous DRAM clock enable control. Signal for supporting synchronous DRAM self-refresh |
| $\overline{IVECF}$ | O | O | Interrupt vector fetch |
| DREQ0 | I | I | DMA request 0 |
| DACK0 | O | O | DMA acknowledge 0 |

**Table 7.1 Pin Configuration (cont)**

| Signal | I/O | With Bus Released | Description |
|---|---|---|---|
| DREQ1 | I | I | DMA request 1 |
| DACK1 | O | O | DMA acknowledge 1 |
| REFOUT | O | O | Refresh execution request output when bus is released |
| DQMUU, $\overline{WE3}$ | O | Hi-Z | When synchronous DRAM is used, connected to DQM pin for the most significant byte. For ordinary space, indicates writing to the most significant byte |
| DQMUL, $\overline{WE2}$ | O | Hi-Z | When synchronous DRAM is used, connected to DQM pin for the second byte. For ordinary space, indicates writing to the second byte |
| DQMLU, $\overline{WE1}$ | O | Hi-Z | When synchronous DRAM is used, connected to DQM pin for the third byte. For ordinary space, indicates writing to the third byte |
| DQMLL, $\overline{WE0}$ | O | Hi-Z | When synchronous DRAM is used, connected to DQM pin for the least significant byte. For ordinary space, indicates writing to the least significant byte |
| $\overline{CAS3}$ | O | Hi-Z | When DRAM is used, connected to $\overline{CAS}$ pin for the most significant byte (D31–D24) |
| $\overline{CAS2}$ | O | Hi-Z | When DRAM is used, connected to $\overline{CAS}$ pin for the second byte (D23–D16) |
| $\overline{CAS1}$ | O | Hi-Z | When DRAM is used, connected to $\overline{CAS}$ pin for the third byte (D15–D8) |
| $\overline{CAS0}$ | O | Hi-Z | When DRAM is used, connected to $\overline{CAS}$ pin for the least significant byte (D7–D0) |

Note: Hi-Z: High impedance

### 7.1.4 Register Configuration

The BSC has seven registers. These registers are used to control wait states, bus width, interfaces with memories like DRAM, synchronous DRAM, and burst ROM, and DRAM and synchronous DRAM refreshing. The register configurations are shown in table 7.2.

The size of the registers themselves is 16 bits. If read as 32 bits, the upper 16 bits are 0. *In order to prevent writing mistakes, 32-bit writes are accepted only when the value of the upper 16 bits of the write data is H'A55A; no other writes are performed.* Initialize the reserved bits.

**Initialization Procedure:** Do not access a space other than CS0 until the settings for the interface to memory are completed.

**Table 7.2 Register Configuration**

| Name | Abbr. | R/W | Initial Value | Address*[1] | Access Size |
|---|---|---|---|---|---|
| Bus control register 1 | BCR1 | R/W | H'03F0 | H'FFFFFFE0 | 16*[2], 32 |
| Bus control register 2 | BCR2 | R/W | H'00FC | H'FFFFFFE4 | 16*[2], 32 |
| Bus control register 3 | BCR3 | R/W | H'0F00 | H'FFFFFFFC | 16*[2], 32 |
| Wait control register 1 | WCR1 | R/W | H'AAFF | H'FFFFFFE8 | 16*[2], 32 |
| Wait control register 2 | WCR2 | R/W | H'000B | H'FFFFFFC0 | 16*[2], 32 |
| Wait control register 3 | WCR3 | R/W | H'0000 | H'FFFFFFC4 | 16*[2], 32 |
| Individual memory control register | MCR | R/W | H'0000 | H'FFFFFFEC | 16*[2], 32 |
| Refresh timer control/status register | RTCSR | R/W | H'0000 | H'FFFFFFF0 | 16*[2], 32 |
| Refresh timer counter | RTCNT | R/W | H'0000 | H'FFFFFFF4 | 16*[2], 32 |
| Refresh time constant register | RTCOR | R/W | H'0000 | H'FFFFFFF8 | 16*[2], 32 |

Notes: 1. This address is for 32-bit accesses; for 16-bit accesses add 2.
2. 16-bit access is for read only.

### 7.1.5 Address Map

The address map, which has a memory space of 320 Mbytes, is divided into five spaces. The types and data width of devices that can be connected are specified for each space. The overall space address map is shown in table 7.3. Since the spaces of the cache area and the cache-through area are actually the same, and the maximum memory space that can be connected is 160 Mbytes. This means that when address H'20000000 is accessed in a program, the data accessed is actually in H'00000000.

The chip has 16-kbyte RAM as on-chip memory. The on-chip RAM is divided into an X area and a Y area, which can be accessed in parallel with the DSP instruction. See the *SH-1/SH-2/SH-DSP Programming Manual* for more information.

In a reset, the CPU reads the initial values of the program counter (PC) and the stack pointer (SP) from the reset vector area, and set the values. In a power-on reset, the PC address is set H'00000000 and the SP address is set H'00000004; in a manual reset, the PC address is set H'00000008 and the SP address is set H'0000000C.

There are several spaces for cache control. These include the associative purge space for cache purges, address array read/write space for reading and writing addresses (address tags), and data array read/write space for forced reads and writes of data arrays.

**Table 7.3 Address Map**

| Address | Space | Memory | Size |
|---|---|---|---|
| H'00000000–H'01FFFFFF | CS0 space, cache area | Ordinary space or burst ROM | 32 Mbytes |
| H'02000000–H'03FFFFFF | CS1 space, cache area | Ordinary space | 32 Mbytes |
| H'04000000–H'05FFFFFF | CS2 space, cache area | Ordinary space or synchronous DRAM*[1] | 32 Mbytes |
| H'06000000–H'07FFFFFF | CS3 space, cache area | Ordinary space, synchronous DRAM, or DRAM | 32 Mbytes |
| H'08000000–H'09FFFFFF | CS4 space, cache area | Ordinary space (I/O device) | 32 Mbytes |
| H'0A000000–H'0FFFFFFF | Reserved | | |
| H'10000000–H'1000DFFF | Reserved | | |
| H'1000E000–H'1000FFFF | On-chip X RAM area | | 8 kbytes |
| H'10010000–H'1001DFFF | Reserved | | |
| H'1001E000–H'1001FFFF | On-chip Y RAM area | | 8 kbytes |
| H'10020000–H'1FFFFFFF | Reserved | | |
| H'20000000–H'21FFFFFF | CS0 space, cache-through area | Ordinary space or burst ROM | 32 Mbytes |
| H'22000000–H'23FFFFFF | CS1 space, cache-through area | Ordinary space | 32 Mbytes |
| H'24000000–H'25FFFFFF | CS2 space, cache-through area | Ordinary space or synchronous DRAM | 32 Mbytes |
| H'26000000–H'27FFFFFF | CS3 space, cache-through area | Ordinary space, synchronous DRAM*[2], or DRAM | 32 Mbytes |
| H'28000000–H'29FFFFFF | CS4 space, cache-through area | Ordinary space (I/O device) | 32 Mbytes |
| H'2A000000–H'3FFFFFFF | Reserved | | |
| H'40000000–H'49FFFFFF | Associative purge space | | 160 Mbytes |
| H'4A000000–H'5FFFFFFF | Reserved | | |
| H'60000000–H'7FFFFFFF | Address array, read/write space | | 512 Mbytes |
| H'80000000–H'BFFFFFFF | Reserved | | |
| H'C0000000–H'C0000FFF | Data array, read/write space | | 4 kbytes |

**Table 7.3 Address Map (cont)**

| Address | Space | Memory | Size |
|---|---|---|---|
| H'C0001000–H'DFFFFFFF | Reserved | | |
| H'E0000000–H'FFFEFFFF | Reserved | | |
| H'FFFF0000–H'FFFF0FFF | For setting synchronous DRAM mode | | 4 kbytes |
| H'FFFF1000–H'FFFF7FFF | Reserved | | |
| H'FFFF8000–H'FFFF8FFF | For setting synchronous DRAM mode | | 16 kbytes |
| H'FFFF9FFF–H'FFFFBFFF | | | |
| H'FFFFC000–H'FFFFFBFF | Reserved | | |
| H'FFFFFC00–H'FFFFFFFF | On-chip peripheral modules | | |

Notes: 1. Do not access reserved spaces, as operation cannot be guaranteed.
2. Bank-active mode is supported for CS3 space synchronous DRAM access.

## 7.2 Register Descriptions

### 7.2.1 Bus Control Register 1 (BCR1)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | A4LW1 | A4LW0 | A2EN DIAN | BST ROM | — | AHLW1 | AHLW0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R/W | R/W | R/W | R/W | R | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | A1LW1 | A1LW0 | A0LW1 | A0LW0 | A4EN DIAN | DRAM2 | DRAM1 | DRAM0 |
| Initial value: | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Initialize the ENDIAN, BSTROM, PSHR, and DRAM2–DRAM0 bits after a power-on reset, and do not change their values thereafter. To change other bits by writing to them, write the same value as they are initialized to. Do not access any space other than CS0 until the register initialization ends.

Bit 15—Reserved: This bit always reads 0. The write value should always be 0.

Bits 14 and 13—Long Wait Specification for Area 4 (A4LW1, A4LW0): From 3 to 14 wait cycles are inserted in CS4 space accesses when the bits that specify the wait in the wait control register specify long wait (i.e., are set to 11).

Bit 12—Endian Specification for Area 2 (A2ENDIAN): In big-endian format, the MSB of byte data is the lowest byte address and byte data goes in order toward the LSB. For little-endian format, the LSB of byte data is the lowest byte address and byte data goes in order toward the MSB. When this bit is 1, the data is rearranged into little-endian format before transfer when the CS2 space is read or written to. It is used when handling data with little-endian processors.

| Bit 12: A2ENDIAN | Description | |
|---|---|---|
| 0 | Big-endian | (Initial value) |
| 1 | Little-endian | |

Bit 11—Area 0 Burst ROM Enable (BSTROM)

| Bit 11: BSTROM | Description | |
|---|---|---|
| 0 | Area 0 is accessed normally | (Initial value) |
| 1 | Area 0 is accessed as burst ROM | |

Bit 10—Reserved: This bit always reads 0. The write value should always be 0.

Bits 9 and 8—Long Wait Specification for Areas 2 and 3 (AHLW1, AHLW0): When the basic memory interface setting is made for CS2 and CS3, from 3 to 14 wait cycles are inserted in CS2 or CS3 accesses when the bits specifying the respective area waits in the wait control register (W21/W20 or W31/W30) are set as long waits (i.e., are set to 11) (see table 7.4).

Bits 7 and 6—Long Wait Specification for Area 1 (A1LW1, A1LW0): When the basic memory interface setting is made for CS1, from 3 to 14 wait cycles are inserted in CS1 accesses when the bits specifying the wait in the wait control register are set as long wait (i.e., are set to 11) (see table 7.4).

Bits 5 and 4—Long Wait Specification for Area 0 (A0LW1, A0LW0): When the basic memory interface setting is made for CS0, from 3 to 14 wait cycles are inserted in CS0 accesses when the bits specifying the wait in the wait control register are set as long wait (i.e., are set to 11) (see table 7.4).

Bit 3—Endian Specification for Area 4 (A4ENDIAN): In big-endian mode, the most significant byte (MSB) is the lowest byte address, and byte data is aligned in order toward the least significant byte (LSB). In little-endian mode, the LSB is the lowest byte address, and byte data is aligned in order toward the MSB. When this bit is set to 1, data in read/write accesses to the CS4 space is rearranged into little endian order before being transferred. This is used for data exchange with a little-endian processor.

| Bit 3: A4ENDIAN | Description | |
|---|---|---|
| 0 | Big endian | (Initial value) |
| 1 | Little endian | |

Bits 2 to 0—Enable for DRAM and Other Memory (DRAM2–DRAM0)

| DRAM2 | DRAM1 | DRAM0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | CS2 and CS3 are ordinary spaces | (Initial value) |
| | | 1 | CS2 is ordinary space; CS3 is synchronous DRAM space | |
| | 1 | 0 | CS2 is ordinary space; CS3 is DRAM space | |
| | | 1 | Reserved (do not set) | |
| 1 | 0 | 0 | CS2 is synchronous DRAM space, CS3 is ordinary space | |
| | | 1 | CS2 and CS3 are synchronous DRAM spaces | |
| | 1 | 0 | Reserved (do not set) | |
| | | 1 | Reserved (do not set) | |

**Table 7.4 Wait Values Corresponding to BCR1 and BCR3 Register Settings (All Spaces)**

| BCR3 | BCR1 | | |
|---|---|---|---|
| AnLW2 | AnLW1 | AnLW0 | Wait Value |
| 0 | 0 | 0 | 3 cycles inserted |
| | | 1 | 4 cycles inserted |
| | 1 | 0 | 5 cycles inserted |
| | | 1 | 6 cycles inserted |
| 1 | 0 | 0 | 8 cycles inserted |
| | | 1 | 10 cycles inserted |
| | 1 | 0 | 12 cycles inserted |
| | | 1 | 14 cycles inserted (Initial value) |

Note: n: Area number (0 to 4)
AHLW2, AHLW1, and AHLW0 are common to CS2 and CS3.

### 7.2.2 Bus Control Register 2 (BCR2)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | A4SZ1 | A4SZ0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | A3SZ1 | A3SZ0 | A2SZ1 | A2SZ0 | A1SZ1 | A1SZ0 | — | — |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

Initialize BCR2 after a power-on reset and do not write to it thereafter. When writing to it, write the same values as those the bits are initialized to. Do not access any space other than CS0 until the register initialization ends.

Bits 15 to 10—Reserved: These bits always read 0. The write value should always be 0.

Bits 9 and 8—Bus Size Specification for Area 4 (CS4) (A4SZ1, A4SZ0)

| Bit 9: A4SZ1 | Bit 8: A4SZ0 | Description | |
|---|---|---|---|
| 0 | 0 | Longword (32-bit) size | (Initial value) |
| | 1 | Byte (8-bit) size | |
| 1 | 0 | Word (16-bit) size | |
| | 1 | Longword (32-bit) size | |

Bits 7 and 6—Bus Size Specification for Area 3 (CS3) (A3SZ1, A3SZ0). Effective only when ordinary space is set.

| Bit 7: A3SZ1 | Bit 6: A3SZ0 | Description | |
|---|---|---|---|
| 0 | 0 | Reserved (do not set) | |
| | 1 | Byte (8-bit) size | |
| 1 | 0 | Word (16-bit) size | |
| | 1 | Longword (32-bit) size | (Initial value) |

Bits 5 and 4—Bus Size Specification for Area 2 (CS2) (A2SZ1, A2SZ0): Effective only when ordinary space is set.

| Bit 5: A2SZ1 | Bit 4: A2SZ0 | Description | |
|---|---|---|---|
| 0 | 0 | Reserved (do not set) | |
| | 1 | Byte (8-bit) size | |
| 1 | 0 | Word (16-bit) size | |
| | 1 | Longword (32-bit) size | (Initial value) |

Bits 3 and 2—Bus Size Specification for Area 1 (CS1) (A1SZ1, A1SZ0)

| Bit 3: A1SZ1 | Bit 2: A1SZ0 | Description | |
|---|---|---|---|
| 0 | 0 | Reserved (do not set) | |
| | 1 | Byte (8-bit) size | |
| 1 | 0 | Word (16-bit) size | |
| | 1 | Longword (32-bit) size | (Initial value) |

Bits 1 and 0—Reserved: These bits always read 0. The write value should always be 0.

### 7.2.3 Bus Control Register 3 (BCR3)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | A4LW2 | AHLW2 | A1LW2 | A0LW2 |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | DSWW1 | DSWW0 | — | — | — | BASEL | EDO | BWE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R | R/W | R/W | R/W |

Initialize the BASEL, EDO, and BWE bits after a power-on reset and do not write to them thereafter. To change other bits by writing to them, write the same value as they are initialized to. Do not access any space other than CS0 until the register initialization ends.

Bits 15 to 12—Reserved bits: These bits always read 0. The write value should always be 0.

Bits 11 to 8—Long Wait Specification for Areas 0 to 4 (AnLW2): When the basic memory interface setting is made for CS n, from 3 to 14 wait cycles are inserted in CS n accesses,

according to the combination with the long wait specification bits (AnLW1 and AnLW0) in BCR1, when the bits specifying the wait in the wait control register are set as long wait (i.e., are set to 11). For a basic description of long waits, see section 7.2.1, Bus Control Register 1 (BCR1).

Bits 7 and 6—DMA Single-Write Wait (DSWW1, DSWW0): These bits determine the number of wait states inserted between DACK assertion and CASn assertion when writing to DRAM or EDO RAM in DMA single address mode.

| Bit 7: DSWW1 | Bit 6: DSWW0 | Description | |
|---|---|---|---|
| 0 | 0 | 0 waits | (Initial value) |
| | 1 | 1 wait | |
| 1 | 0 | 2 waits | |
| | 1 | Reserved (do not set) | |

Bits 5 to 3—Reserved bits: These bits always read 0. The write value should always be 0.

Bit 2—Number of Banks Specification when Using 64M Synchronous DRAM (BASEL): When 64M synchronous DRAM is specified by AMX2–AMX0 in MCR, the number of banks can be specified.

| Bit 2: BASEL | Description | |
|---|---|---|
| 0 | 4 banks | (Initial value) |
| 1 | 2 banks | |

Bit 1—EDO Mode Specification (EDO): Enables EDO mode to be specified when DRAM is specified for CS3 space.

| Bit 1: EDO | Description | |
|---|---|---|
| 0 | High-speed page mode | (Initial value) |
| 1 | EDO mode | |

Bit 0—Synchronous DRAM Burst Write Specification (BWE): Enables burst write mode to be specified when synchronous DRAM is specified for CS2 or CS3 space.

| Bit 0: BWE | Description | |
|---|---|---|
| 0 | Single write mode | (Initial value) |
| 1 | Burst write mode | |

### 7.2.4 Wait Control Register 1 (WCR1)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IW31 | IW30 | IW21 | IW20 | IW11 | IW10 | IW01 | IW00 |
| Initial value: | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | W31 | W30 | W21 | W20 | W11 | W10 | W01 | W00 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Do not access a space other than CS0 until the settings for register initialization are completed.

Bits 15 to 8—Idles between Cycles for Areas 3 to 0 (IW31–IW00): These bits specify idle cycles inserted between consecutive accesses to different CS spaces. Idles are used to prevent data conflict between ROM or the like, which is slow to turn the read buffer off, and fast memories and I/O interfaces. Even when access is to the same space, idle cycles must be inserted when a read access is followed immediately by a write access. The idle cycles to be inserted comply with the specification for the previously accessed space.

| IW31, IW21, IW11, IW01 | IW30, IW20, IW10, IW00 | Description |
|---|---|---|
| 0 | 0 | No idle cycle |
| | 1 | One idle cycle inserted |
| 1 | 0 | Two idle cycles inserted (Initial value) |
| | 1 | Four idle cycles inserted |

Bits 7 to 0—Wait Control for Areas 3 to 0 (W31–W00)

- When the CSn space is set as ordinary space, the number of CSn space waits can be specified with Wn1 and Wn0.

| W31, W21, W11, W01 | W30, W20, W10, W00 | Description |
|---|---|---|
| 0 | 0 | External wait input disabled without wait |
| | 1 | External wait input enabled with one wait |
| 1 | 0 | External wait input enabled with two waits |
| | 1 | Complies with the long wait specification of bus control register 1, 3 (BCR1, BCR3). External wait input is enabled (Initial value) |

- When CS3 is DRAM, the number of CAS assert cycles is specified by wait control bits W31 and W30

| Bit 7: W31 | Bit 6: W30 | Description |
|---|---|---|
| 0 | 0 | 1 cycle |
| | 1 | 2 cycles |
| 1 | 0 | 3 cycles |
| | 1 | Reserved (do not set) |

When external wait mask bit A3WM in WCR2 is 0 and the number of CAS assert cycles is set to 2 or more, external wait input is enabled.

- When CS2 or CS3 is synchronous DRAM, CAS latency is specified by wait control bits W31 and W30, and W21 and W20, respectively

| W31, W21 | W30, W20 | Description |
|---|---|---|
| 0 | 0 | 1 cycle |
| | 1 | 2 cycles |
| 1 | 0 | 3 cycles |
| | 1 | 4 cycles (Initial value) |

With synchronous DRAM, external wait input is ignored regardless of any setting.

### 7.2.5 Wait Control Register 2 (WCR2)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | A4WD1 | A4WD0 | — | A4WM | A3WM | A2WM | A1WM | A0WM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | IW41 | IW40 | W41 | W40 |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W |

Bits 15 and 14—Number of External Waits Specification for Area 4 (A4WD1, A4WD0): These bits specify the number of cycles between acceptance of CS4 space external wait negation and $\overline{RD}$ or $\overline{WEn}$ negation.

| Bit 15: A4WD1 | Bit 14: A4WD0 | Description | |
|---|---|---|---|
| 0 | 0 | 1 cycle | (Initial value) |
| | 1 | 2 cycles | |
| 1 | 0 | 4 cycles | |
| | 1 | Reserved (do not set) | |

Bit 13—Reserved bit. This bit always reads 0. The write value should always be 0.

Bits 12 to 8—External Wait Mask Specification for Areas 0 to 4 (A4WM–A0WM): These bits enable waits to be masked for CS spaces 0 to 4. When a value other than 00 is set in the wait control bits for CS spaces 0 to 4 (W41–W00), external wait input can be enabled, but the wait input can be masked by setting these bits to 1. With synchronous DRAM, external wait input is ignored regardless of the settings.

| A4WM<br>A3WM<br>A2WM<br>A1WM<br>A0WM | W41<br>W31<br>W21<br>W11<br>W01 | W40<br>W30<br>W20<br>W10<br>W00 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | External wait input ignored | |
| | | 1 | External wait input enabled | |
| | 1 | 0 | External wait input enabled | |
| | | 1 | External wait input enabled | (Initial value) |
| 1 | No effect | No effect | External wait input ignored | |

Bits 7 to 4—Reserved bits: These bits always read 0. The write value should always be 0.

Bits 3 and 2—Idles between Cycles for Area 4 (IW41, IW40): These bits specify idle cycles inserted between cycles in CS4 in the same way as for CS 0 to 3.

| Bit 3: IW41 | Bit 2: IW40 | Description | |
|---|---|---|---|
| 0 | 0 | No idle cycle | |
| | 1 | One idle cycle inserted | |
| 1 | 0 | Two idle cycles inserted | (Initial value) |
| | 1 | Four idle cycles inserted | |

Bits 1 and 0—Wait Control for Area 4 (W41, W40): These bits specify waits for CS4 in the same way as for areas 0 to 3.

| Bit 1: W41 | Bit 0: W40 | Description |
|---|---|---|
| 0 | 0 | External wait input disabled without wait |
| | 1 | External wait input enabled with one wait |
| 1 | 0 | External wait input enabled with two waits |
| | 1 | Complies with the long wait specification of bus control registers 1 and 3 (BCR1, BCR3). External wait input is enabled (Initial value) |

### 7.2.6 Wait Control Register 3 (WCR3)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | A4SW2 | A4SW1 | A4SW0 | — | A4HW1 | A4HW0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | A3SHW1 | A3SHW0 | A2SHW1 | A2SHW0 | A1SHW1 | A1SHW0 | A0SHW1 | A0SHW0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 14—Reserved bits: These bits always read 0. The write value should always be 0.

Bits 13 to 11—CS4 Address/$\overline{\text{CS4}}$ to $\overline{\text{RD}}$/$\overline{\text{WEn}}$ Assertion (A4SW2–A4SW0): These bits specify the number of cycles from address/$\overline{\text{CS4}}$ output to $\overline{\text{RD}}$/$\overline{\text{WEn}}$ assertion for the CS4 space.

| Bit 13: A4SW2 | Bit 12: A4SW1 | Bit 11: A4SW0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | 0.5 cycles (Initial value) |
| | | 1 | 1.5 cycle |
| | 1 | 0 | 3.5 cycles |
| | | 1 | 5.5 cycles |
| 1 | 0 | 0 | 7.5 cycles |
| | | 1 | Reserved (do not set) |
| | 1 | 0 | Reserved (do not set) |
| | | 1 | Reserved (do not set) |

Bit 10—Reserved bit: This bit always reads 0. The write value should always be 0.

Bits 9 and 8—Area 4 $\overline{RD}/\overline{WEn}$ Negation to Address/$\overline{CS4}$ Hold (A4HW1, A4HW0): These bits specify the number of cycles from $\overline{RD}/\overline{WEn}$ negation to address/$\overline{CS4}$ hold for the CS4 space.

| Bit 9: A4HW1 | Bit 8: A4HW0 | Description | |
|---|---|---|---|
| 0 | 0 | 0.5 cycle, $\overline{CS4}$ hold cycle = 0 cycles | (Initial value) |
| | 1 | 1.5 cycle, $\overline{CS4}$ hold cycle = 1 cycle | |
| 1 | 0 | 3.5 cycle, $\overline{CS4}$ hold cycle = 3 cycles | |
| | 1 | 5.5 cycle, $\overline{CS4}$ hold cycle = 5 cycles | |

Bits 7 to 0—Area 3 to 0 $\overline{CSn}$ Assert Period Extension (A3SHW1–A0SHW0): These bits specify the number of cycles from address/$\overline{CSn}$ output to $\overline{RD}/\overline{WEn}$ assertion and from $\overline{RD}/\overline{WEn}$ negation to address/$\overline{CSn}$ hold for areas 3 to 0.

| A3SHW1<br>A2SHW1<br>A1SHW1<br>A0SHW1 | A3SHW0<br>A2SHW0<br>A1SHW0<br>A0SHW0 | Description | |
|---|---|---|---|
| 0 | 0 | 0.5 cycle, $\overline{CSn}$ hold cycle = 0 cycles | (Initial value) |
| | 1 | 1.5 cycle, $\overline{CSn}$ hold cycle = 1 cycle | |
| 1 | 0 | 2.5 cycle, $\overline{CSn}$ hold cycle = 2 cycles | |
| | 1 | Reserved (do not set) | |

Note: n = 0 to 3

### 7.2.7 Individual Memory Control Register (MCR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TRP0 | RCD0 | TRWL0 | TRAS1 | TRAS0 | BE | RASD | TRWL1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | AMX2 | SZ | AMX1 | AMX0 | RFSH | RMODE | TRP1 | RCD1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The TRP1–TRP0, RCD1–RCD0, TRWL1–TRWL0, TRAS1–TRAS0, BE, RASD, AMX2–AMX0 and SZ bits are initialized after a power-on reset. Do not write to them thereafter. When writing to

them, write the same values as they are initialized to. Do not access CS2 or CS3 until register initialization is completed.

Bits 1 and 15—RAS Precharge Time (TRP1, TRP0): When DRAM is connected, specifies the minimum number of cycles after $\overline{\text{RAS}}$ is negated before the next assert. When synchronous DRAM is connected, specifies the minimum number of cycles after precharge until a bank active command is output. See section 7.5, Synchronous DRAM Interface, for details.

| Bit 1: TRP1 | Bit 15: TRP0 | Description | |
|---|---|---|---|
| 0 | 0 | 1 cycle | (Initial value) |
| | 1 | 2 cycles | |
| 1 | 0 | 3 cycles | |
| | 1 | 4 cycles | |

Bits 0 and 14—RAS-CAS Delay (RCD1, RCD0): When DRAM is connected, specifies the number of cycles after $\overline{\text{RAS}}$ is asserted before $\overline{\text{CAS}}$ is asserted. When synchronous DRAM is connected, specifies the number of cycles after a bank active (ACTV) command is issued until a read or write command (READ, READA, WRIT, WRITA) is issued.

| Bit 0: RCD1 | Bit 14: RCD0 | Description | |
|---|---|---|---|
| 0 | 0 | 1 cycle | (Initial value) |
| | 1 | 2 cycles | |
| 1 | 0 | 3 cycles | |
| | 1 | Reserved (do not set) | |

Bits 8 and 13—Write-Precharge Delay (TRWL1, TRWL0): When the synchronous DRAM is not in the bank active mode, this bit specifies the number of cycles after the write cycle before the start-up of the auto-precharge. Based on this number of cycles, the timing at which the next active command can be issued is calculated within the bus controller. In bank active mode, this bit specifies the number of cycles before the precharge command is issued after the write command is issued. This bit is ignored when memory other than synchronous DRAM is connected.

| Bit 8: TRWL1 | Bit 13: TRWL0 | Description | |
|---|---|---|---|
| 0 | 0 | 1 cycle | (Initial value) |
| | 1 | 2 cycles | |
| 1 | 0 | 3 cycles | |
| | 1 | Reserved (do not set) | |

Bits 12 and 11—CAS-Before-RAS Refresh RAS Assert Time (TRAS1, TRAS0): These bits specify the RAS assertion width when DRAM is connected.

| Bit 12: TRAS1 | Bit 11: TRAS0 | Description | |
|---|---|---|---|
| 0 | 0 | 2 cycles | (Initial value) |
| | 1 | 3 cycles | |
| 1 | 0 | 4 cycles | |
| | 1 | 5 cycles | |

After an auto-refresh command is issued, a bank active command is not issued for TRAS cycles, regardless of the TRP bit setting. For synchronous DRAM, there is no $\overline{\text{RAS}}$ assertion period, but there is a limit for the time from the issue of a refresh command until the next access. This value is set to observe this limit. Commands are not issued for TRAS cycles when self-refresh is cleared.

| Bit 12: TRAS1 | Bit 11: TRAS0 | Description | |
|---|---|---|---|
| 0 | 0 | 3 cycles | (Initial value) |
| | 1 | 4 cycles | |
| 1 | 0 | 6 cycles | |
| | 1 | 9 cycles | |

Bit 10—Burst Enable (BE)

| Bit 10: BE | Description | |
|---|---|---|
| 0 | Burst disabled | (Initial value) |
| 1 | High-speed page mode during DRAM and ED0 interfacing is enabled.<br>Burst access conditions are as follows:<br>• Longword access, cache fill access, or DMAC 16-byte transfer, with 16-bit bus width<br>• Cache fill access or DMAC 16-byte transfer, with 32-bit bus width<br>During synchronous DRAM access, burst operation is always enabled regardless of this bit | |

Bit 9—Bank Active Mode (RASD)

| Bit 9: RASD | Description |
|---|---|
| 0 | For DRAM, RAS is negated after access ends (normal operation)<br>For synchronous DRAM, a read or write is performed using auto-precharge mode. The next access always starts with a bank active command<br>(Initial value) |
| 1 | For DRAM, after access ends RAS down mode is entered in which RAS is left asserted. When using this mode with an external device connected which performs writes other than to DRAM, see section 7.6.5, Burst Access<br>For synchronous DRAM, access ends in the bank active state. This is only valid for area 3. When area 2 is synchronous DRAM, the mode is always auto-precharge |

Bits 7, 5, and 4—Address Multiplex (AMX2–AMX0)

- For DRAM interface

| Bit 7: AMX2 | Bit 5: AMX1 | Bit 4: AMX0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | 8-bit column address DRAM |
| | | 1 | 9-bit column address DRAM |
| | 1 | 0 | 10-bit column address DRAM |
| | | 1 | 11-bit column address DRAM |
| 1 | 0 | 0 | Reserved (do not set) |
| | | 1 | Reserved (do not set) |
| | 1 | 0 | Reserved (do not set) |
| | | 1 | Reserved (do not set) |

- For synchronous DRAM interface

| Bit 7: AMX2 | Bit 5: AMX1 | Bit 4: AMX0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | 16-Mbit DRAM (1M × 16 bits) |
| | | 1 | 16-Mbit DRAM (2M × 8 bits)*[1] |
| | 1 | 0 | 16-Mbit DRAM (4M × 4 bits)*[1] |
| | | 1 | 4-Mbit DRAM (256k × 16 bits) |
| 1 | 0 | 0 | 64-Mbit DRAM (4M × 16 bits)*[2] |
| | | 1 | 64-Mbit DRAM (8M × 8 bits)*[1] |
| | 1 | 0 | Reserved (do not set) |
| | | 1 | 2-Mbit DRAM (128k × 16 bits) |

Notes: 1. Reserved. Do not set when SZ bit in MCR is 0 (16-bit bus width).
2. With the E800 emulator and debug chip, this setting is reserved and must not be set when the SZ bit in MCR is 0 (16-bit bus width).

Bit 6—Memory Data Size (SZ): For synchronous DRAM and DRAM space, the data bus width of BCR2 is ignored in favor of the specification of this bit.

| Bit 6: SZ | Description | |
|---|---|---|
| 0 | Word (16 bits) | (Initial value) |
| 1 | Longword (32 bits) | |

Bit 3—Refresh Control (RFSH): This bit determines whether or not the refresh operation of DRAM/synchronous DRAM is performed.

| Bit 3: RFSH | Description | |
|---|---|---|
| 0 | No refresh | (Initial value) |
| 1 | Refresh | |

Bit 2—Refresh Mode (RMODE): When the RFSH bit is 1, this bit selects normal refresh or self-refresh. When the RFSH bit is 0, do not set this bit to 1. When the RFSH bit is 1, self-refresh mode is entered immediately after the RMODE bit is set to 1. When the RFSH bit is 1 and this bit is 0, a CAS-before-RAS refresh or auto-refresh is performed at the interval set in the 8-bit interval timer. When a refresh request occurs during an external area access, the refresh is performed after the access cycle is completed. When set for self-refresh, self-refresh mode is entered immediately unless the chip is in the middle of a synchronous DRAM area access, in which case self-refresh mode is entered when the access ends. Refresh requests from the interval timer are ignored during self-refresh.

| Bit 2: RMODE | Description | |
|---|---|---|
| 0 | Normal refresh | (Initial value) |
| 1 | Self-refresh | |

### 7.2.8 Refresh Timer Control/Status Register (RTCSR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CMF | CMIE | CKS2 | CKS1 | CKS0 | RRC2 | RRC1 | RRC0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 to 8—Reserved: These bits always read 0. The write value should always be 0.

Bit 7—Compare Match Flag (CMF): This status flag, which indicates that the values of RTCNT and RTCOR match, is set/cleared under the following conditions:

| Bit 7: CMF | Description |
|---|---|
| 0 | RTCNT and RTCOR match<br>Clear condition: After RTCSR is read when CMF is 1, 0 is written in CMF |
| 1 | RTCNT and RTCOR do not match<br>Set condition: RTCNT = RTCOR |

Bit 6—Compare Match Interrupt Enable (CMIE): Enables or disables an interrupt request caused by the CMF bit of RTCSR when CMF is set to 1.

| Bit 6: CMIE | Description | |
|---|---|---|
| 0 | Interrupt request caused by CMF is disabled | (Initial value) |
| 1 | Interrupt request caused by CMF is enabled | |

Bits 5 to 3—Clock Select Bits (CKS2–CKS0)

| Bit 5: CKS2 | Bit 4: CKS1 | Bit 3: CKS0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | Count-up disabled | (Initial value) |
| | | 1 | Pϕ/4 | |
| | 1 | 0 | Pϕ/16 | |
| | | 1 | Pϕ/64 | |
| 1 | 0 | 0 | Pϕ/256 | |
| | | 1 | Pϕ/1024 | |
| | 1 | 0 | Pϕ/2048 | |
| | | 1 | Pϕ/4096 | |

Bits 2 to 0—Refresh Count (RRC2–RRC0): These bits specify the number of consecutive refreshes to be performed when the refresh timer counter (RTCNT) and refresh time constant register (RTCOR) values match and a refresh request is issued.

| Bit 2: RRC2 | Bit 1: RRC1 | Bit 0: RRC0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 refresh | (Initial value) |
| | | 1 | 2 refreshes | |
| | 1 | 0 | 4 refreshes | |
| | | 1 | 6 refreshes | |
| 1 | 0 | 0 | 8 refreshes | |
| | | 1 | Reserved (do not set) | |
| | 1 | 0 | Reserved (do not set) | |
| | | 1 | Reserved (do not set) | |

### 7.2.9 Refresh Timer Counter (RTCNT)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The 8-bit counter RTCNT counts up with input clocks. The clock select bit of RTCSR selects an input clock. RTCNT values can always be read/written by the CPU. When RTCNT matches RTCOR, RTCNT is cleared. Returns to 0 after it counts up to 255.

Bits 15 to 8—Reserved: These bits always read 0. The write value should always be 0.

### 7.2.10 Refresh Time Constant Register (RTCOR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

RTCOR is an 8-bit read/write register. The values of RTCOR and RTCNT are constantly compared. When the values correspond, the compare match flag (CMF) in RTCSR is set and RTCNT is cleared to 0. When the refresh bit (RFSH) in the individual memory control register (MCR) is set to 1, a refresh request signal occurs. The refresh request signal is held until refresh operation is actually performed. If the refresh request is not processed before the next match, the previous request becomes ineffective.

When the CMIE bit in RTCSR is set to 1, an interrupt request is sent to the controller by this match signal. The interrupt request is output continuously until the CMF bit in RTCSR is cleared. When the CMF bit clears, it only affects the interrupt; the refresh request is not cleared by this operation. When a refresh is performed and refresh requests are counted using interrupts, a refresh can be set simultaneously with the interval timer interrupt.

Bits 15 to 8—Reserved: These bits always read 0. The write value should always be 0.

## 7.3 Access Size and Data Alignment

### 7.3.1 Connection to Ordinary Devices

Byte, word, and longword are supported as access units. Data is aligned based on the data width of the device. Therefore, reading longword data from a byte-width device requires four read operations. The bus state controller automatically converts data alignment and data length between interfaces. An 8-bit, 16-bit, or 32-bit external device data width can be connected by using the mode pins for the CS0 space, or by setting BCR2 for the CS1–CS4 spaces. However, the data width of devices connected to the respective spaces is specified statically, and the data width cannot be changed for each access cycle. Figures 7.2 to 7.4 show the relationship between device data widths and access units.

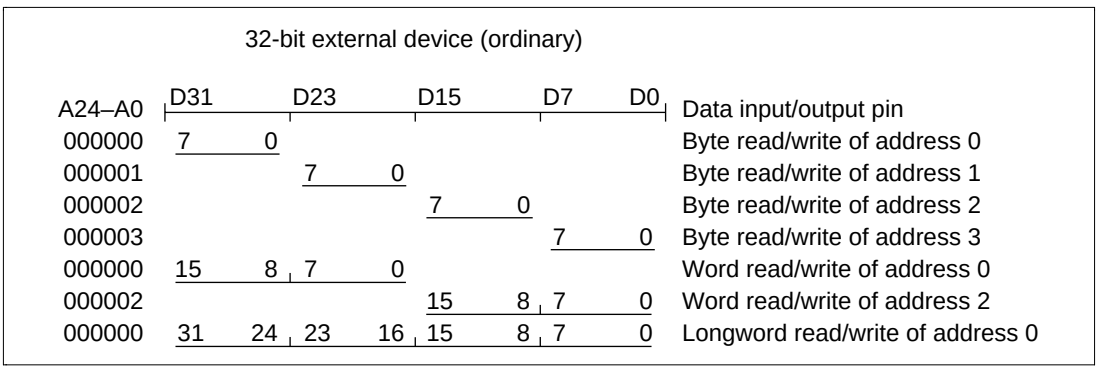



**Figure 7.2 32-Bit External Devices and Their Access Units**

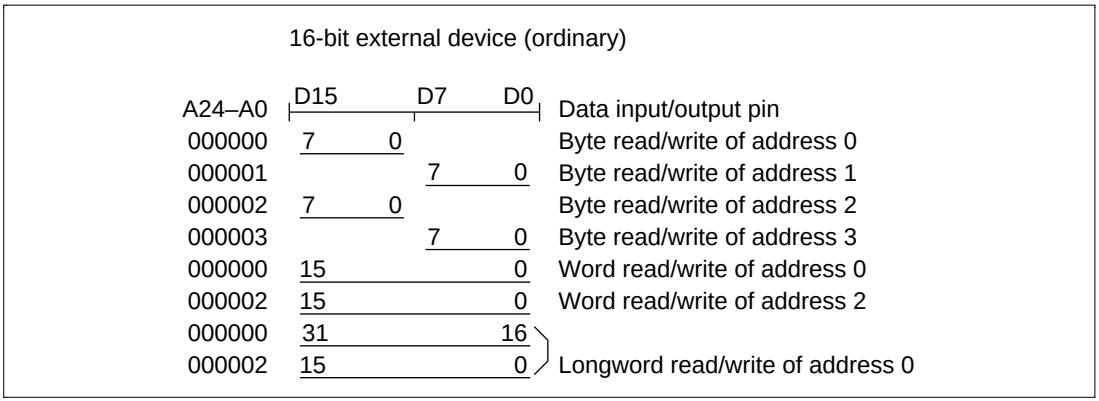



**Figure 7.3 16-Bit External Devices and Their Access Units**

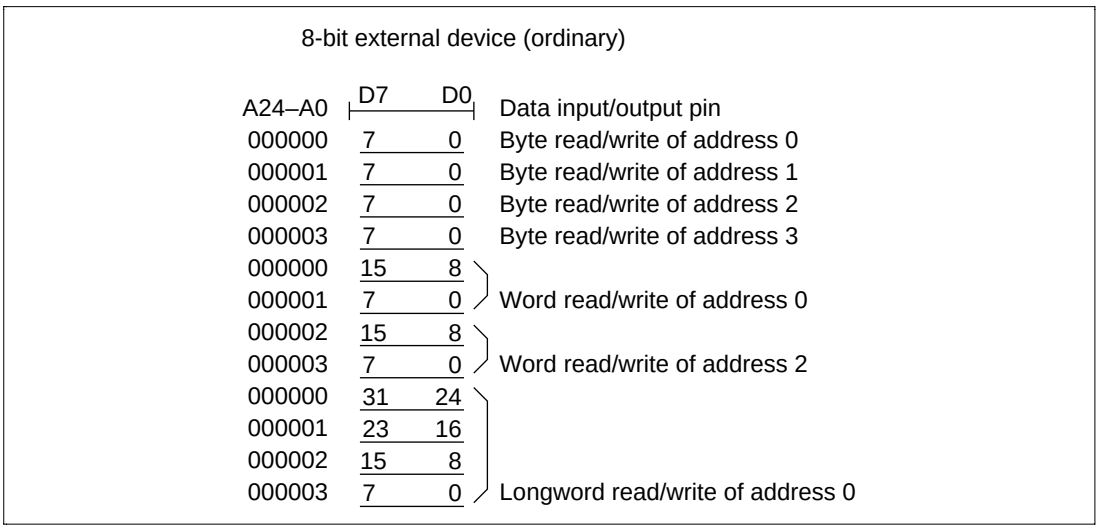



**Figure 7.4 8-Bit External Devices and Their Access Units**

### 7.3.2 Connection to Little-Endian Devices

The chip provides a conversion function in CS2, CS4 space for connection to and to maintain data compatibility with devices that use little-endian format (in which the LSB is the 0 position in the byte data lineup). When the endian specification bit of BCR1 is set to 1, CS2, CS4 space is little-endian. The relationship between device data width and access unit for little-endian format is shown in figures 7.5, 7.6, and 7.7. When sharing memory or the like with a little-endian bus master, the SH7612 connects D31–D24 to the least significant byte (LSB) of the other bus master and D7–D0 to the most significant byte (MSB), when the bus width is 32 bits. When the width is 16 bits, the SH7612 connects D15–D8 to the least significant byte of the other bus master and D7–D0 to the most significant byte. Figure 7.8 shows an example of connection to a 16-bit external device.

Only data conversion is supported by this function. For this reason, be careful not to place program code or constants in the CS2, CS4 space. When this function is used, make sure that the access unit is the same for writing and reading. For example, data written by longword access should be read by longword access. If the read access unit is different from the write access unit, an incorrect value will be read.

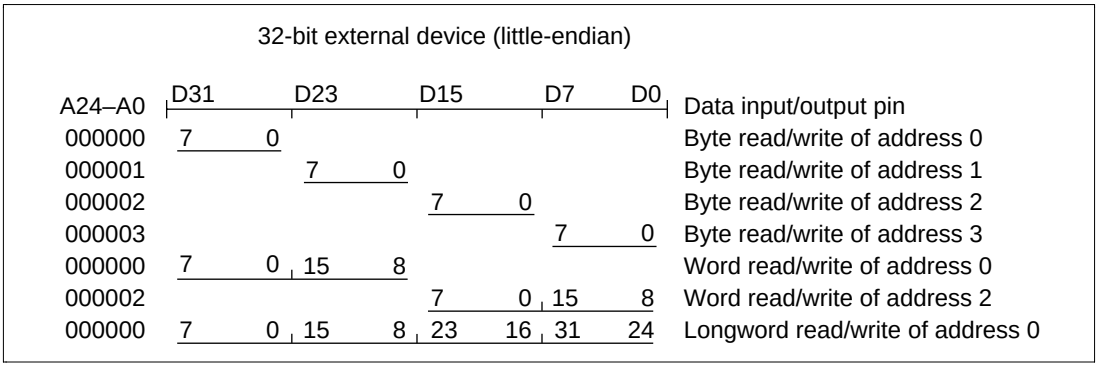



**Figure 7.5 32-Bit External Devices and Their Access Units**

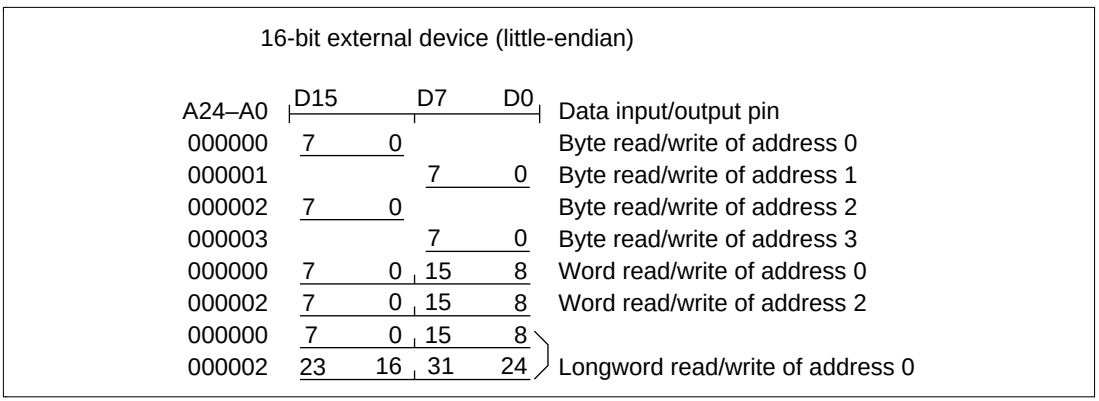



**Figure 7.6 16-Bit External Devices and Their Access Units**

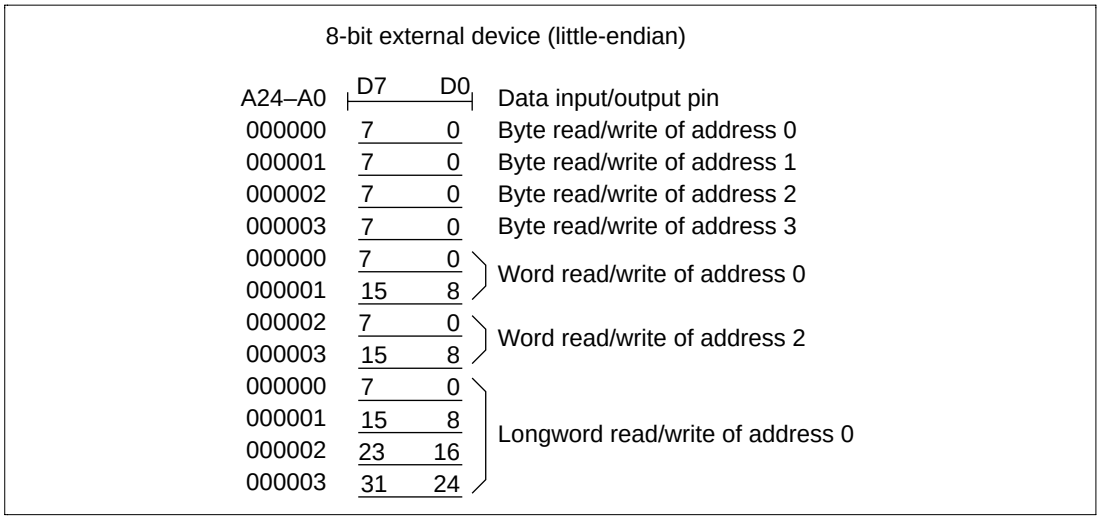



**Figure 7.7 8-Bit External Devices and Their Access Units**

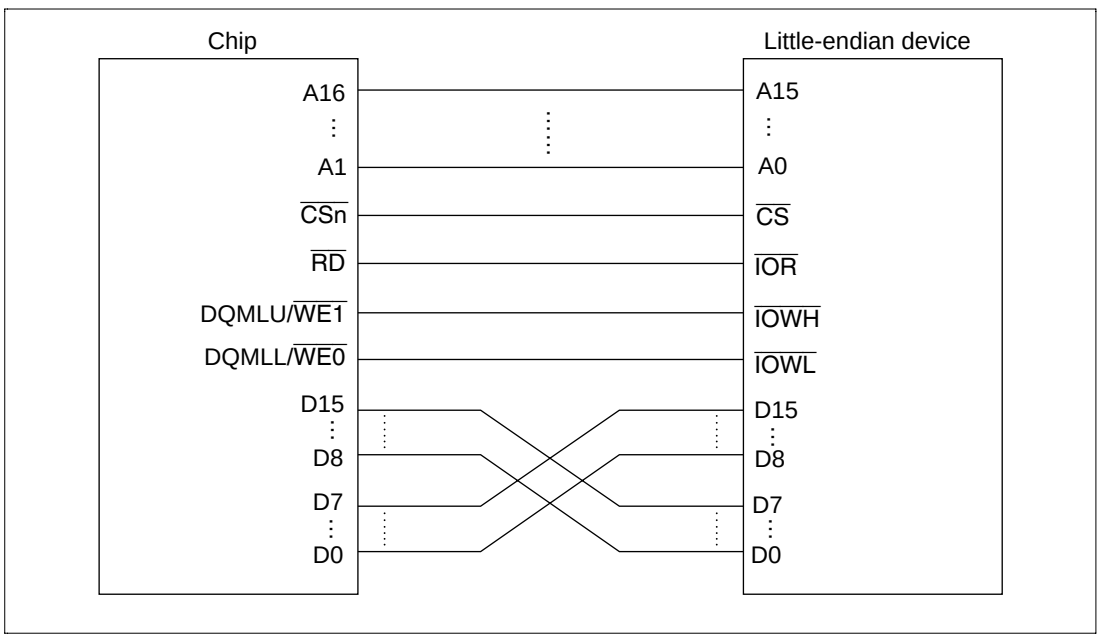



**Figure 7.8 Example of Connection to a 16-bit External Device**

## 7.4 Accessing Ordinary Space

### 7.4.1 Basic Timing

A strobe signal is output by ordinary space accesses of CS0–CS4 spaces to provide primarily for SRAM direct connections. Figure 7.8 shows the basic timing of ordinary space accesses. Ordinary accesses without waits end in 2 cycles. The $\overline{BS}$ signal is asserted for 1 cycle to indicate the start of the bus cycle. The $\overline{CSn}$ signal is negated by the fall of clock T2 to ensure the negate period. The negate period is thus half a cycle when accessed at the minimum pitch.

The access size is not specified during a read. The correct access start address will be output to the LSB of the address, but since no access size is specified, the read will always be 32 bits for 32-bit devices and 16 bits for 16-bit devices. For writes, only the $\overline{WE}$ signal of the byte that will be written is asserted. For 32-bit devices, $\overline{WE3}$ specifies writing to a 4n address and $\overline{WE0}$ specifies writing to a 4n+3 address. For 16-bit devices, $\overline{WE1}$ specifies writing to a 2n address and $\overline{WE0}$ specifies writing to a 2n+1 address. For 8-bit devices, only $\overline{WE0}$ is used.

When data buses are provided with buffers, the $\overline{RD}$ signal must be used for data output in the read direction. When RD/$\overline{WR}$ signals do not perform accesses, the chip stays in read status, so there is a danger of conflicts occurring with output when this is used to control the external data buffer.

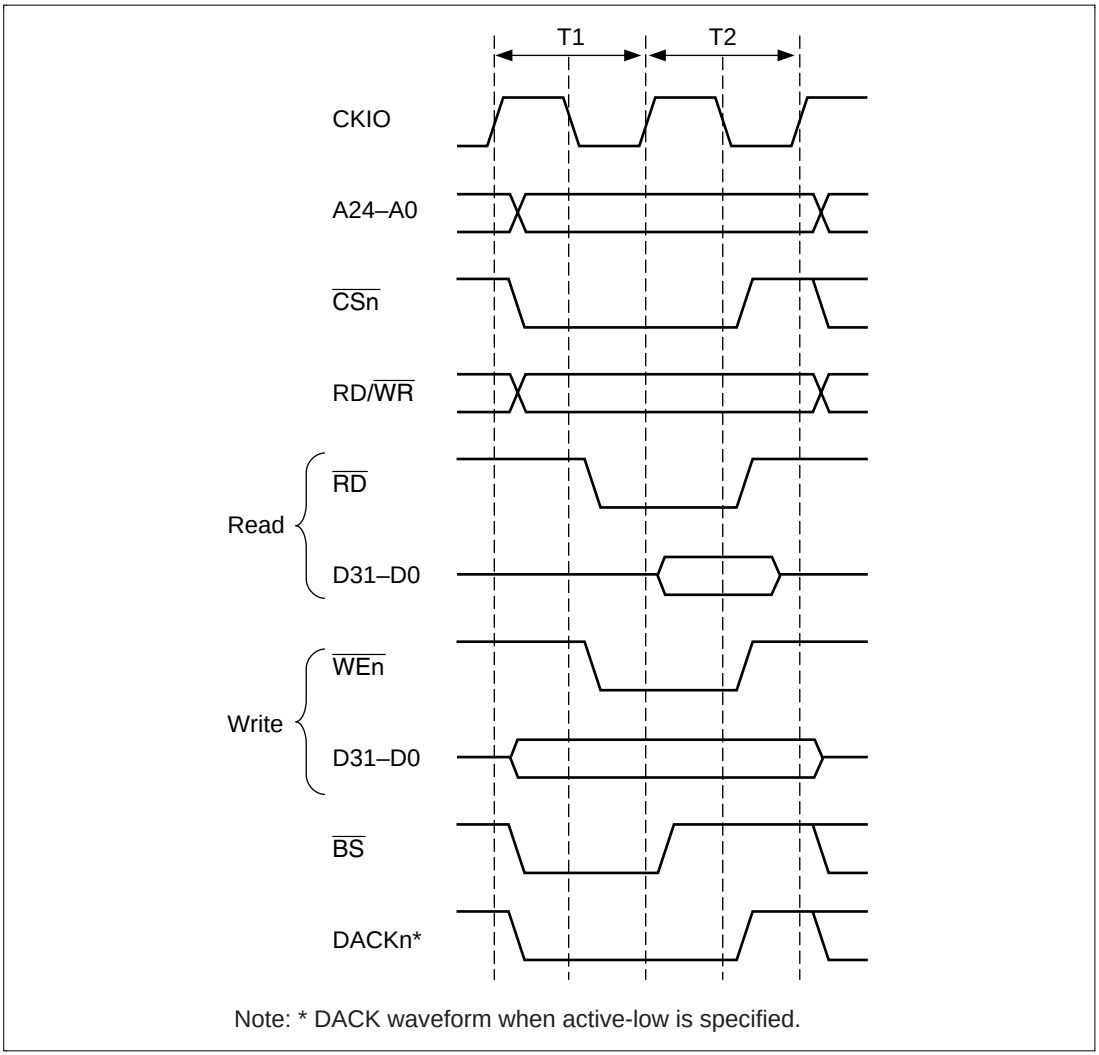



**Figure 7.9 Basic Timing of Ordinary Space Access**

When making a word or longword access with an 8-bit bus width, or a longword access with a 16-bit bus width, the bus state controller performs multiple accesses.

When clock ratio Iø : Eø is other than 1 : 1, the basic timing shown in figure 7.9 is repeated, but when clock ratio Iϕ : Eϕ is 1 : 1, burst access with no $\overline{\text{CSn}}$ negate period is performed as shown in figure 7.10.

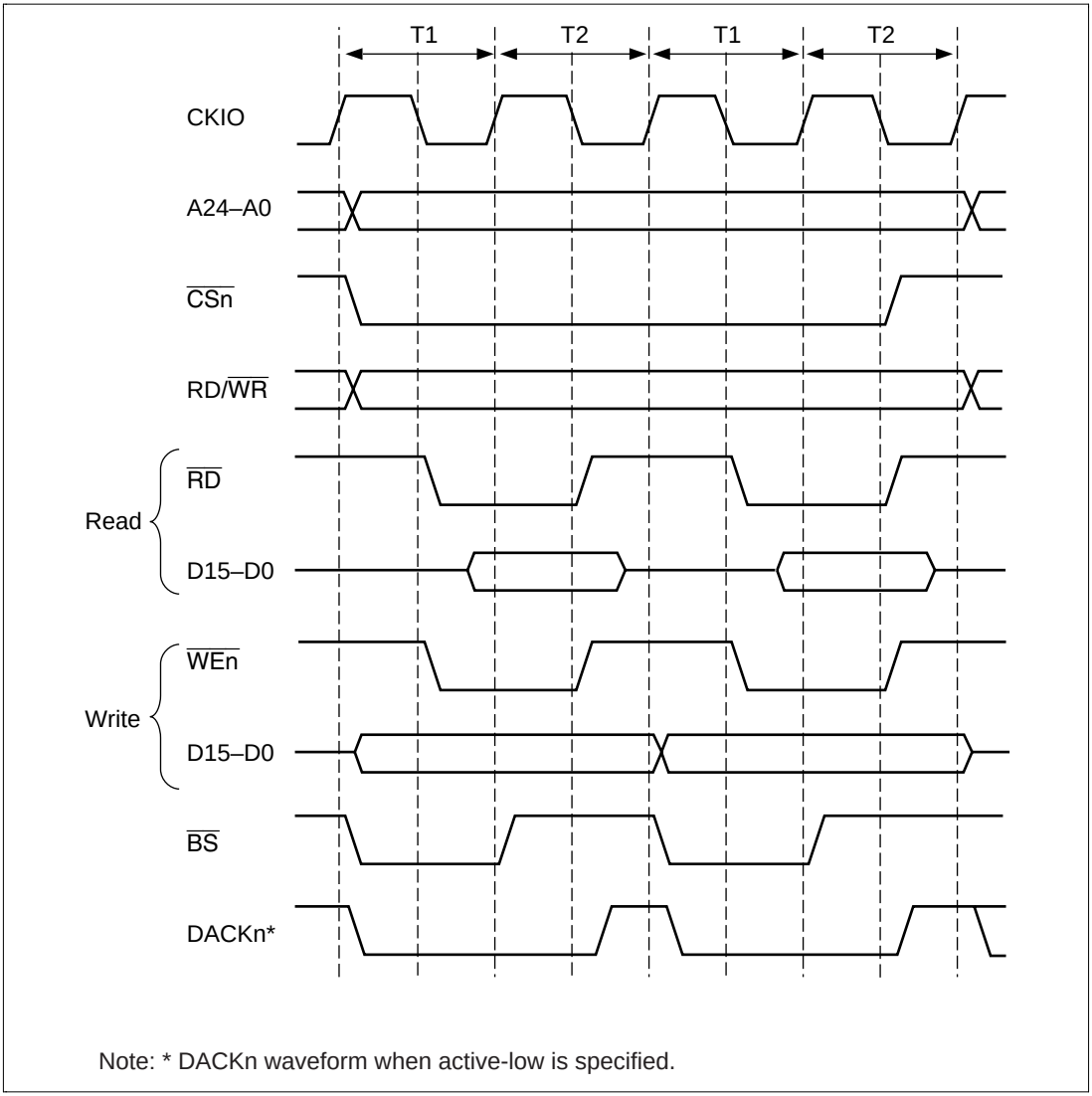



**Figure 7.10 Timing of Longword Access in Ordinary Space Using 16-Bit Bus Width (Clock Ratio Iϕ : Eϕ = 1 : 1)**

Figure 7.11 shows an example of 32-bit data width SRAM connection, figure 7.12 an example of 16-bit data width SRAM connection, and figure 7.13 an example of 8-bit data width SRAM connection.

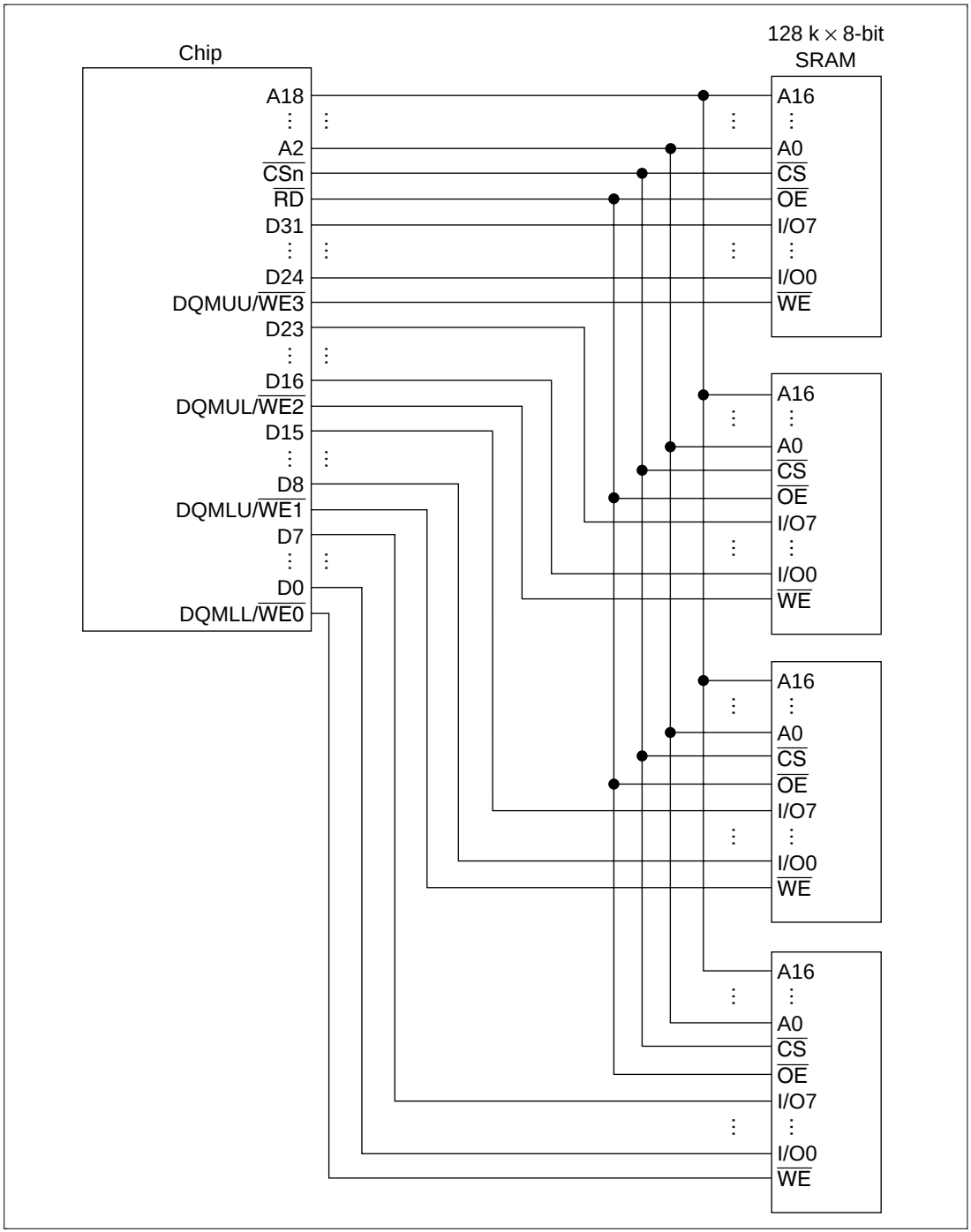



**Figure 7.11 Example of 32-Bit Data Width SRAM Connection**

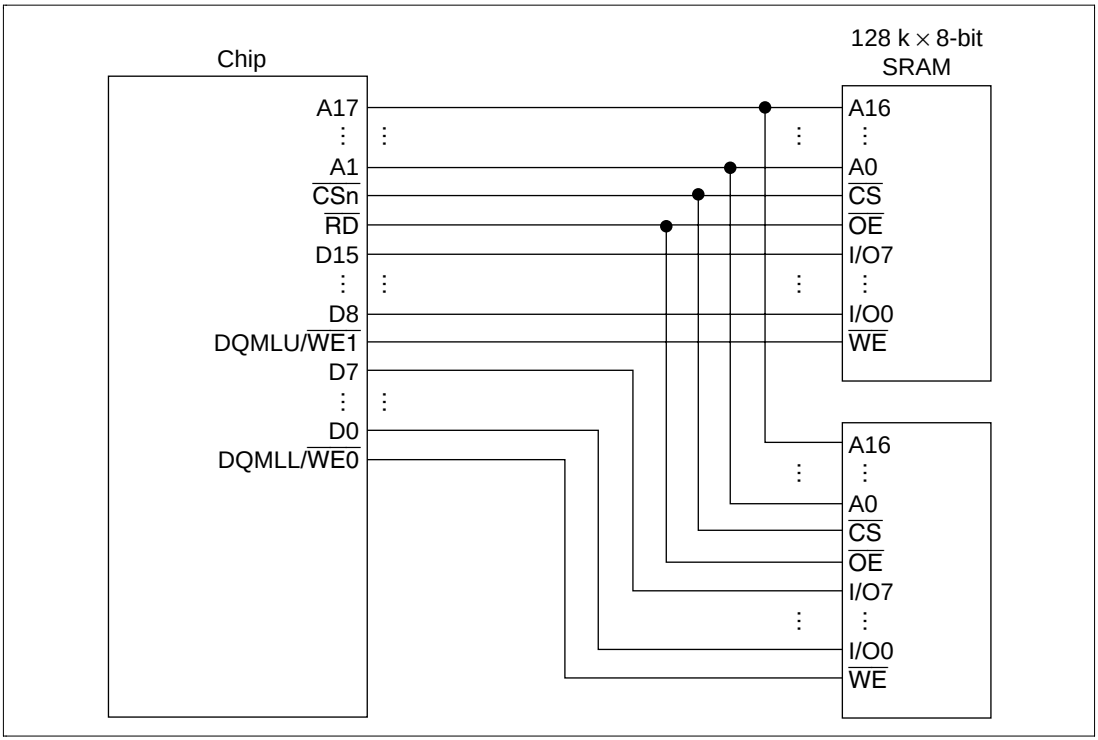



**Figure 7.12 Example of 16-Bit Data Width SRAM Connection**

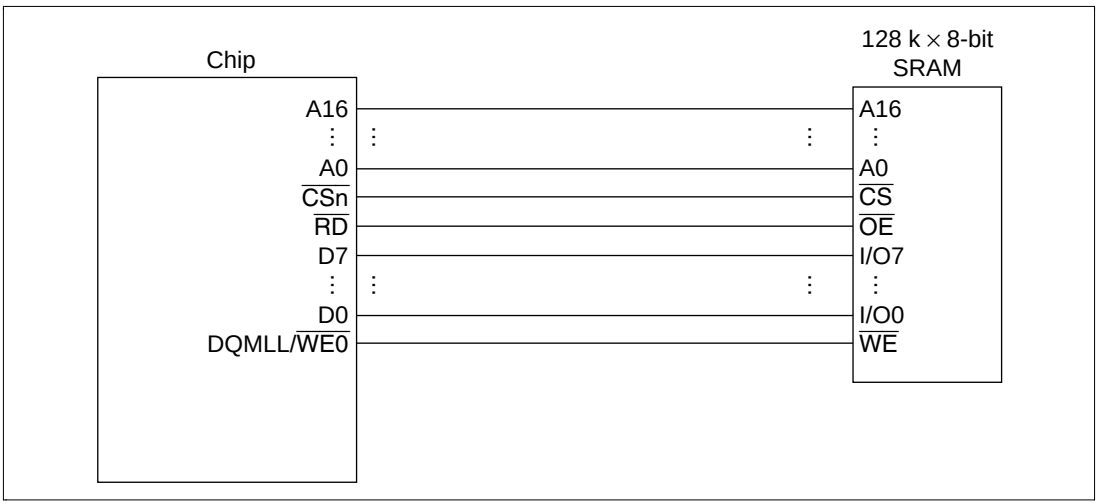



**Figure 7.13 Example of 8-Bit Data Width SRAM Connection**

### 7.4.2 Wait State Control

The number of wait states inserted into ordinary space access states can be controlled using the WCR1, WCR2, BCR1 and BCR3 register settings. When the Wn1 and Wn0 wait specification bits in WCR1, WCR2 for the given CS space are 01 or 10, software waits are inserted according to the wait specification. When Wn1 and Wn0 are 11, wait cycles are inserted according to the long wait specification bit AnLW in BCR1, BCR3. The long wait specification in BCR1, BCR3 can be made independently for CS0, CS1 and CS4 spaces, but the same value must be specified for CS2 and CS3 spaces. All WCR1 specifications are independent. By means of WCR1, WCR2, BCR1, and BCR3, a Tw cycle is inserted as a wait cycle as long as the number of specified cycles at the wait timing for ordinary access space shown in figure 7.14. The names of the control bits that specify Tw for each CS space are shown in table 7.5.

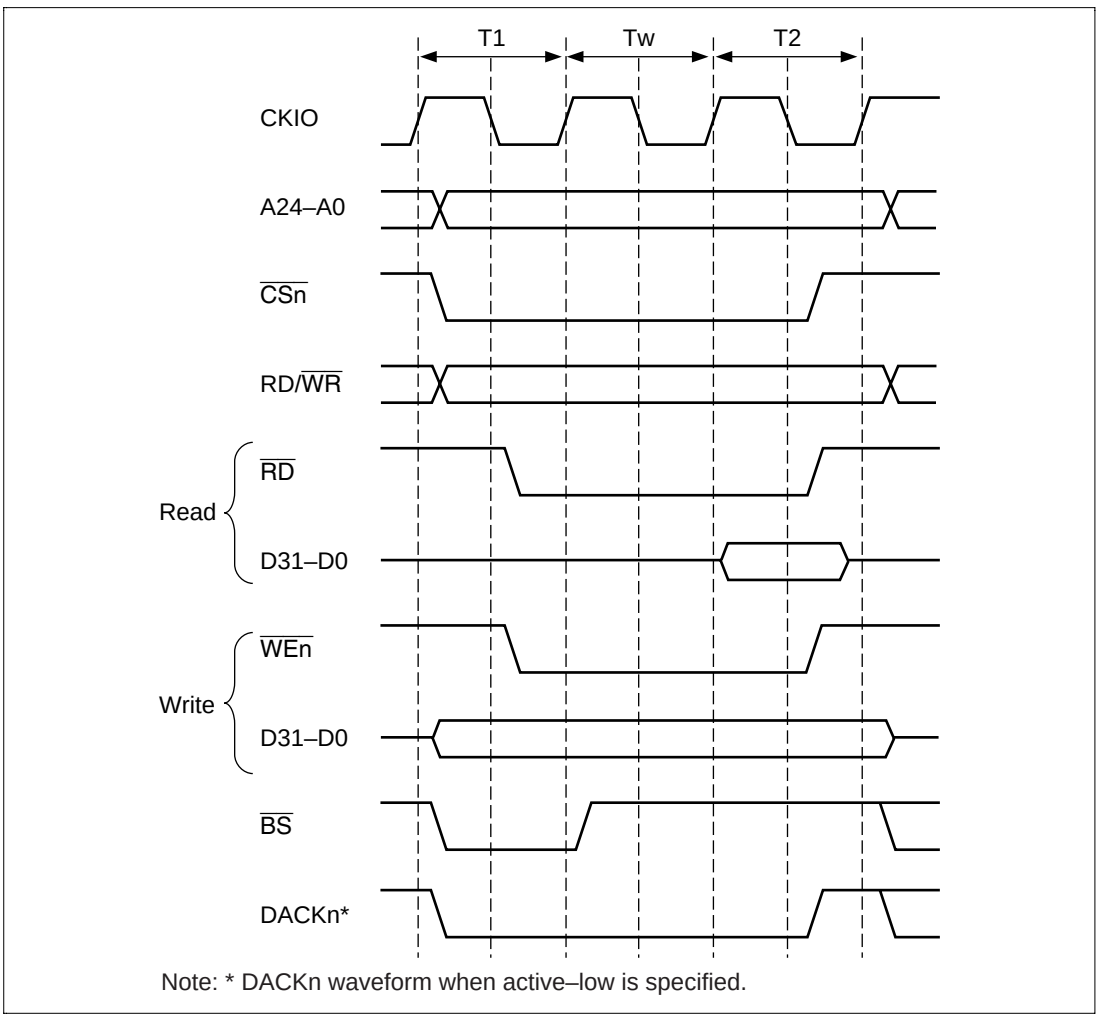



**Figure 7.14 Wait Timing of Ordinary Space Access (Software Wait Only)**

**Table 7.5 $\overline{\text{CSn}}$ Spaces and Tw Specification Bits**

| | BCR3 | BCR1 | | WCR1 | | WCR2 | | Tw |
|---|---|---|---|---|---|---|---|---|
| CS0 | A0LW2 | A0LW1 | A0LW0 | W01 | W00 | — | — | 0–14 |
| CS1 | A1LW2 | A1LW1 | A2LW0 | W11 | W10 | — | — | 0–14 |
| CS2 | AHLW2 | AHLW1 | AHLW0 | W21 | W20 | — | — | 0–14 |
| CS3 | AHLW2 | AHLW1 | AHLW0 | W31 | W30 | — | — | 0–14 |
| CS4 | A4LW2 | A4LW1 | A4LW0 | — | — | W41 | W40 | 0–14 |

When a wait is specified by software using WCR1 and WCR2 (Wn1, Wn0), and the external wait mask bit (AnWM) is cleared to 0 in WCR2, the wait input $\overline{\text{WAIT}}$ signal from outside is sampled. Figure 7.15 shows $\overline{\text{WAIT}}$ signal sampling. A 2-cycle wait is specified as a software wait. The sampling is performed when the Tw state shifts to the T2 state, so there is no effect even when the $\overline{\text{WAIT}}$ signal is asserted in the T1 cycle or the first Tw cycle. The $\overline{\text{WAIT}}$ signal is sampled at the clock fall.

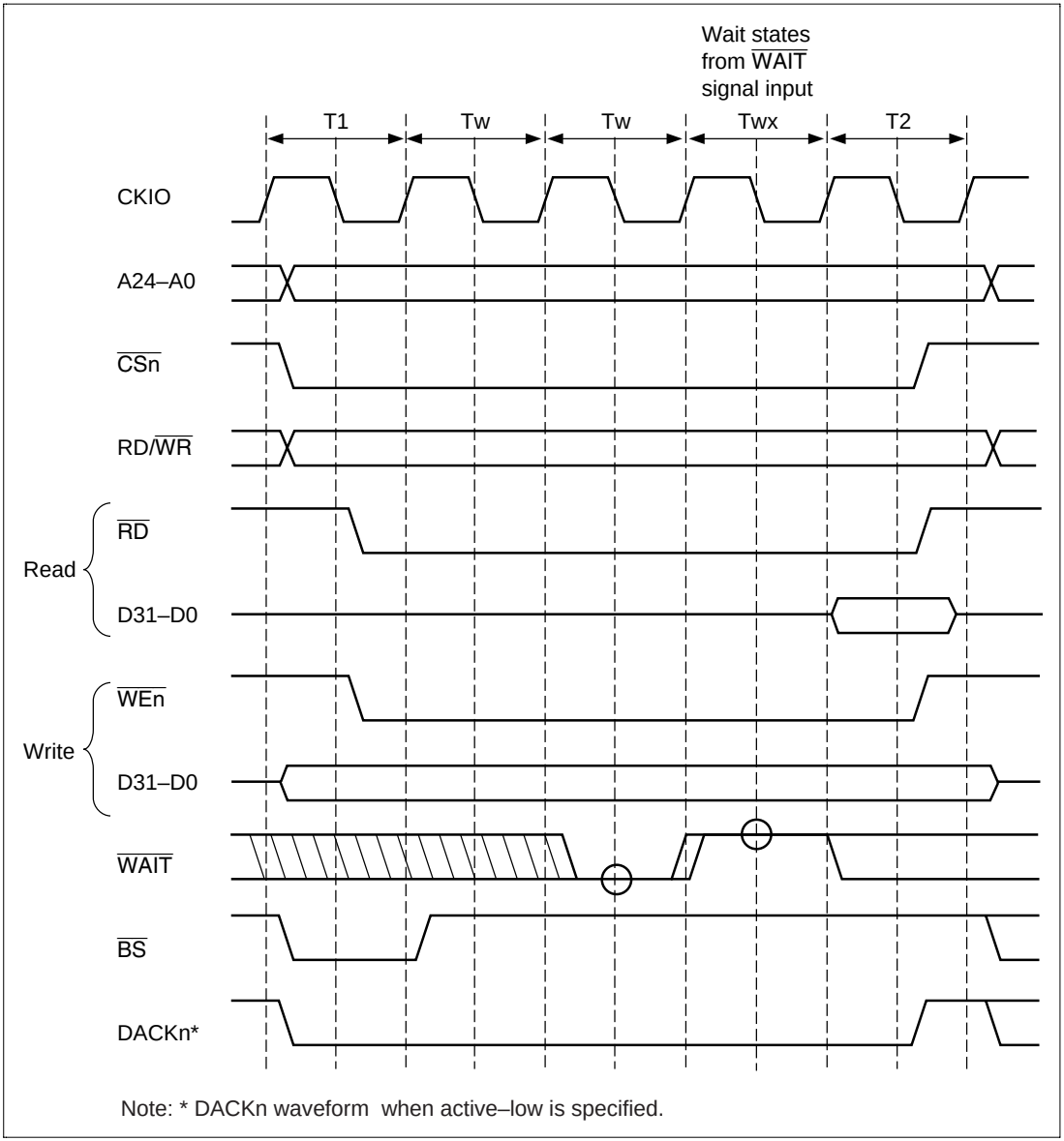



**Figure 7.15 Wait State Timing of Ordinary Space Access (Wait States from $\overline{\text{WAIT}}$ Signal)**

For the CS0–CS3 spaces, $\overline{CS}$, $\overline{RD}$, and $\overline{WEn}$ are negated for one cycle after negation of the external wait signal is accepted, as shown in figure 7.15. For the CS4 space, the number of cycles before $\overline{CS}$, $\overline{RD}$, and $\overline{WEn}$ are negated after acceptance of external wait negation can be set as 1, 2, or 4 by means of bits A4WD1 and A4WD0 in WCR. Figure 7.16 shows an example.

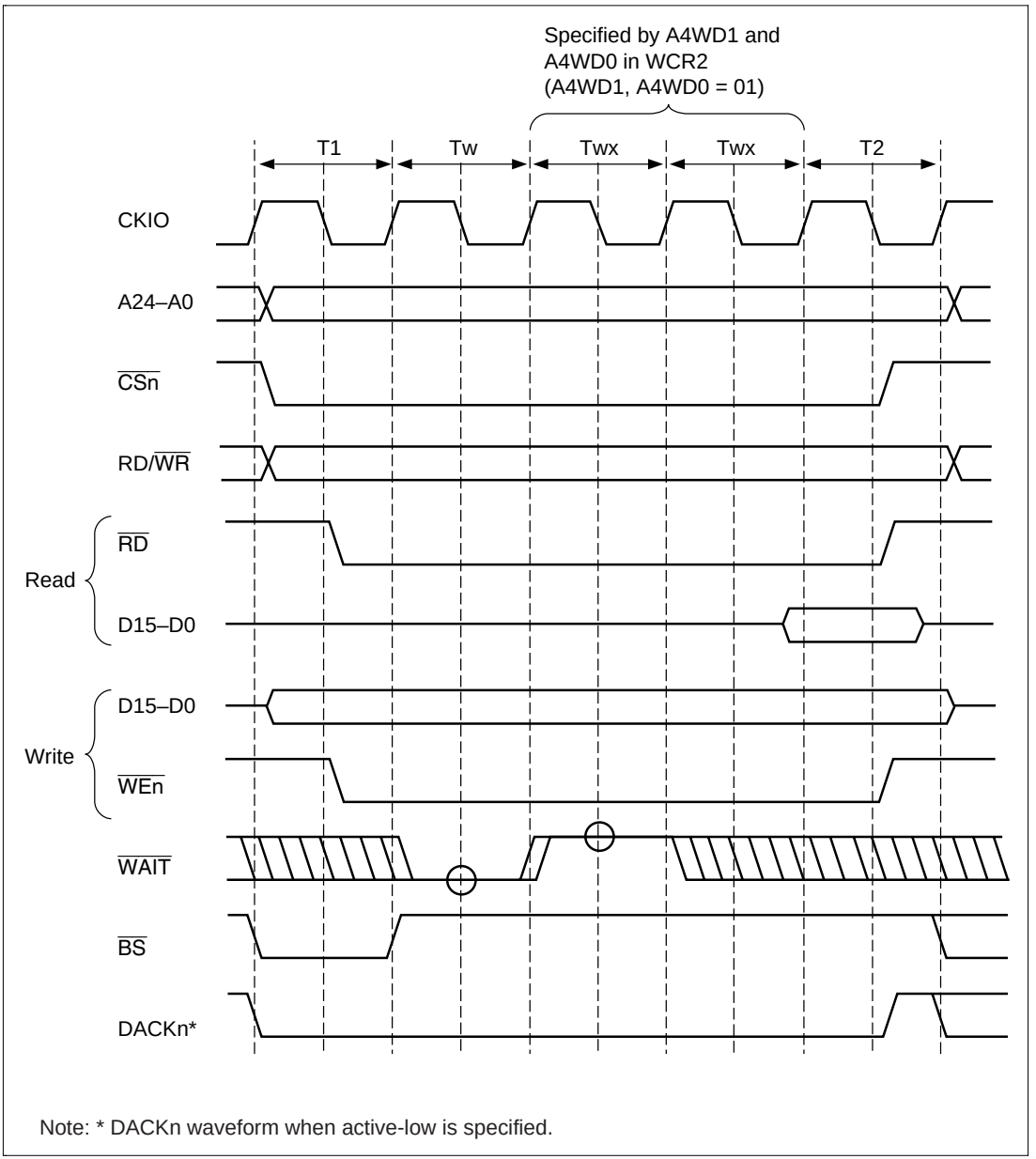



**Figure 7.16 Wait State Timing of Ordinary Space in CS4 Space**

### 7.4.3 $\overline{CS}$ Assertion Period Extension

Idle cycles can be inserted to prevent extension of the $\overline{RD}$ or $\overline{WEn}$ assertion period beyond the length of the $\overline{CSn}$ assertion period by setting control bits in WCR3. This allows for flexible interfacing to external circuit. The timing is shown in figure 7.17. Th and Tf cycles are added respectively before and after the ordinary cycle. Signals other than $\overline{RD}$ and $\overline{WEn}$ are asserted in this cycle, but $\overline{RD}$ and $\overline{WEn}$ are not. In addition, data is extended up to the Tf cycle, which is effective for devices with slow write operations.

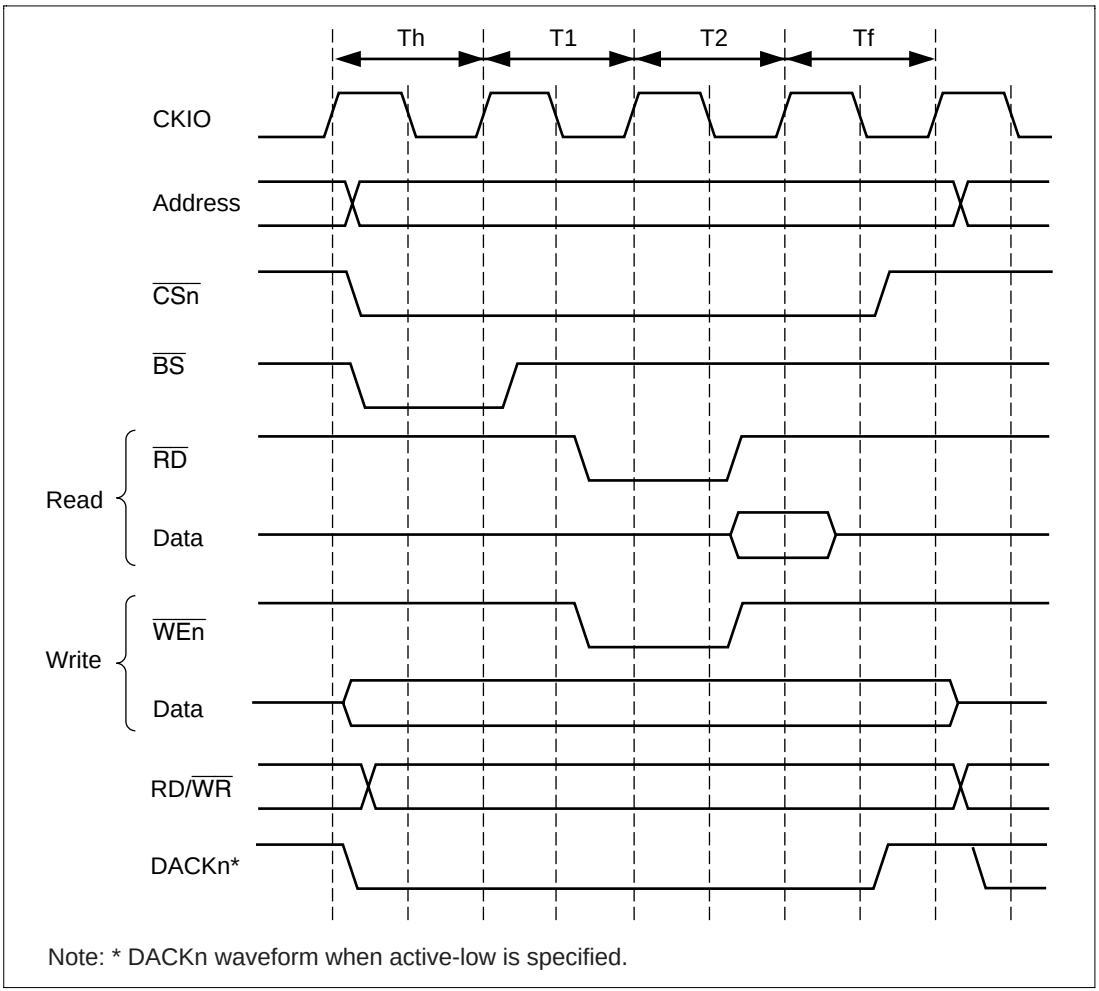



**Figure 7.17 $\overline{CS}$ Assertion Period Extension Function**

For the CS0–CS4 spaces, For spaces CS0—CS4, Th and Tf can be set as follows.

| | Th | Tf | WCR3 |
|---|---|---|---|
| CS0—3 | 0—2 | | AnSHW1, AnSHW0 (Th = Tf) n =0—3 |
| CS4 | 0—7 | 0—5 | Th: A4SW2—0<br>Tf: A4HW1—0 |

## 7.5 Synchronous DRAM Interface

### 7.5.1 Synchronous DRAM Direct Connection

Seven kinds of synchronous DRAM can be connected: 2-Mbit (128k × 16), 4-Mbit (256k × 16), 16-Mbit (1M × 16, 2M × 8, and 4M × 4), and 64-Mbit (4M × 16 and 8M × 8). This chip supports 64-Mbit synchronous DRAMs internally divided into two or four banks, and other synchronous DRAMs internally divided into two banks. Since synchronous DRAM can be selected by the $\overline{CS}$ signal, CS2 and CS3 spaces can be connected using a common $\overline{RAS}$ or other control signal. When the enable bits for DRAM and other memory (DRAM2–DRAM0) in BCR1 are set to 001, CS2 is ordinary space and CS3 is synchronous DRAM space. When the DRAM2–0 bits are set to 100, CS2 is synchronous DRAM space and CS3 is ordinary space. When the bits are set to 101, both CS2 and CS3 are synchronous DRAM spaces.

Supported synchronous DRAM operating modes are burst read/single write mode (initial setting) and burst read/burst write mode. The burst length depends on the data bus width, comprising 8 bursts for a 16-bit width, and 4 bursts for a 32-bit width. The data bus width is specified by the SZ bit in MCR. Burst operation is always performed, so the burst enable (BE) bit in MCR is ignored. Switching to burst write mode is performed by means of the BWE bit in BCR3.

Control signals for directly connecting synchronous DRAM are the $\overline{RAS}$, $\overline{CAS}$/$\overline{OE}$, RD/$\overline{WR}$, $\overline{CS2}$ or $\overline{CS3}$, DQMUU, DQMUL, DQMLU, DQMLL, and CKE signals. Signals other than $\overline{CS2}$ and $\overline{CS3}$ are common to every area, and signals other than CKE are valid and fetched only when $\overline{CS2}$ or $\overline{CS3}$ is true. Therefore, synchronous DRAM can be connected in parallel in multiple areas. CKE is negated (to the low level) only when a self-refresh is performed; otherwise it is always asserted (to the high level).

Commands can be specified for synchronous DRAM using the $\overline{RAS}$, $\overline{CAS}$/$\overline{OE}$, RD/$\overline{WR}$, and certain address signals. These commands are NOP, auto-refresh (REF), self-refresh (SELF), all-bank precharge (PALL), specific bank precharge (PRE), row address strobe/bank active (ACTV), read (READ), read with precharge (READA), write (WRIT), write with precharge (WRITA), and mode register write (MRS).

Bytes are specified using DQMUU, DQMUL, DQMLU, and DQMLL. The read/write is performed on the byte whose DQM is low. For 32-bit data, DQMUU specifies 4n address access and DQMLL specifies 4n + 3 address access. For 16-bit data, only DQMLU and DQMLL are

used. Figure 7.18 shows an example in which a 32-bit connection uses a 256k × 16 bit synchronous DRAM. Figure 7.19 shows an example with a 16-bit connection.

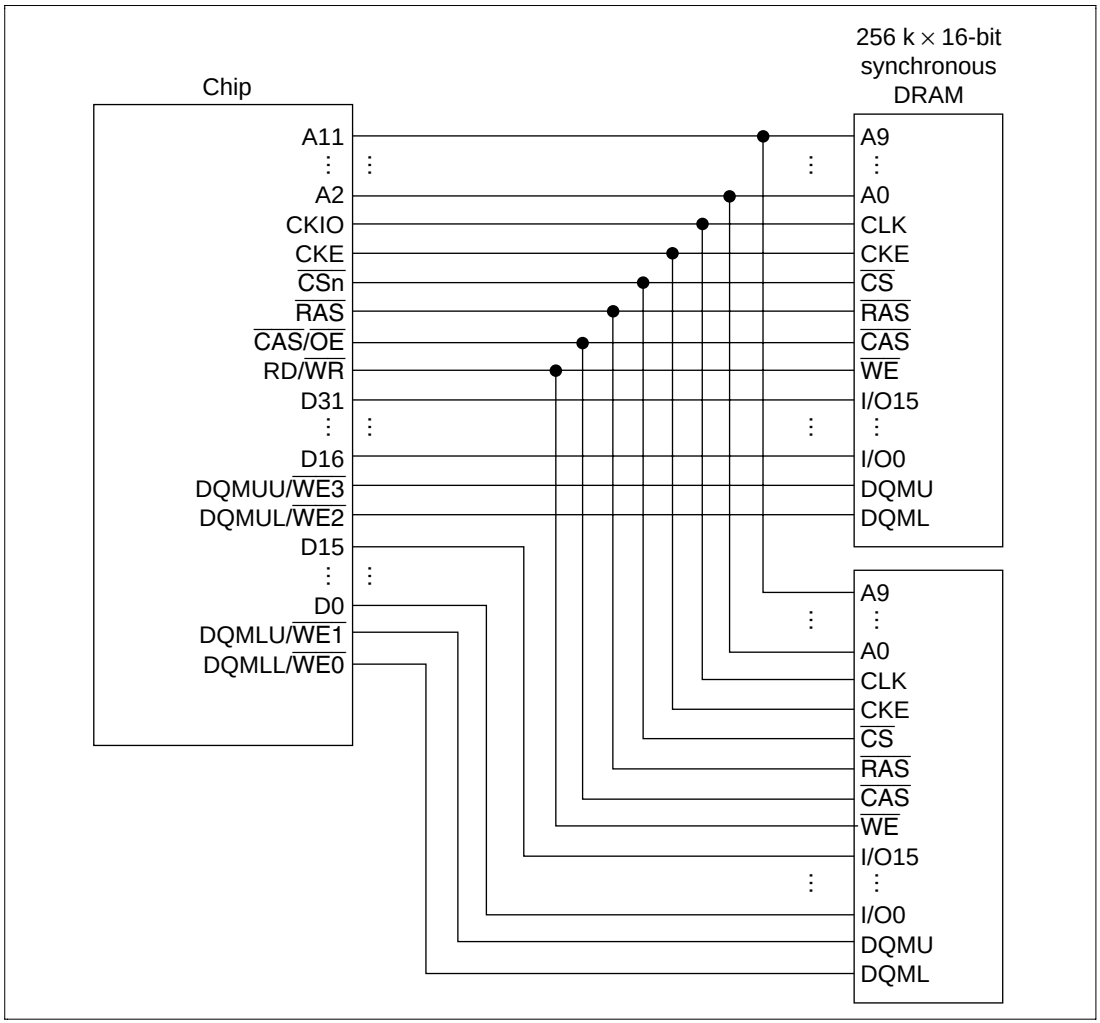



**Figure 7.18 Synchronous DRAM 32-bit Device Connection**

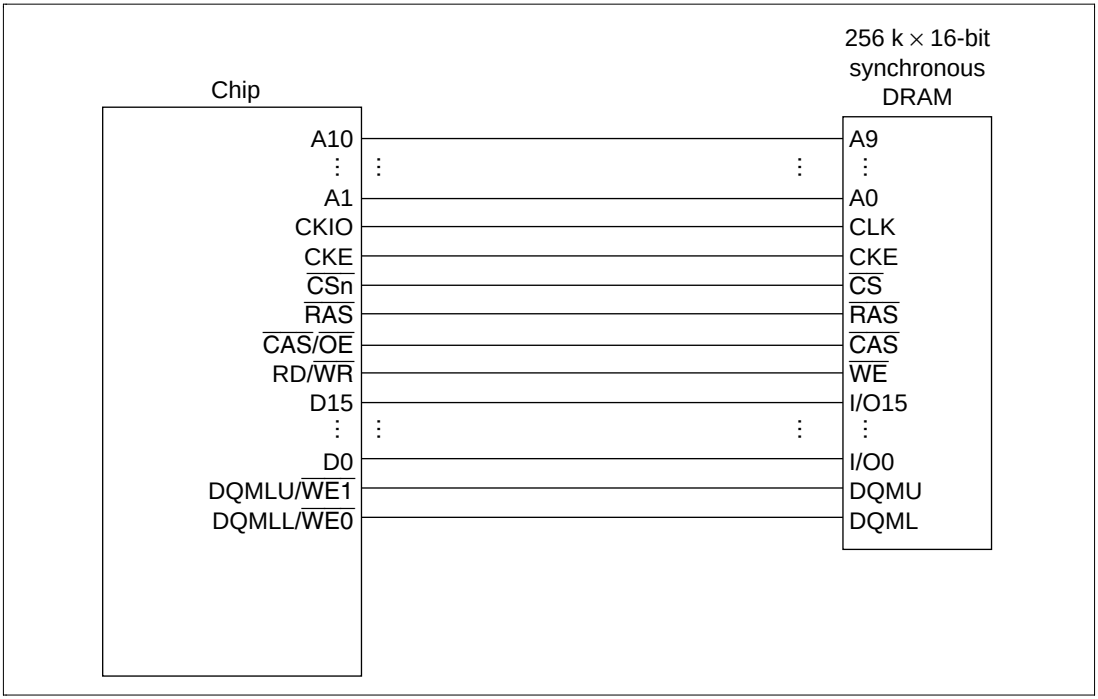



**Figure 7.19 Synchronous DRAM 16-bit Device Connection**

### 7.5.2 Address Multiplexing

Addresses are multiplexed according to the MCR's address multiplex specification bits AMX2–AMX0 and size specification bit SZ so that synchronous DRAMs can be connected to the SH7612 directly without an external multiplex circuit. Table 7.6 shows the relationship between the multiplex specification bits and bit output to the address pins.

A24–A16 always output the original value regardless of multiplexing.

When SZ = 0, the data width on the synchronous DRAM side is 16 bits and the LSB of the device's address pins (A0) specifies word address. The A0 pin of the synchronous DRAM is thus connected to the A1 pin of the SH7612, the rest of the connection proceeding in the same order, beginning with the A1 pin to the A2 pin.

When SZ = 1, the data width on the synchronous DRAM side is 32 bits and the LSB of the device's address pins (A0) specifies longword address. The A0 pin of the synchronous DRAM is thus connected to the A2 pin of the SH7612, the rest of the connection proceeding in the same order, beginning with the A1 pin to the A3 pin.

**Table 7.6 SZ and AMX Bits and Address Multiplex Output**

| Setting | | | | | External Address Pins | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SZ | AMX2 | AMX1 | AMX0 | Output Timing | A1–A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 |
| 1 | 0 | 0 | 0 | Column address | A1–A8 | A9 | A10 | A11 | L/H[*1] | A21[*2] | A14 | A15 |
| | | | | Row address | A9–A16 | A17 | A18 | A19 | A20 | A21[*2] | A22 | A23 |
| 1 | 0 | 0 | 1 | Column address | A1–A8 | A9 | A10 | A11 | L/H[*1] | A22[*2] | A14 | A15 |
| | | | | Row address | A10–A17 | A18 | A19 | A20 | A21 | A22[*2] | A23 | A24 |
| 1 | 0 | 1 | 0 | Column address | A1–A8 | A9 | A10 | A11 | L/H[*1] | A23[*2] | A14 | A15 |
| | | | | Row address | A11–A18 | A19 | A20 | A21 | A22 | A23[*2] | A24 | A25 |
| 1 | 0 | 1 | 1 | Column address | A1–A8 | A9 | L/H[*1] | A19[*2] | A12 | A13 | A14 | A15 |
| | | | | Row address | A9–A16 | A17 | A18 | A19[*2] | A20 | A21 | A22 | A23 |
| 1 | 1 | 0 | 0 | Column address | A1–A8 | A9 | A10 | A11 | L/H[*1] | A13 | A22[*3] | A23[*2] |
| | | | | Row address | A9–A16 | A17 | A18 | A19 | A20 | A21 | A22[*3] | A23[*2] |
| 1 | 1 | 0 | 1 | Column address | A1–A8 | A9 | A10 | A11 | L/H[*1] | A13 | A23[*3] | A24[*2] |
| | | | | Row address | A10–A17 | A18 | A19 | A20 | A21 | A22 | A23[*3] | A24[*2] |
| 1 | 1 | 1 | 1 | Column address | A1–A8 | A9 | L/H[*1] | A18[*2] | A12 | A13 | A14 | A15 |
| | | | | Row address | A9–A16 | A17 | A17 | A18[*2] | A20 | A21 | A22 | A23 |
| 0 | 0 | 0 | 0 | Column address | A1–A8 | A9 | A10 | L/H[*1] | A20[*2] | A13 | A14 | A15 |
| | | | | Row address | A9–A16 | A17 | A18 | A19 | A20[*2] | A21 | A22 | A23 |

**Table 7.6 SZ and AMX Bits and Address Multiplex Output (cont)**

| Setting | | | | | External Address Pins | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SZ | AMX2 | AMX1 | AMX0 | Output Timing | A1–A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 |
| 0 | 1 | 0 | 0 | Column address | A1–A8 | A9 | A10 | LH*1 | A12 | A21*3 | A22*2 | A15 |
| | | | | Row address | A9–A16 | A17 | A18 | A19 | A20 | A21*3 | A22*2 | A23 |
| 0 | 0 | 1 | 1 | Column address | A1–A8 | L/H*1 | A18*2 | A11 | A12 | A13 | A14 | A15 |
| | | | | Row address | A9–A16 | A17 | A18*2 | A19 | A20 | A21 | A22 | A23 |
| 0 | 1 | 1 | 1 | Column address | A1–A8 | L/H*1 | A17*2 | A11 | A12 | A13 | A14 | A15 |
| | | | | Row address | A9–A16 | A16 | A17*2 | A19 | A20 | A21 | A22 | A23 |

Notes: AMX2–AMX0 setting 110 is reserved and must not be used. When SZ = 0, AMX2–AMX0 settings 001, 010, and 101 are also reserved and must not be used.

1. L/H is a bit used to specify commands. It is fixed at L or H according to the access mode.
2. Bank address specification.
3. Bank address specification when using four banks.

### 7.5.3 Burst Reads

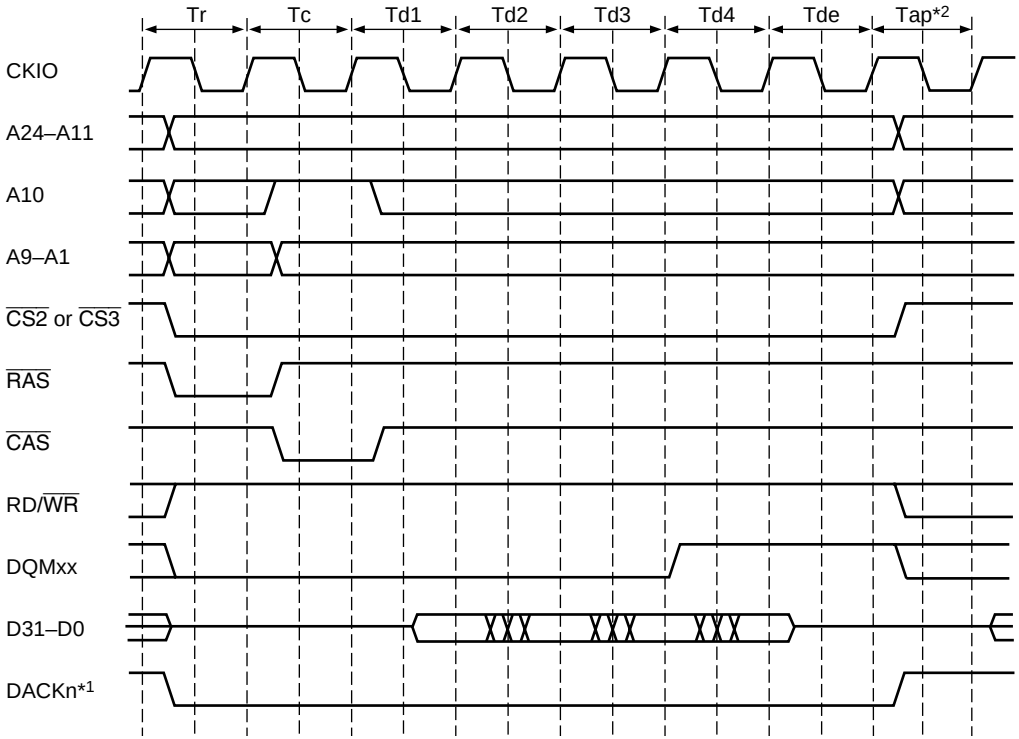
Figure 7.20 shows the timing chart for burst reads. In the following example, 2 synchronous DRAMs of 256k × 16 bits are connected, the data width is 32 bits and the burst length is 4. After a Tr cycle that performs ACTV command output, a READA command is issued in the Tc cycle, read data is accepted in cycles Td1 to Td4, and the end of the read sequence is waited for in the Tde cycle. One Tde cycle is issued when Iø : Eø ≠ 1 : 1, and two cycles when Iø : Eø = 1 : 1. Tap is a cycle for waiting for the completion of the auto-precharge based on the READA command within the synchronous DRAM. During this period, no new access commands are issued to the same bank. Accesses of the other bank of the synchronous DRAM by another CS space are possible. Depending on the TRP1, TRP0 specification in MCR, the chip determines the number of Tap cycles and does not issue a command to the same bank during that period.

Figure 7.20 shows an example of the basic cycle. Because a slower synchronous DRAM is connected, setting WCR1 and MCR bits can extend the cycle. The number of cycles from the ACTV command output cycle Tr to the READA command output cycle Tc can be specified by bits RCD1 and RCD0 in MCR. 00 specifies 1 cycle, 01 specifies 2 cycles, and 10 specifies 3 cycles. For 2 or 3 cycles, a NOP command issue cycle Trw for the synchronous DRAM is inserted

between the Tr cycle and the Tc cycle. The number of cycles between the READA command output cycle Tc and the initial read data fetch cycle Td1 can be specified between 1 cycle and 4 cycles using the W21/W20 and W31/W30 bits in WCR1. The number of cycles at this time corresponds to the number of CAS latency cycles of the synchronous DRAM. When 2 cycles or more, a NOP command issue cycle Tw is inserted between the Tc cycle and the Td1 cycle. The number of cycles in the precharge completion waiting cycle Tap is specified by bits TRP1 and TRP0 in MCR. When CAS latency is 1, a Tap cycle comprising the number of cycles specified by TRP1 and TRP0 is generated. When the CAS latency is 2 or more, a Tap cycle equal to the TRP specification – 1 is generated. During the Tap cycle, no commands other than NOP are issued to the same bank. Figure 7.20 shows an example of burst read timing when RCD1/RCD0 is 01, W31/W30 is 01, and TRP1/TRP0 is 01.

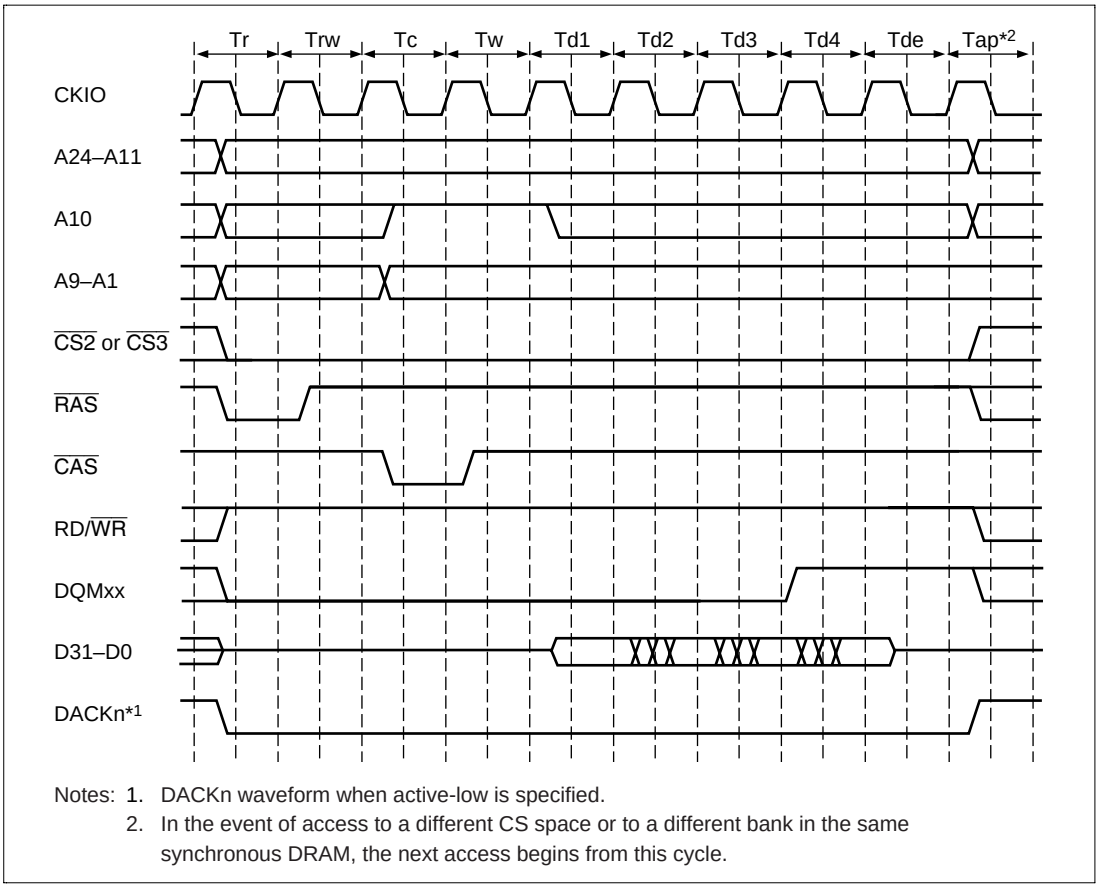
When the data width is 16 bits, 8 burst cycles are required for a 16-byte data transfer. The data fetch cycle goes from Td1 to Td8.

Synchronous DRAM CAS latency is up to 3 cycles, but the CAS latency of the bus state controller can be specified up to 4. This is so that circuits containing latches can be installed between synchronous DRAMs and the chip.




Notes: 1. DACKn waveform when active-low is specified.
2. In the event of access to a different CS space or to a different bank in the same synchronous DRAM, the next access begins from this cycle.

**Figure 7.20 Basic Burst Read Timing (Auto-Precharge)**




Notes: 1. DACKn waveform when active-low is specified.
2. In the event of access to a different CS space or to a different bank in the same synchronous DRAM, the next access begins from this cycle.

**Figure 7.21 Burst Read Wait Specification Timing (Auto-Precharge)**

### 7.5.4 Single Reads

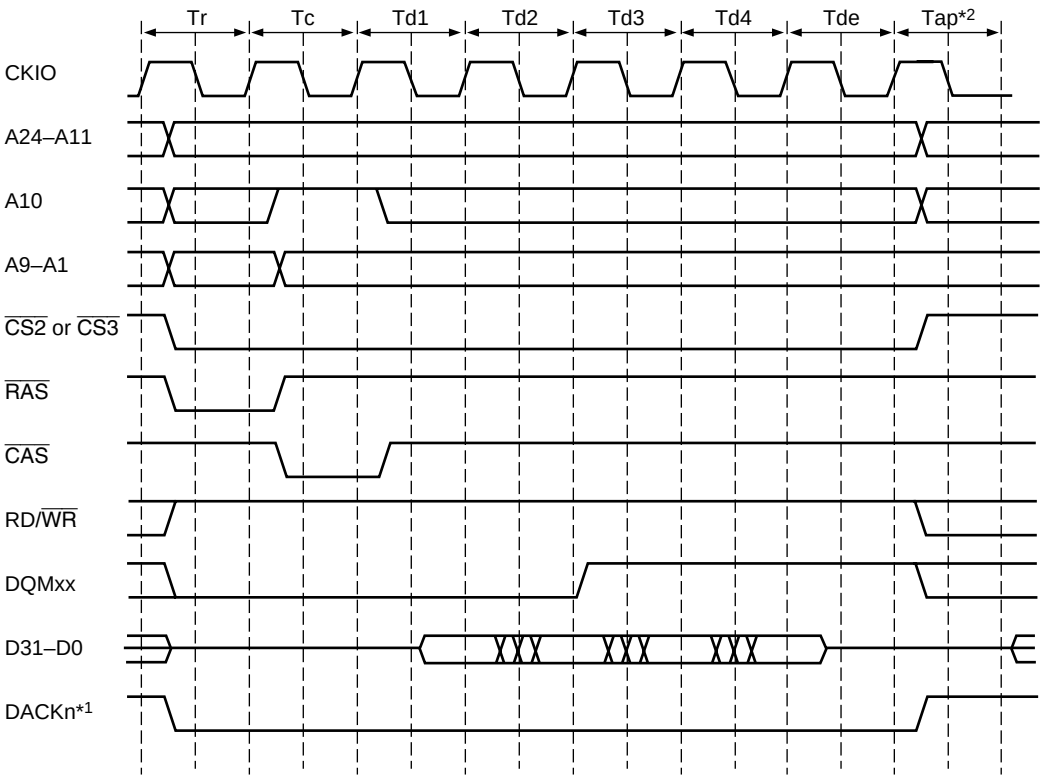
When a cache area is accessed and there is a cache miss, the cache fill cycle is performed in 16-byte units. This means that all the data read in the burst read is valid. On the other hand, when a cache-through area is accessed the required data is a maximum length of 32 bits, and the remaining 12 bytes are wasted. The same kind of wasted data access is produced when synchronous DRAM is specified as the source in a DMA transfer by the DMAC and the transfer unit is other than 16 bytes. Figure 7.22 shows the timing of a single address read. Because the synchronous DRAM is set to the burst read mode, the read data output continues after the required data is received. To avoid data conflict, an empty read cycle is performed from Td2 to Td4 after the required data is read in Td1 and the device waits for the end of synchronous DRAM operation.

When the data width is 16 bits, the number of burst transfers during a read is 8. Data is fetched in cache-through and other DMA read cycles only in the Td1 and Td2 cycles (of the 8 cycles from Td1 to Td8) for longword accesses, and only in the Td1 cycle for word or byte accesses.

The read sequence ends with the Tde cycle. There is one Tde cycle when Iø:Eø ≠ 1:1, and there are two cycles when Iø:Eø = 1:1.

Empty cycles tend to increase the memory access time, lower the program execution speed, and lower the DMA transfer speed, so it is important to avoid accessing unnecessary cache-through areas and to use data structures that enable 16-byte unit transfers by placing data on 16-byte boundaries when performing DMA transfers that specify synchronous DRAM as the source.

In the Tap cycle, a new access command cannot be issued for the same bank. However, access is possible to a different CS space or to a different bank in the same synchronous DRAM.



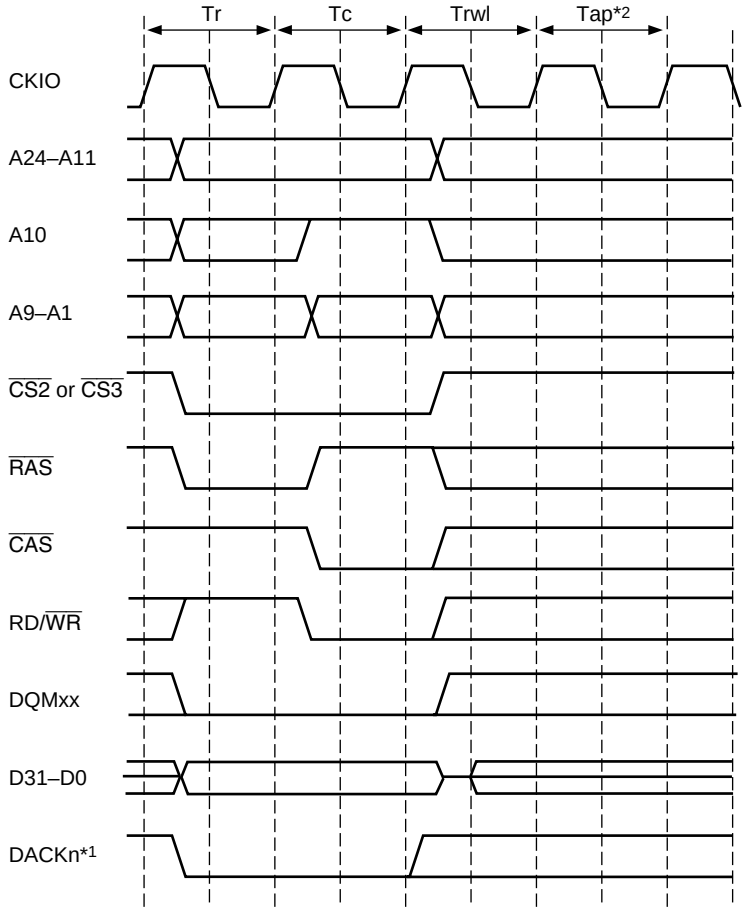

Notes: 1. DACKn waveform when active-low is specified.
2. In the event of access to a different CS space or to a different bank in the same synchronous DRAM, the next access begins from this cycle.

**Figure 7.22 Single Read Timing (Auto-Precharge)**

### 7.5.5 Single Writes

Synchronous DRAM writes are executed as single writes or burst writes according to the specification by the BWE bit in BCR3. Figure 7.23 shows the basic timing chart for single write accesses. After the ACTV command Tr, a WRITA command is issued in Tc to perform an auto-precharge. In the write cycle, the write data is output simultaneously with the write command. When writing with an auto-precharge, the bank is precharged after the completion of the write command within the synchronous DRAM, so no command can be issued to that bank until the precharge is completed. For that reason, besides a Tap cycle to wait for the precharge during read accesses, a Trw1 cycle is added to wait until the precharge is started, and the issuing of any new commands to the same bank is delayed during this period. However, access is possible to a different CS space or to a different bank in the same synchronous DRAM. The number of cycles in the Trw1 cycle can be specified using the TRWL1 and TRWL0 bits in MCR.

The specification in bits TRWL1 and TRWL0 is the number of cycles from Tc to Tap. Therefore, when TRWL1—TRWL0 = 00, the number of Trwl cycles is 0.




Note: 1. DACKn waveform when active-low is specified.
2. In the event of access to a different CS space or to a different bank in the same synchronous DRAM, the next access begins from this cycle.

**Figure 7.23 Basic Single Write Cycle Timing (Auto-Precharge)**

### 7.5.6 Burst Write Mode

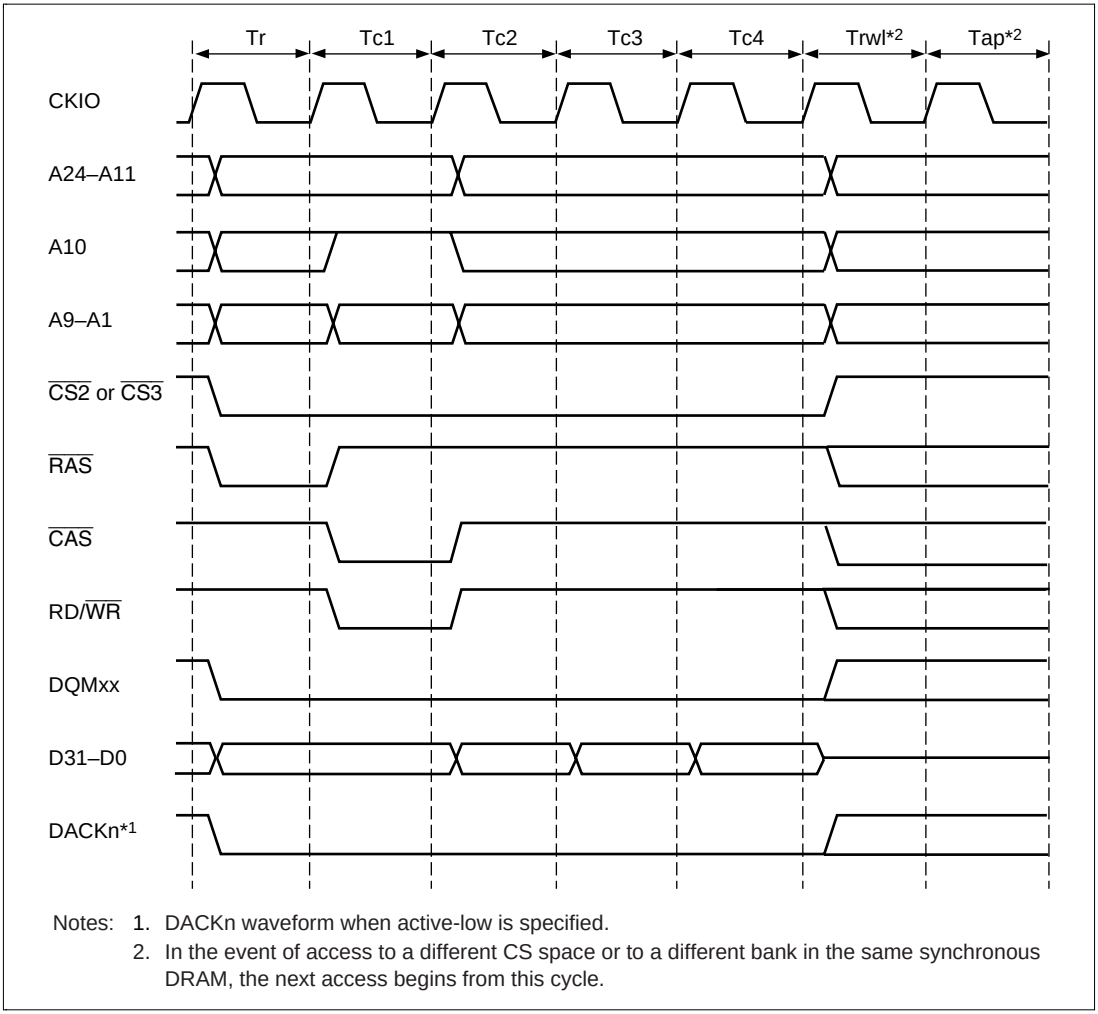
Burst write mode can be selected by setting the BWE bit to 1 in BCR3. The basic timing chart for burst write access is shown in figure 7.24. This example assumes a 32-bit bus width and a burst length of 4. In the burst write cycle, the WRITA command that performs auto-precharge is issued in Tc1 following the ACTV command Tr cycle. The first 4 bytes of write data are output simultaneously with the WRITA command in Tc, and the remaining 12 bytes of data are output consecutively in Tc2, Tc3, and Tc4. In a write with auto-precharge, as with a single write, when the same bank is accessed, a Trw1 cycle that provides the waiting time until precharge is started is inserted after output of the write data, followed by a Tap cycle for the precharge wait in a write access. Access is possible to a different CS space or to a different bank in the same synchronous DRAM. The Trw1 and Tap cycles can be set respectively in MCR by bits TRWL1 and TRWL0, and bits TRP1 and TRP0.

The specification in bits TRWL1 and TRWL0 is the number of cycles from Tc4 to Tap. Therefore, when TRWL1—TRWL0 = 00, the number of Trwl cycles is 0.

When a single write is performed in burst write mode, the synchronous DRAM setting is for a burst length of 4. After data is written in Tc1, empty writes are performed in Tc2, Tc3, and Tc4 by driving the DQMxx signal high.

These empty cycles increase the memory access time and tend to reduce program execution speed and DMA transfer speed. Therefore, unnecessary cache-through area accesses should be avoided, and copy-back should be selected for the cache setting. Also, in DMA transfer, it is important to use a data structure that allows transfer in 16-bit units.

When clock ratio Iø:Eø = 1:1, a dummy cycle is inserted after Tc4.




Notes: 1. DACKn waveform when active-low is specified.
2. In the event of access to a different CS space or to a different bank in the same synchronous DRAM, the next access begins from this cycle.

**Figure 7.24 Basic Burst Write Timing (Auto-Precharge)**

### 7.5.7 Bank Active Function

A synchronous DRAM bank function is used to support high-speed accesses of the same row address. When the RASD bit in MCR is set to 1, read/write accesses are performed using commands without auto-precharge (READ, WRIT). In this case, even when the access is completed, no precharge is performed. When accessing the same row address in the same bank, a READ or WRIT command can be called immediately without calling an ACTV command, just like the RAS down mode of the DRAM's high-speed page mode. Synchronous DRAM is divided into two banks, so one row address in each can stay active. When the next access is to a different row address, a PRE command is called first to precharge the bank, and access is performed by an ACTV command and READ or WRIT command in order, after the precharge is completed. With successive accesses to different row addresses, the precharge is performed after the access request occurs, so the access time is longer. When writing, performing an auto-precharge means that no command can be called for $t_{RWL} + t_{AP}$ cycles after a WRITA command is called. When the bank active mode is used, READ or WRIT commands can be issued consecutively if the row address is the same. This shortens the number of cycles by $t_{RWL} + t_{AP}$ for each write. The number of cycles between the issue of the precharge command and the row address strobe command is determined by the TRP1, TRP0 in MCR.

Whether execution is faster when the bank active mode is used or when basic access is used is determined by the proportion of accesses to the same row address (P1) and the average number of cycles from the end of one access to the next access ($t_A$). When $t_A$ is longer than $t_{AP}$, the delay waiting for the precharge during a read becomes invisible. If $t_A$ is longer than $t_{RWL} + t_{AP}$, the delay waiting for the precharge also becomes invisible during writes. The difference between the bank active mode and basic access speeds in these cases is the number of cycles between the start of access and the issue of the read/write command: $(t_{RP} + t_{RCD}) \times (1 - P1)$ and $t_{RCD}$, respectively.

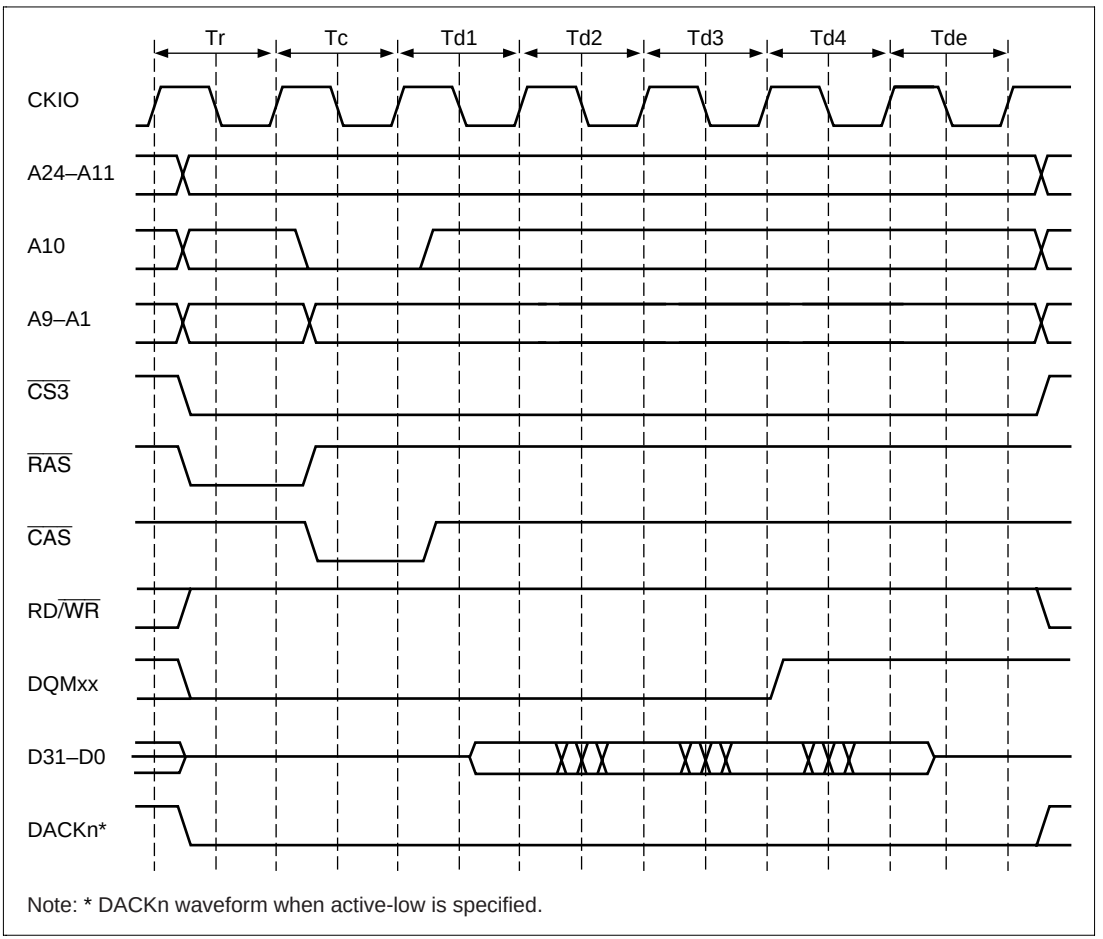
The time that a bank can be kept active, $t_{RAS}$, is limited. When the period will be provided by program execution, and it is not assured that another row address will be accessed without a hit to the cache, the synchronous DRAM must be set to auto-refresh and the refresh cycle must be set to the maximum value $t_{RAS}$ or less. This enables the limit on the maximum active period for each bank to be ensured. When auto-refresh is not being used, some measure must be taken in the program to ensure that the bank does not stay active for longer than the prescribed period.

Figure 7.25 shows a burst read cycle that is not an auto-precharge cycle, figure 7.26 shows a burst read cycle to a same row address, figure 7.27 shows a burst read cycle to different row addresses, figure 7.28 shows a write cycle without auto-precharge, figure 7.29 shows a write cycle to a same row address, and figure 7.30 shows a write cycle to different row addresses.

In figure 7.26, a cycle that does nothing, Tnop, is inserted before the Tc cycle that issues the READ command. Synchronous DRAMs have a 2 cycle latency during reads for the DQMxx signals that specify bytes. If the Tc cycle is performed immediately without inserting a Tnop cycle, the DQMxx signal for the Td1 cycle data output cannot be specified. This is why the Tnop

cycle is inserted. When the CAS latency is 2 or more, however, the Tnop cycle is not inserted so that timing requirements will be met even when a DQMxx signal is set after the Tc cycle.

When the bank active mode is set, the access will start with figure 7.25 or figure 7.28 and repeat figure 7.26 or figure 7.29 for as long as the same row address continues to be accessed when only accesses to the respective banks of the CS3 space are considered. Accesses to other CS spaces during this period do not affect this operation. When an access occurs to a different row address while the bank is active, figure 7.27 or figure 7.30 will be substituted for figures 7.26 and 7.29 after this is detected. Both banks will become inactive even in the bank active mode after the refresh cycle ends or after the bus is released by bus arbitration.




Note: * DACKn waveform when active-low is specified.

**Figure 7.25 Burst Read Timing (No Precharge)**

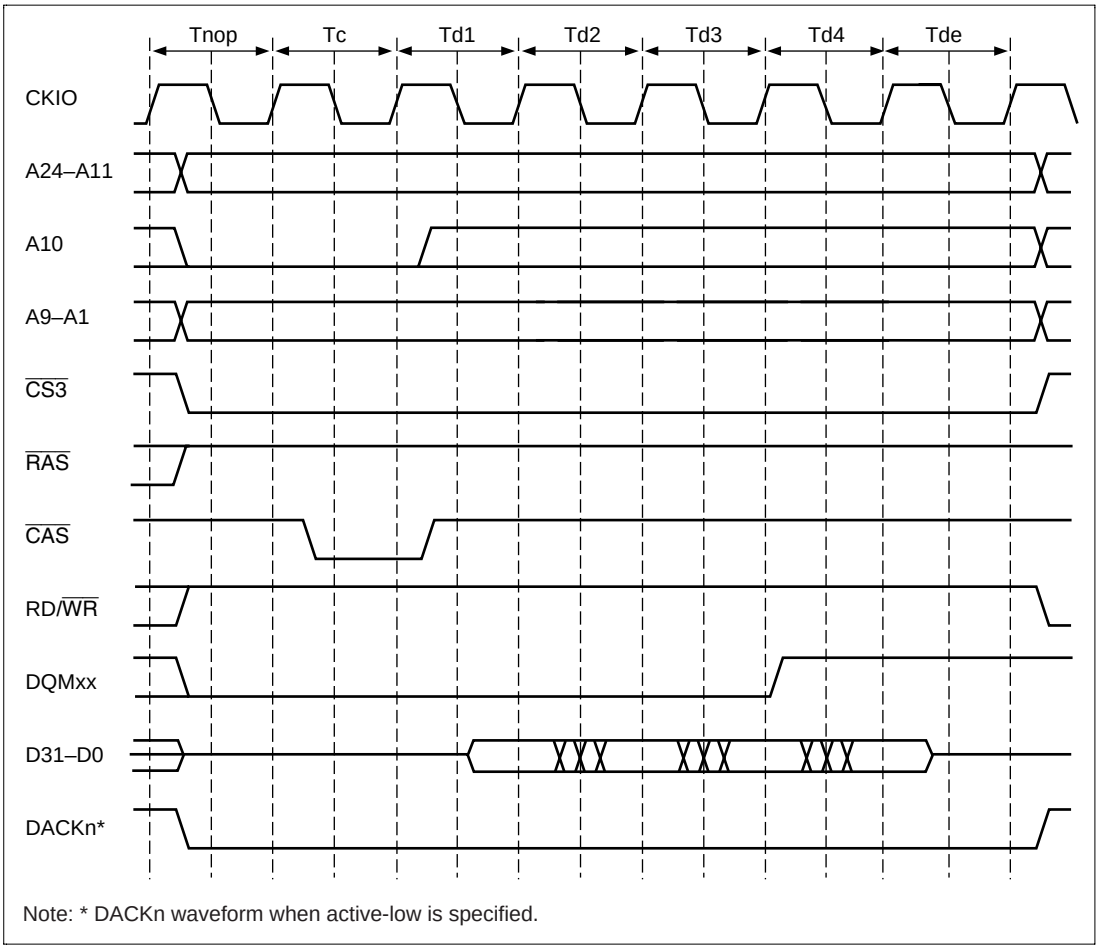



**Figure 7.26 Burst Read Timing (Bank Active, Same Row Address)**

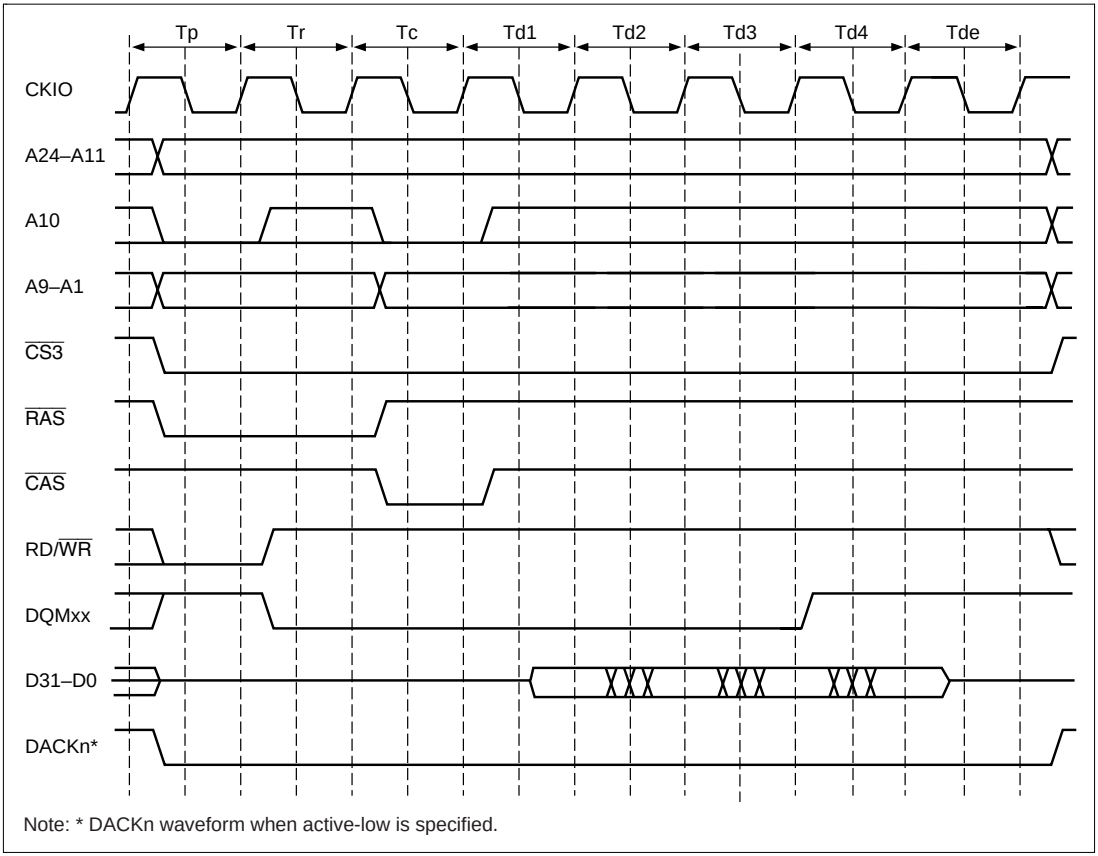
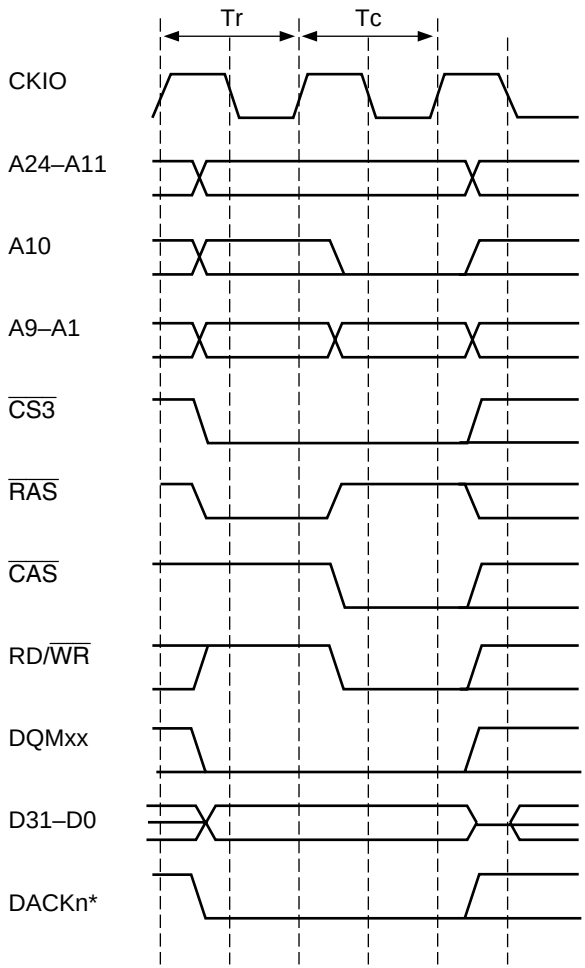
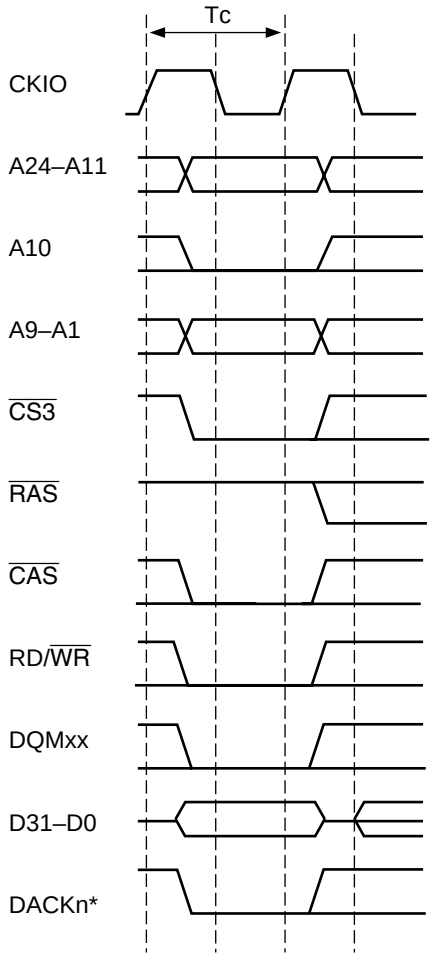



**Figure 7.27 Burst Read Timing (Bank Active, Different Row Addresses)**



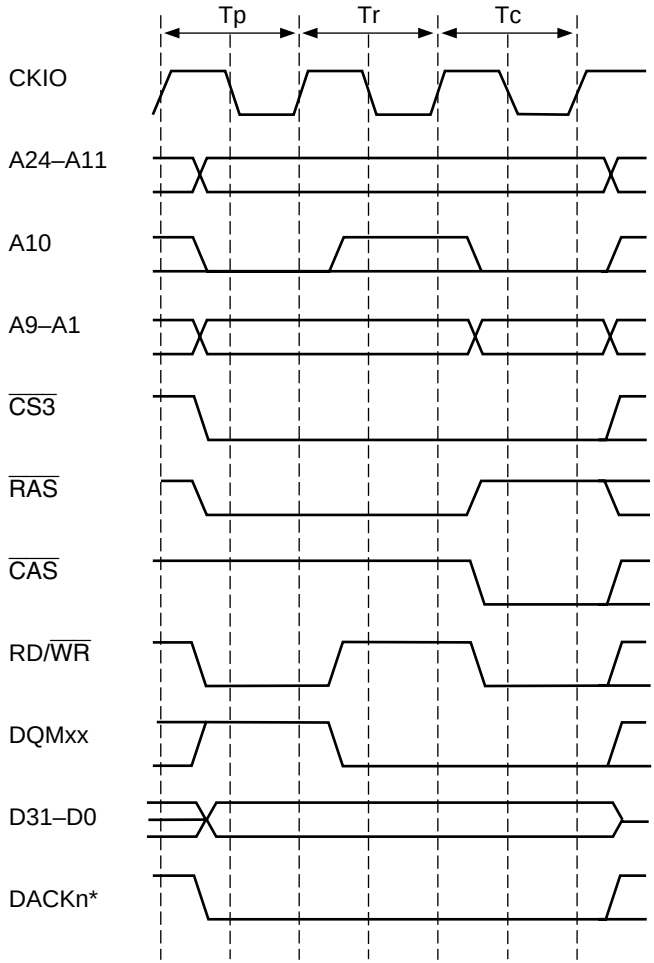

Note: * DACKn waveform when active-low is specified.

**Figure 7.28 Single Write Mode Timing (No Precharge)**




Note: * DACKn waveform when active-low is specified.

**Figure 7.29 Single Write Mode Timing (Bank Active, Same Row Address)**




Note: * DACKn waveform when active-low is specified.

**Figure 7.30 Single Write Mode Timing (Bank Active, Different Row Addresses)**

### 7.5.8 Refreshes

The bus state controller is equipped with a function to control refreshes of synchronous DRAM. Auto-refreshes can be performed by setting the RMODE bit to 0 and the RFSH bit to 1 in MCR. Consecutive refreshes can also be generated by setting the RRC2–RRC0 bits in RTCSR. When the synchronous DRAM is not accessed for a long period of time, set the RFSH bit and RMODE bit both to 1 to generate self-refresh mode, which uses low power consumption to retain data.

**Auto-Refresh:** The number of refreshes set in the RRC2–RRC0 bits in RTCSR are performed at the interval determined by the input clock selected by the CKS2–CKS0 bits in RTCSR and the value set in RTCOR. Set the CKS2–CKS0 bits and RTCOR so that the refresh interval specifications of the synchronous DRAM being used are satisfied. First , set RTCOR, RTCNT, and the RMODE and RFSH bits in MCR, then set the CKS2–CKS0 and RRC2–RRC0 bits in RTCSR. When a clock is selected with the CKS2–CKS0 bits, RTCNT starts counting up from the value at that time. The RTCNT value is constantly compared to the RTCOR value, and when the two values match, a refresh request is made, and the number of auto-refreshes set in RRC2–RRC0 are performed. RTCNT is cleared to 0 at that time and the count up starts again. Figure 7.30 shows the timing for the auto-refresh cycle.

First, a PALL command is issued during the Tp cycle to change all the banks from active to precharge states. Then number of idle cycles equal to one less than the value set in TRP1 and TRP0 are inserted, and a REF command is issued in the Trr cycle. After the Trr cycle, no new commands are output for the number of cycles specified in the TRAS bit in MCR. The TRAS bit must be set to satisfy the refresh cycle time specifications (active/active command delay time) of the synchronous DRAM. When the set value of the TRP1 and TRP0 bits in MCR is 2 or more, an NOP cycle is inserted between the Tp cycle and Trr cycle.

During a manual reset, no refresh request is issued, since there is no RTCNT count-up. To perform a refresh properly, make the manual reset period shorter than the refresh cycle interval and set RTCNT to (RTCOR – 1) so that the refresh is performed immediately after the manual reset is cleared.

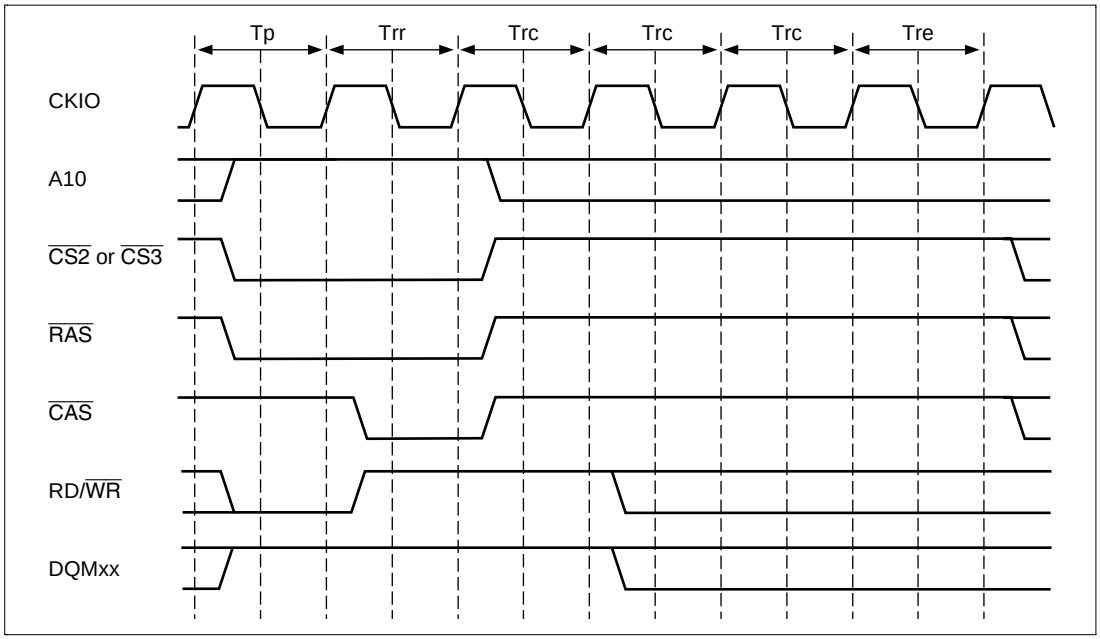



**Figure 7.31 Auto-Refresh Timing**

**Self-Refreshes:** The self-refresh mode is a type of standby mode that produces refresh timing and refresh addresses within the synchronous DRAM. It is started up by setting the RMODE and RFSH bits to 1. The synchronous DRAM is in self-refresh mode when the CKE signal level is low. During the self-refresh, the synchronous DRAM cannot be accessed. Also, bus arbitration is not performed in the self-refresh state. To clear the self-refresh, set the RMODE bit to 0. After self-refresh mode is cleared, issuing of commands is prohibited for the number of cycles specified in the TRAS1 and TRAS0 bits in MCR. Figure 7.32 shows the self-refresh timing. Settings must be made so that self-refresh clearing and data retention are performed correctly, and auto-refreshing is performed without delay at the correct intervals. When self-refresh mode is entered while the synchronous DRAM is set for auto-refresh or when leaving the standby mode with a manual reset or NMI, auto-refresh can be re-started if RFSH is 1 and RMODE is 0 when the self-refresh mode is cleared. When time is required between clearing the self-refresh mode and starting the auto-refresh mode, this time must be reflected in the initial RTCNT setting. When the RTCNT value is set to RTCOR – 1, the refresh can be started immediately.

If the standby function of the chip is used to enter the standby mode after the self-refresh mode is set, the self-refresh state continues; the self-refresh state will also be maintained after returning from a standby using an NMI. A manual reset cannot be used to exit the self-refresh state either.

During a power-on reset, the bus state controller register is initialized, so the self-refresh state is ended.

**Refresh Requests and Bus Cycle Requests:** When a refresh request occurs while a bus cycle is executing, the refresh will not be executed until the bus cycle is completed. When a refresh request occurs while the bus is released using the bus arbitration function, the refresh will not be executed until the bus is recaptured. In the SH7612, the REFOUT pin is provided to send a signal requesting the bus right during the wait for refreshing to be executed. REFOUT is asserted until the bus is acquired. If RTCNT and RTCOR match and a new refresh request occurs while waiting for the refresh to execute, the previous refresh request is erased. To make sure the refresh executes properly, be sure that the bus cycle and bus capture do not exceed the refresh interval.

If a bus arbitration request occurs during a self-refresh, the bus is not released until the self-refresh is cleared.

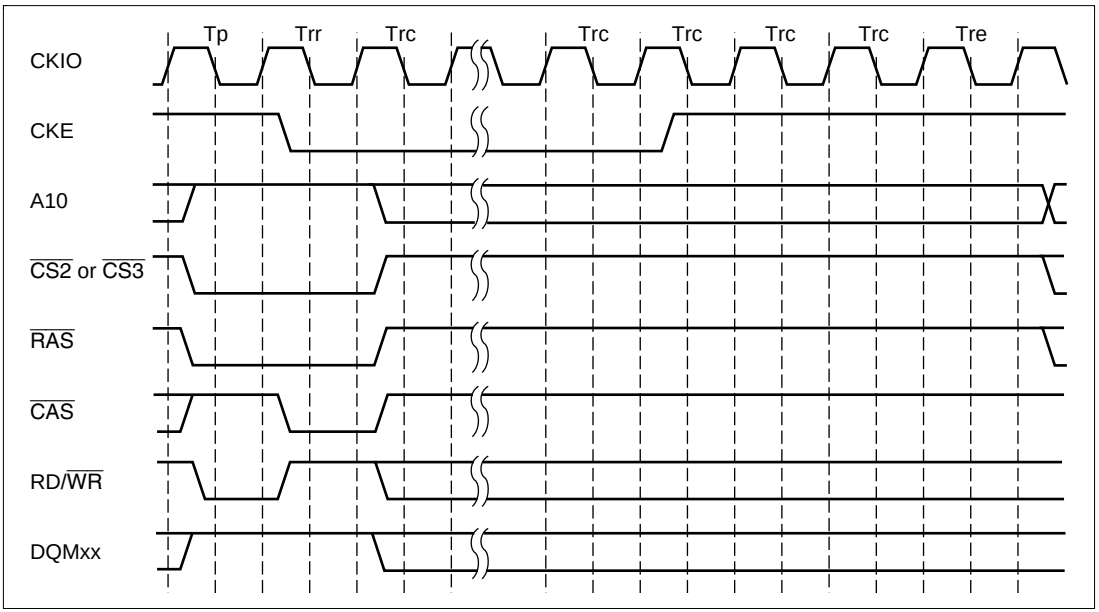



**Figure 7.32 Self-Refresh Timing**

### 7.5.9 Power-On Sequence

To use synchronous DRAM, the mode must first be set after the power is turned on. To properly initialize the synchronous DRAM, the synchronous DRAM mode register must be written to after the registers of the bus state controller have first been set. The synchronous DRAM mode register is set using a combination of the $\overline{CS2}$ or $\overline{CS3}$ signal and the $\overline{RAS}$, $\overline{CAS}$/$\overline{OE}$, and RD/$\overline{WR}$ signals. They fetch the value of the address signal at that time. If the value to be set is X, the bus state controller operates by writing to address X + H'FFFF0000 or X + H'FFFF8000 from the CPU, which allows the value X to be written to the synchronous DRAM mode register. Whether X + H'FFFF0000 or X + H'FFFF8000 is used depends on the specifications of the synchronous DRAM. Use a value in the range H'000 to H'FFF for X. Data is ignored at this time, but the mode is written using word as the size.

Write any data in word size to the following addresses to select the burst read single write supported by the chip, a CAS latency of 1 to 3, a sequential wrap type, and a burst length of 8 or 4 (depending on whether the width is 16 bits or 32 bits).

- Burst Read/Single Write

| | | | |
|---|---|---|---|
| For 16 bits: | CAS latency 1 | H'FFFF0426 | (H'FFFF8426) |
| | CAS latency 2 | H'FFFF0446 | (H'FFFF8446) |
| | CAS latency 3 | H'FFFF0466 | (H'FFFF8466) |
| For 32 bits: | CAS latency 1 | H'FFFF0848 | (H'FFFF8848) |
| | CAS latency 2 | H'FFFF0888 | (H'FFFF8888) |
| | CAS latency 3 | H'FFFF08C8 | (H'FFFF88C8) |

To set burst read, burst write, CAS latency 1 to 3, wrap-type sequential, and burst length 8 or 4 (depending on whether the width is 16 bits or 32 bits), arbitrary data is written to the following addresses, using the word size.

- Burst Read/Burst Write

| | | | |
|---|---|---|---|
| 16-bit width: | CAS latency 1 | H'FFFF0026 | (H'FFFF8026) |
| | CAS latency 2 | H'FFFF0046 | (H'FFFF8046) |
| | CAS latency 3 | H'FFFF0066 | (H'FFFF8066) |
| 32-bit width: | CAS latency 1 | H'FFFF0048 | (H'FFFF8048) |
| | CAS latency 2 | H'FFFF0088 | (H'FFFF8088) |
| | CAS latency 3 | H'FFFF00C8 | (H'FFFF80C8) |

Figure 7.33 shows the mode register setting timing.

Writing to address X + H'FFFF0000 or X + H'FFFF8000 first issues an all-bank precharge command (PALL), then issues eight dummy auto-refresh commands (REF) required for the synchronous DRAM power-on sequence. Lastly, a mode register write command (MRS) is issued. Three idle cycles are inserted between the all-bank precharge command and the first auto-refresh command, and eight idle cycles between auto-refresh commands, and between the eighth auto-refresh command and the mode register write command, regardless of the MCR setting.

After writing to the synchronous DRAM mode register, perform a dummy read to each synchronous DRAM bank before starting normal access. This will initialize the SH7612's internal address comparator.

Synchronous DRAM requires a fixed idle time after powering on before the all-bank precharge command is issued. Refer to the synchronous DRAM manual for the necessary idle time. When the pulse width of the reset signal is longer than the idle time, the mode register may be set immediately without problem. However, care is required if the pulse width of the reset signal is shorter than the idle time.

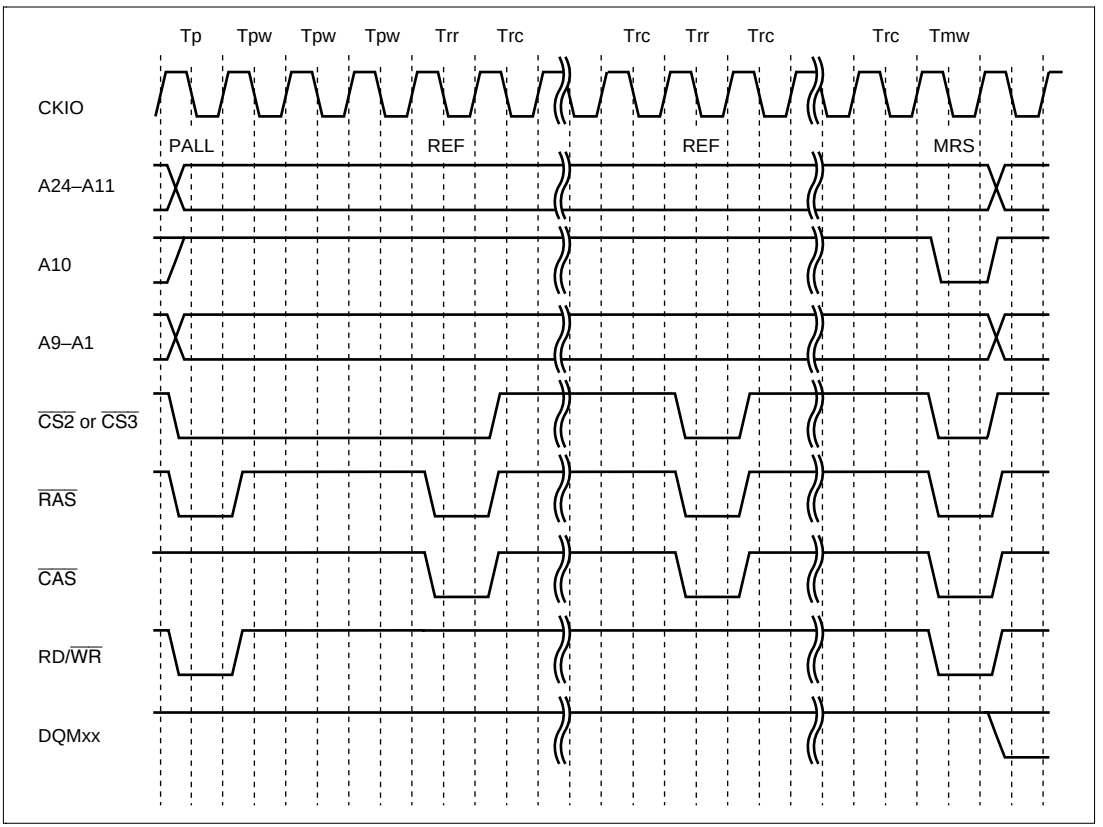



**Figure 7.33 Synchronous DRAM Mode Write Timing**

## 7.6 DRAM Interface

### 7.6.1 DRAM Direct Connection

When the DRAM and other memory enable bits (DRAM2–DRAM0) in BCR1 are set to 010, the CS3 space becomes DRAM space, and a DRAM interface function can be used to directly connect DRAM.

The data width of an interface can be 16 or 32 bits (figures 7.34 and 7.35). Two-CAS 16-bit DRAMs can be connected, since $\overline{\text{CAS}}$ is used to control byte access. The $\overline{\text{RAS}}$, $\overline{\text{CAS3}}$, $\overline{\text{CAS2}}$, $\overline{\text{CAS1}}$, $\overline{\text{CAS0}}$, and RD/$\overline{\text{WR}}$ signals are used to connect the DRAM. When the data width is 16 bits, $\overline{\text{CAS3}}$, and $\overline{\text{CAS2}}$ are not used. In addition to ordinary read and write access, burst access using high-speed page mode is also supported.

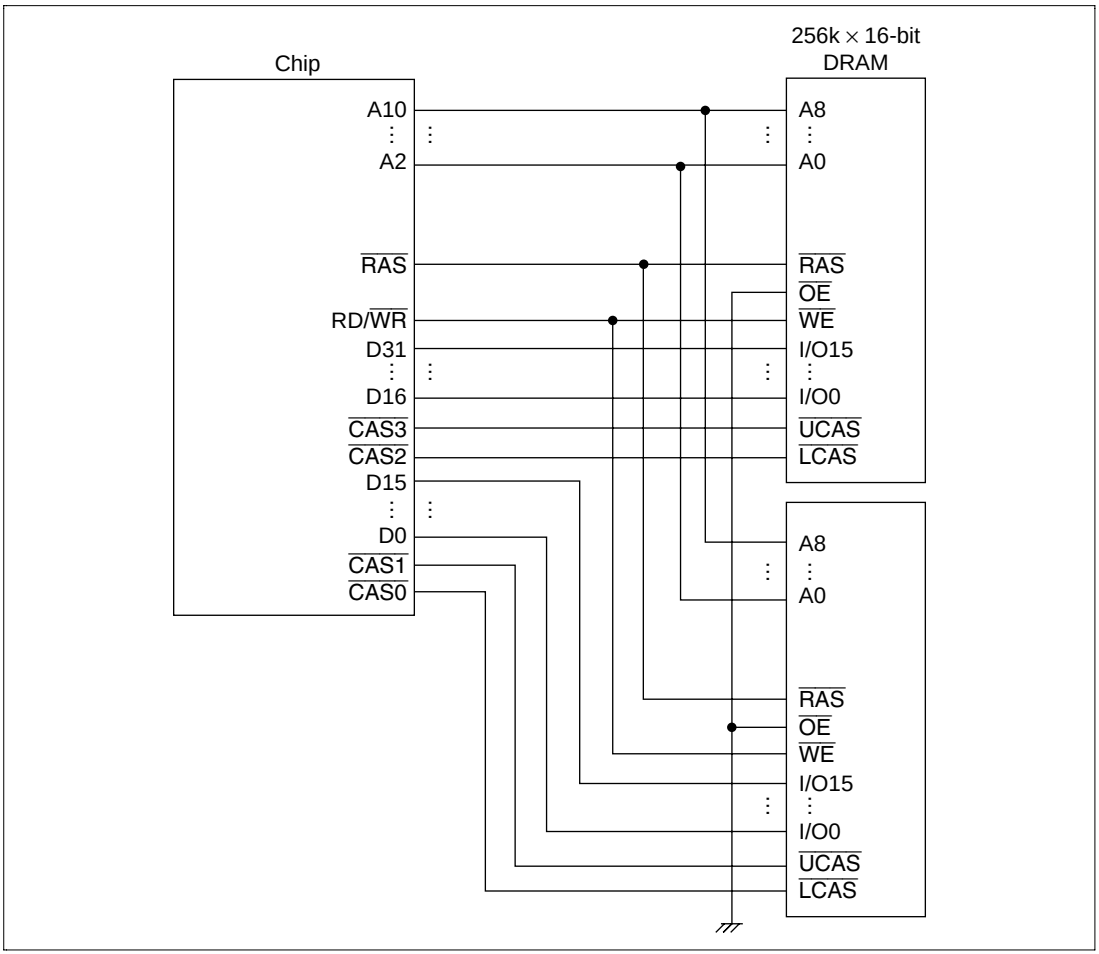



**Figure 7.34 Example of DRAM Connection (32-Bit Data Width)**

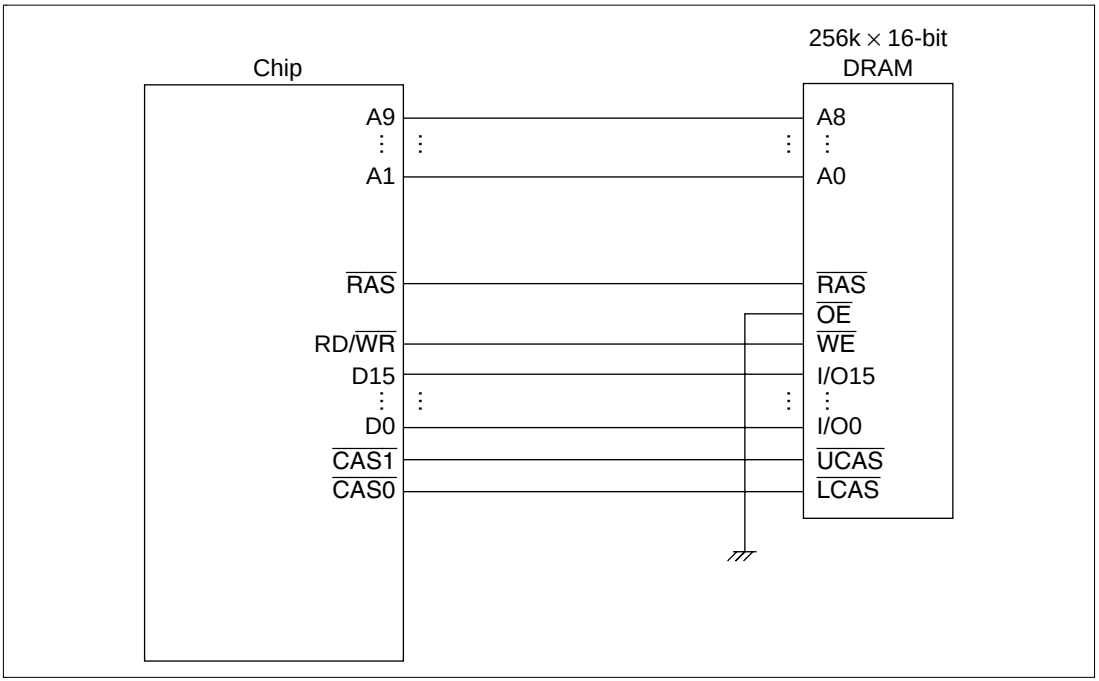



**Figure 7.35 Example of DRAM Connection (16-Bit Data Width)**

### 7.6.2 Address Multiplexing

When the CS3 space is set to DRAM, addresses are always multiplexed. This allows DRAMs that require multiplexing of row and column addresses to be connected directly without additional address multiplexing circuits. There are four ways of multiplexing, which can be selected using the AMX1–AMX0 bits in MCR. Table 7.7 illustrates the relationship between the AMX1–AMX0 bits and address multiplexing. Address multiplexing is performed on address output pins A15–A1. The original addresses are output to pins A24–A16. During DRAM accesses, AMX2 is reserved, so set it to 0.

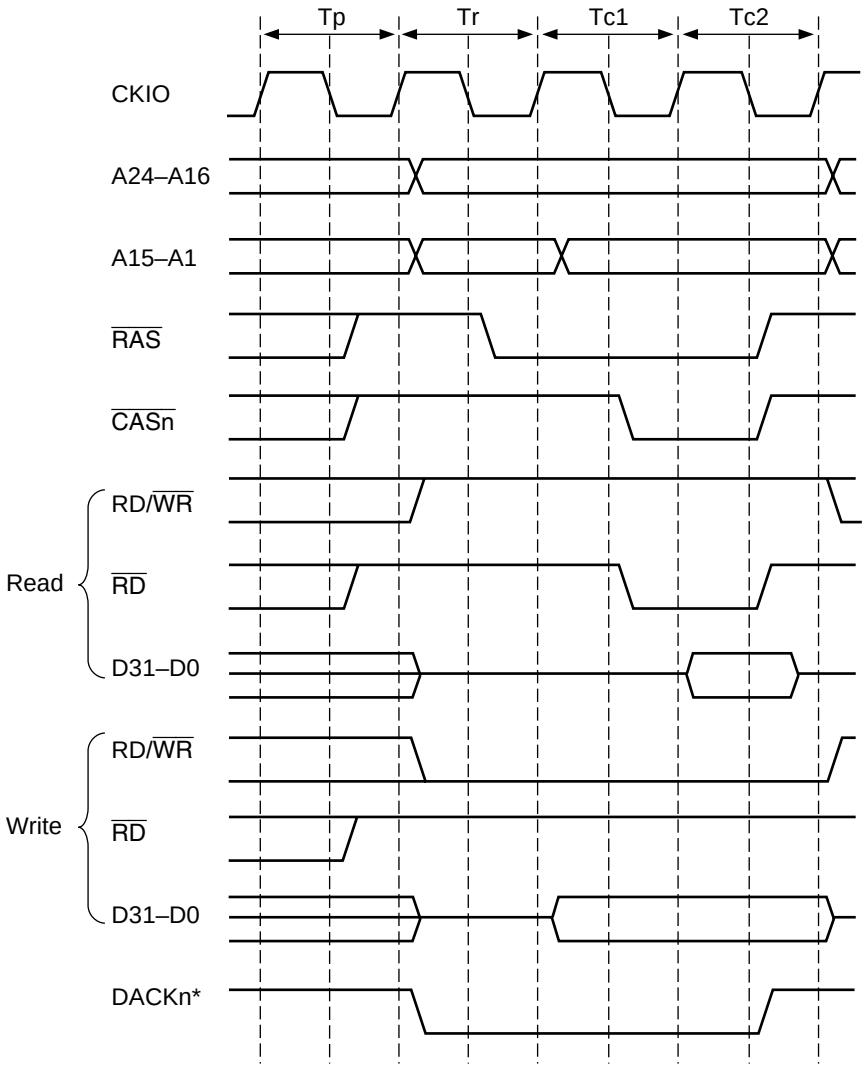
**Table 7.7 Relationship between AMX1–AMX0 and Address Multiplexing**

| AMX1 | AMX0 | No. of Column Address Bits | Row Address Output | Column Address Output |
|---|---|---|---|---|
| 0 | 0 | 8 bits | A23–A9 | A15–A1 |
| | 1 | 9 bits | A24–A10 | A15–A1 |
| 1 | 0 | 10 bits | A24–A11*[1] | A15–A1 |
| | 1 | 11 bits | A24–A12*[2] | A15–A1 |

Notes: 1. Address output pin A15 is high.
2. Address output pins A15 and A14 are high.

### 7.6.3 Basic Timing

The basic timing of a DRAM access is 3 cycles. Figure 7.36 shows the basic DRAM access timing. Tp is the precharge cycle, Tr is the RAS assert cycle, Tc1 is the CAS assert cycle, and Tc2 is the read data fetch cycle. When accesses are consecutive, the Tp cycle of the next access overlaps the Tc2 cycle of the previous access, so accesses can be performed in a minimum of 3 cycles each.




Note: * DACKn waveform when active-low is specified.

**Figure 7.36 Basic Access Timing**

### 7.6.4 Wait State Control

When the clock frequency is raised, 1 cycle may not always be sufficient for all states to end, as in basic access. Setting bits in WCR1, WCR2 and MCR enables the state to be lengthened. Figure 7.37 shows an example of lengthening a state using settings.

The Tp cycle (which ensures a sufficient RAS precharge time) can be extended from 1 cycle to 4 cycles by insertion of a Tpw cycle by means of the TRP1, TRP0 bit in MCR. The number of cycles between RAS assert and CAS assert can be extended from 1 cycle to 3 cycles by inserting a Trw cycle by means of the RCD1, RCD0 bit in MCR. The number of cycles from CAS assert to the end of access can be extended from 1 cycle to 3 cycles by setting the W31/W30 bits in WCR1. When external wait mask bit A3WM in WCR2 is cleared to 0 and bits W31 and W30 in WCR1 are set to a value other than 00, the external wait pin is also sampled, so the number of cycles can be further increased. When bit A3WM in WCR2 is set to 1, external wait input is ignored regardless of the setting of W31 and W30 in WCR1. Figure 7.38 shows the timing of wait state control using the $\overline{\text{WAIT}}$ pin.

In either case, when consecutive accesses occur, the Tp cycle access overlaps the Tc2 cycle of the previous access. In DRAM access, $\overline{\text{BS}}$ is not asserted, and so $\overline{\text{RAS}}$, $\overline{\text{CAS}}$n, $\overline{\text{RD}}$, etc., should be used for $\overline{\text{WAIT}}$ pin control.

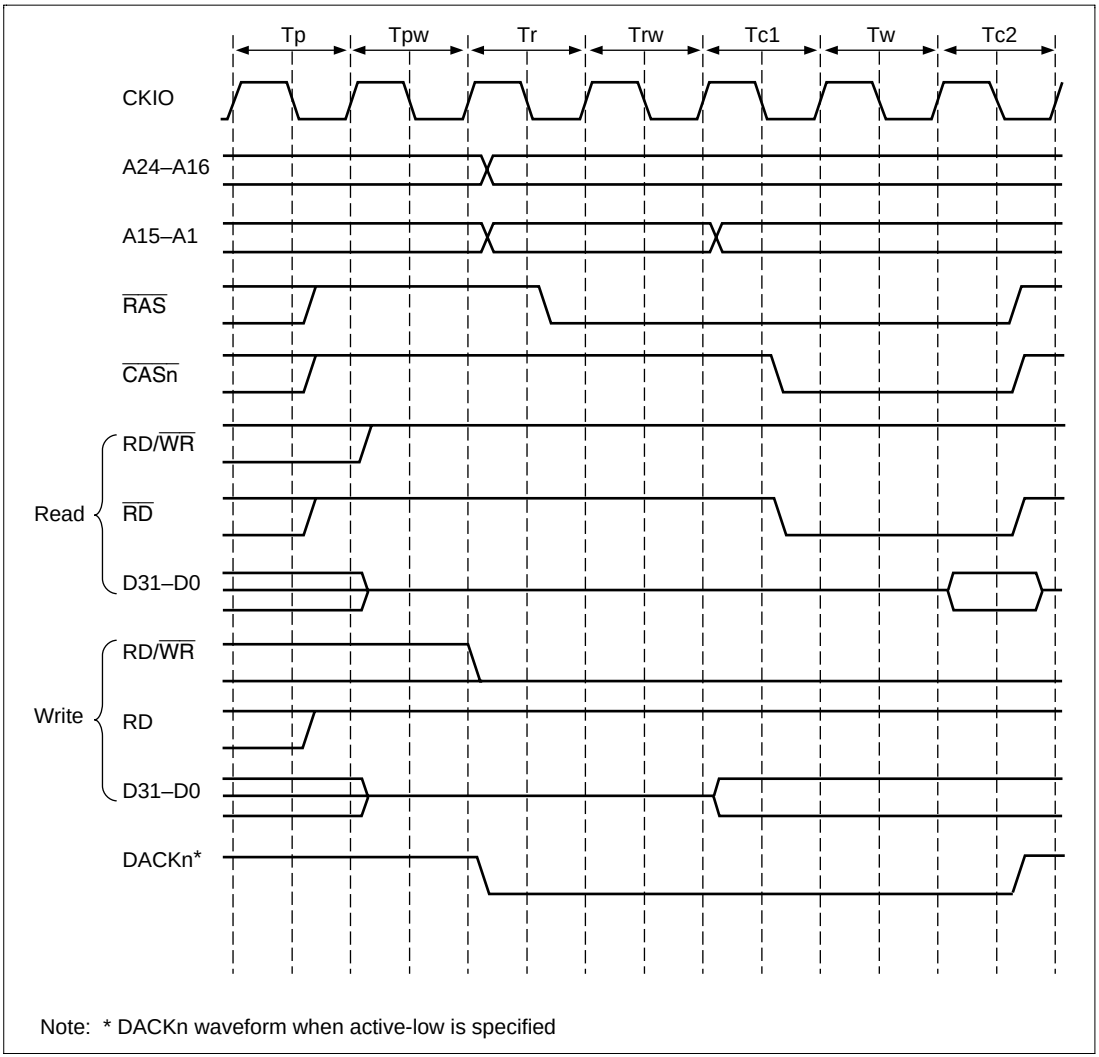



**Figure 7.37 Wait State Timing**

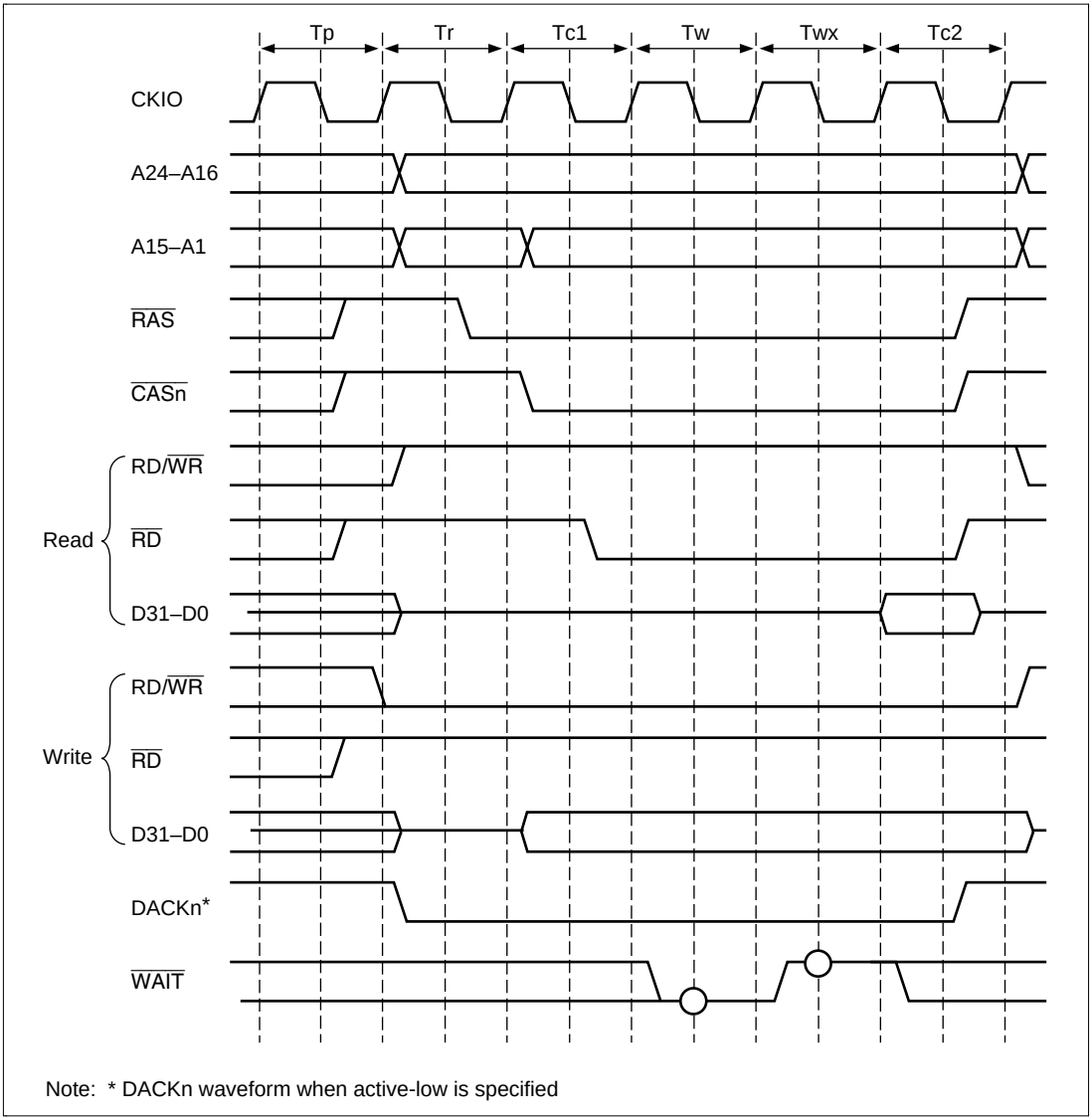



**Figure 7.38 External Wait State Timing**

### 7.6.5 Burst Access

In addition to the ordinary mode of DRAM access, in which row addresses are output at every access and data is then accessed, DRAM also has a high-speed page mode for use when continuously accessing the same row that enables fast access of data by changing only the column address after the row address is output. Select ordinary access or high-speed page mode by setting the burst enable bit (BE) in MCR. Figure 7.39 shows the timing of burst access in high-speed page mode. When performing burst access, cycles can be inserted using the wait state control function.

An address comparator is provided to detect matches of row addresses in burst mode. When this function is used and the BE bit in MCR is set to 1, setting the MCR's RASD bit (which specifies RAS down mode) to 1 places the SH7612 in RAS down mode, which leaves the RAS signal asserted. The access timing in RAS down mode is shown in figures 7.40 and 7.41. When RAS down mode is used, the refresh cycle must be less than the maximum DRAM RAS assert time tRAS when the refresh cycle is longer than the tRAS maximum.

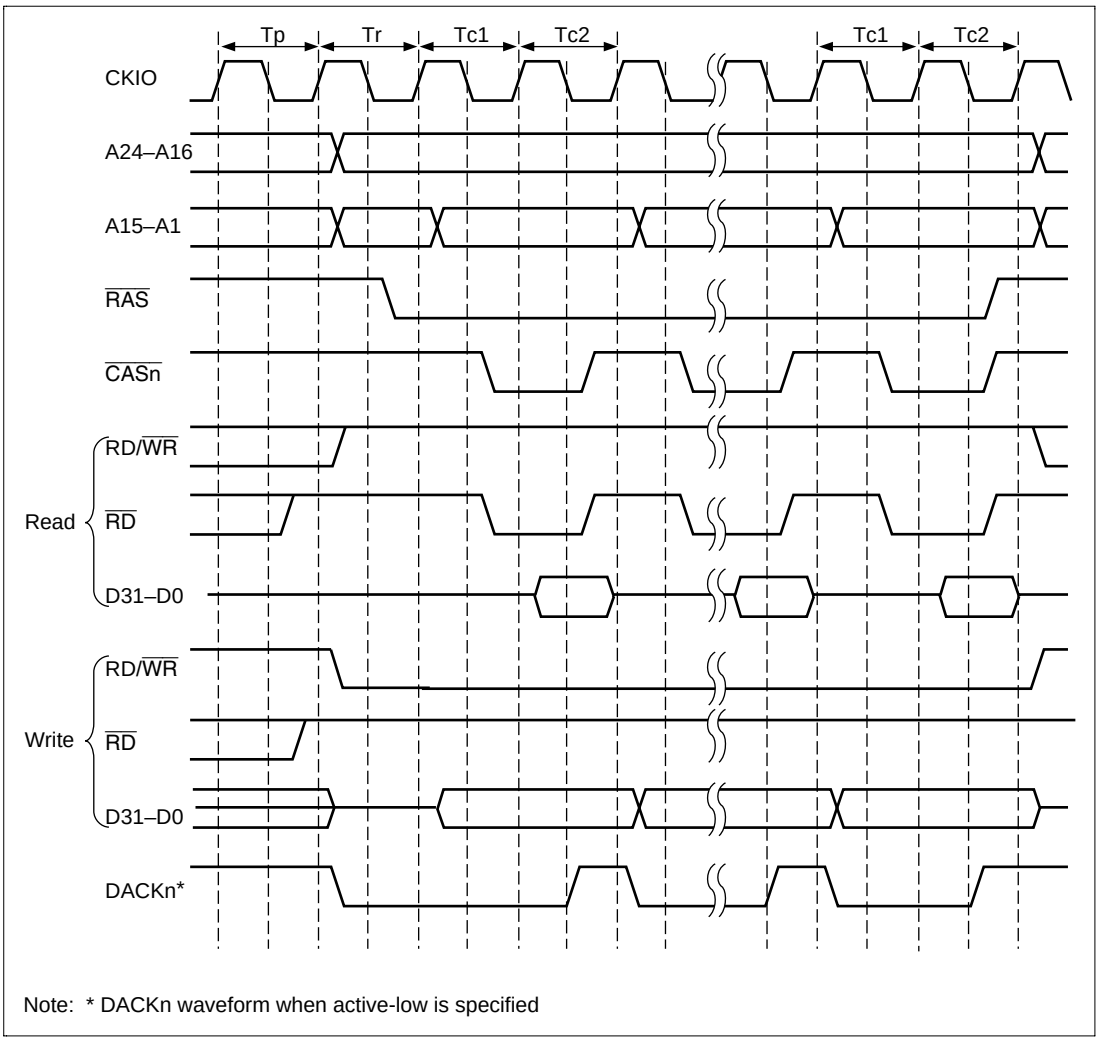
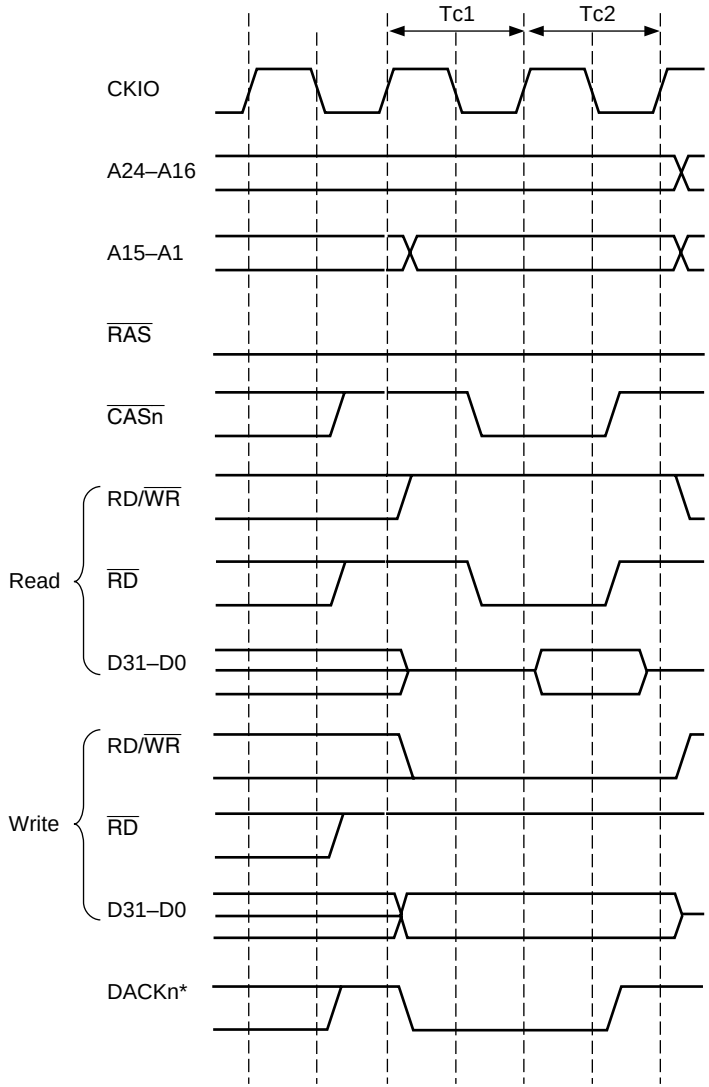



**Figure 7.39 Burst Access Timing**




Note: * DACKn waveform when active-low is specified

**Figure 7.40 RAS Down Mode Same Row Access Timing**

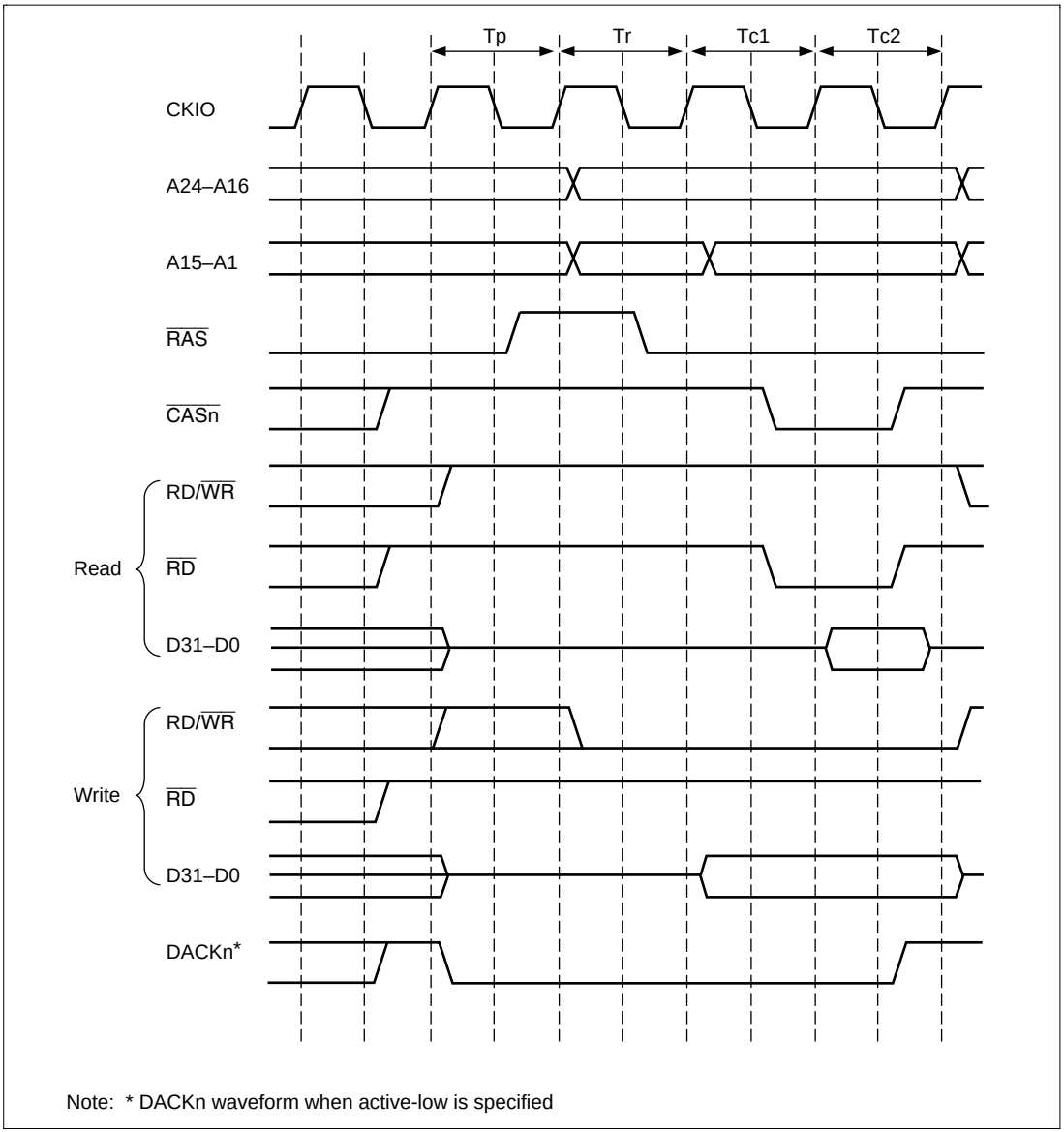



**Figure 7.41 RAS Down Mode Different Row Access Timing**

### 7.6.6 EDO Mode

In addition to the kind of DRAM in which data is output to the data bus only while the $\overline{\text{CASn}}$ signal is asserted in a data read cycle, there is another kind provided with an EDO mode in which, while both $\overline{\text{RAS}}$ and $\overline{\text{OE}}$ are asserted, once the $\overline{\text{CASn}}$ signal is asserted data is output to the data bus until $\overline{\text{CASn}}$ is next asserted, even though $\overline{\text{CASn}}$ is negated during this time.

The EDO mode bit (EDO) in MCR allows selection of ordinary access/high-speed page mode burst access or ordinary access/burst access using EDO mode. Since $\overline{OE}$ control is performed in EDO mode DRAM access, the $\overline{CAS}$ and $\overline{OE}$ pins of the SH7612 must be connected to the $\overline{OE}$ pin of the DRAM.

Ordinary access in EDO mode is shown in figure 7.44, and burst access in figure 7.45.

In EDO mode, in order to extend the timing for data output to the data bus in a read cycle until the next assertion of $\overline{CASn}$, the DRAM access time can be increased by delaying the data latch timing by 1/2 cycle, making it at the rise of the CKIO clock.

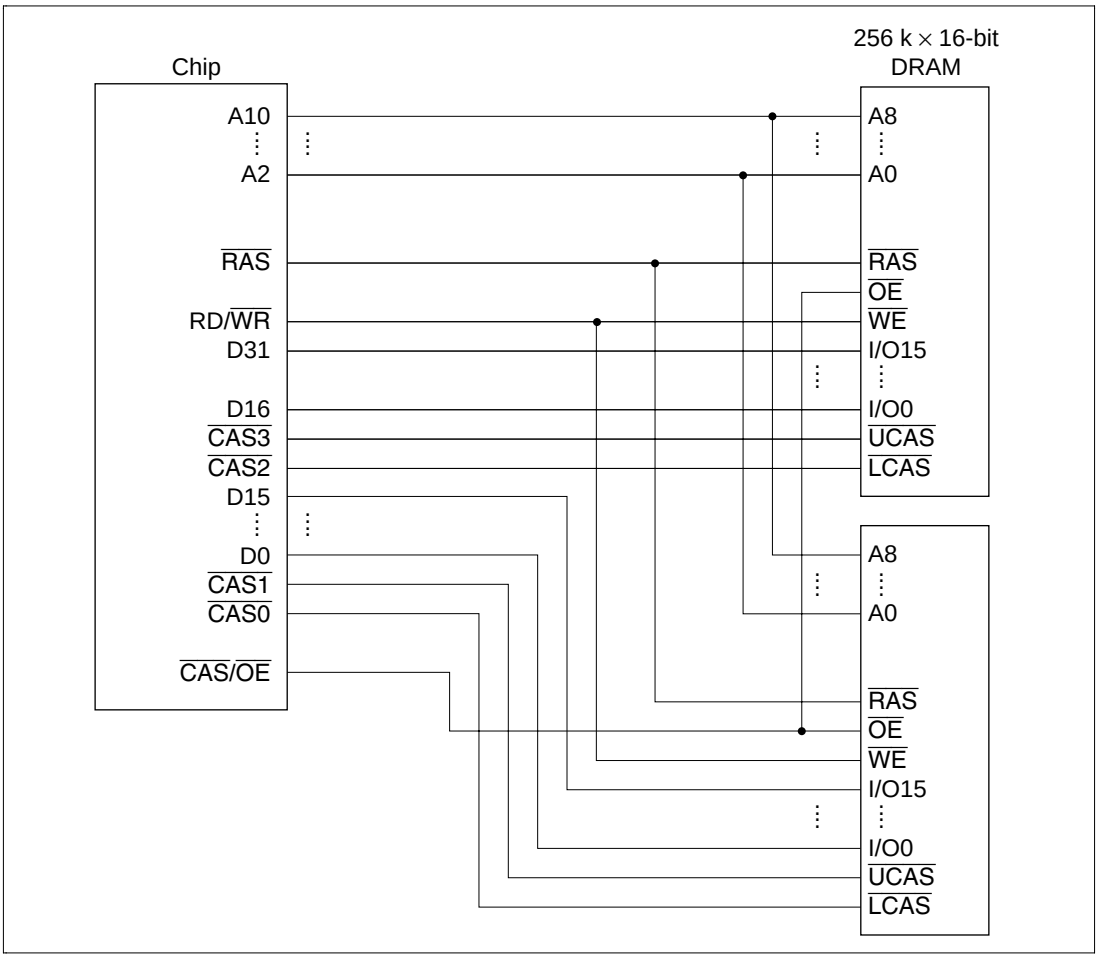



**Figure 7.42 Example of EDO DRAM Connection (32-Bit Data Width)**

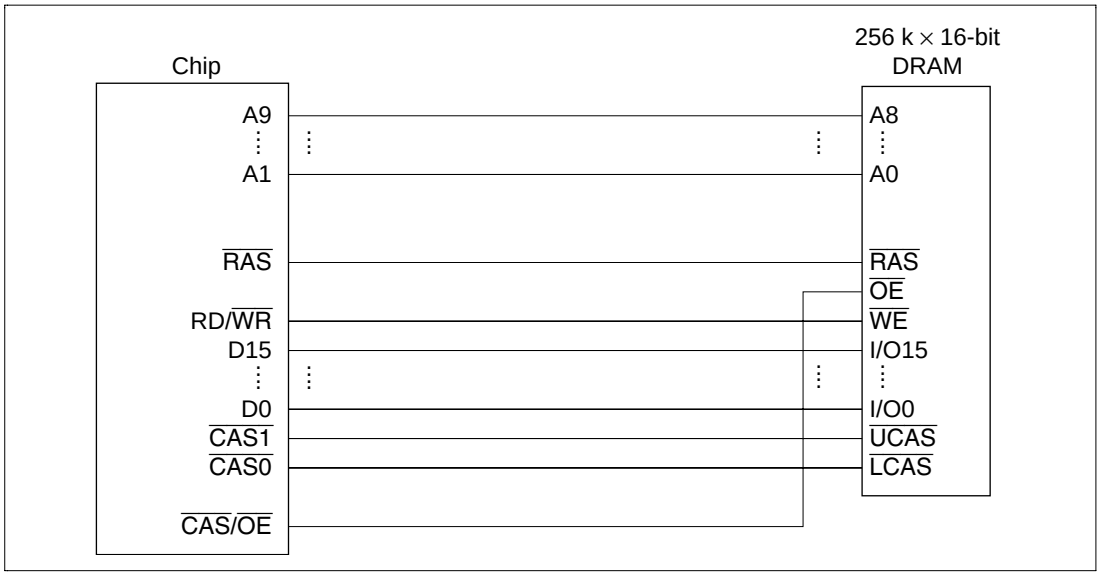



**Figure 7.43 Example of EDO DRAM Connection (16-Bit Data Width)**

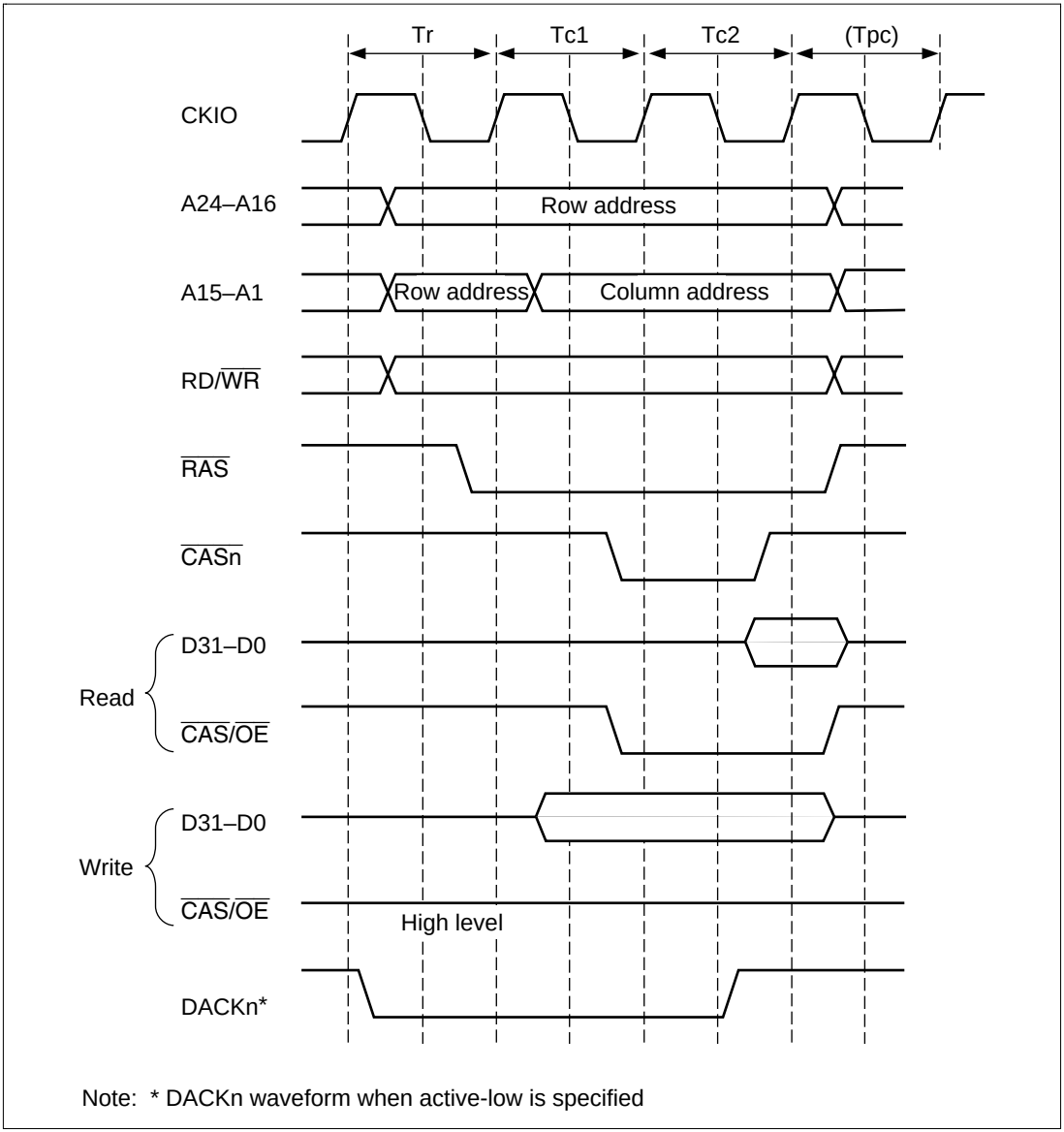



**Figure 7.44 DRAM EDO Mode Ordinary Access Timing**

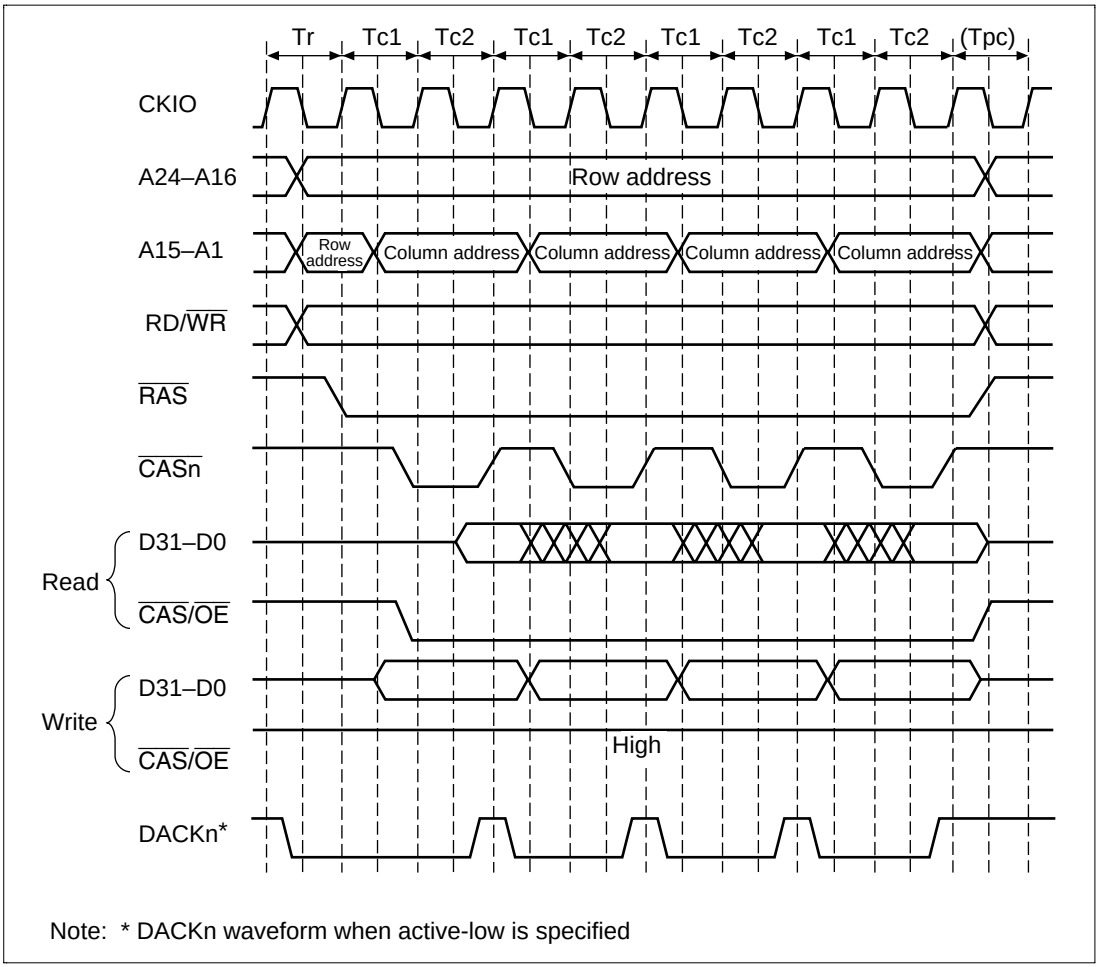
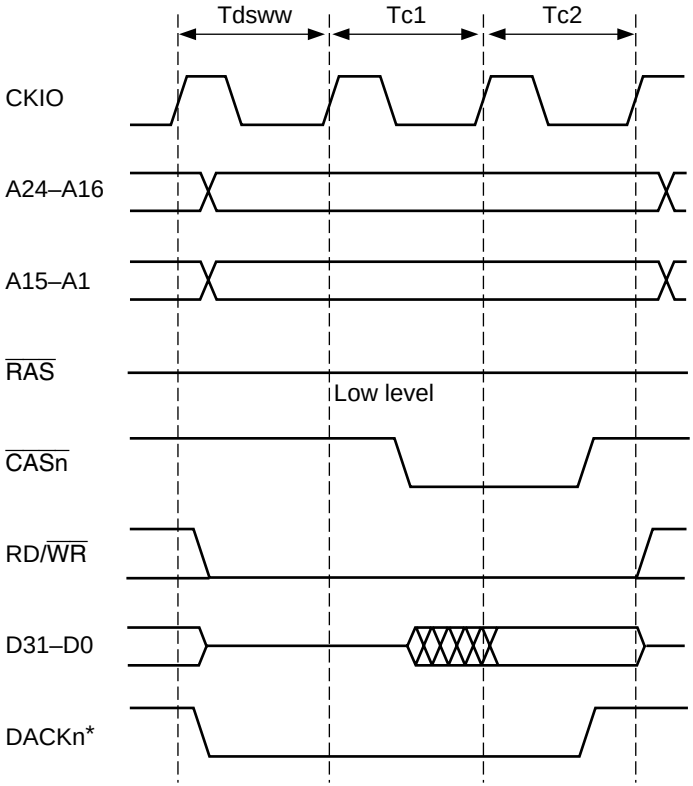



**Figure 7.45 DRAM EDO Mode Burst Access Timing**

### 7.6.7 DRAM Single Transfer

Wait states equivalent to the value set in bits DSWW1 and DSWW0 in BCR3 can be inserted between DACKn assertion and $\overline{\text{CASn}}$ assertion in a write in DMA single address transfer mode. Inserting wait states allows the data setup time for external device memory. Figure 7.46 shows the write cycle timing in DMA single transfer mode when DSWW1/DSWW0 = 01 and RASD = 1. The DMA single transfer mode read cycle is the same as a CPU or DMA dual transfer mode read cycle.




Note: * DACKn waveform when active-low is specified

**Figure 7.46 DMA Single Transfer Mode Write Cycle Timing (RAS Down Mode, Same Row Address)**

### 7.6.8 Refreshing

The bus state controller includes a function for controlling DRAM refreshing. Distributed refreshing using a CAS-before-RAS refresh cycle can be performed by clearing the RMODE bit to 0 and setting the RFSH bit to 1 in MCR. Consecutive refreshes can be generated by setting bits RRC2–RRC0 in RTCSR. If DRAM is not accessed for a long period, self-refresh mode, which uses little power consumption for data retention, can be activated by setting both the RMODE and RFSH bits to 1.

**CAS-Before-RAS Refreshing:** Refreshing is performed at intervals determined by the input clock selected by bits CKS2–CKS0 in RTCSR, and the value set in RTCOR. The RTCOR value and the value of bits CKS2–CKS0 in RTCSR should be set so as to satisfy the refresh interval specification for the DRAM used. First make the settings for RTCOR, RTCNT, and the RMODE and RFSH bits in MCR, then make the CKS2–CKS0 and RRC2–RRC0 settings in RTCSR. When

the clock is selected by CKS2–CKS0, RTCNT starts counting up from the value at that time. The RTCNT value is constantly compared with the RTCOR value, and when the two values match, a refresh request is generated and the number of CAS-before-RAS refreshes set in bits RRC2–RRC0 are performed. At the same time, RTCNT is cleared to zero and the count-up is restarted. Figure 7.47 shows the CAS-before-RAS refresh cycle timing.

The number of RAS assert cycles in the refresh cycle is specified by bits TRAS1 and TRAS0 in MCR. As with ordinary accesses, the specification of the RAS precharge time in the refresh cycle follows the setting of bits TRP1 and TRP0 in MCR.

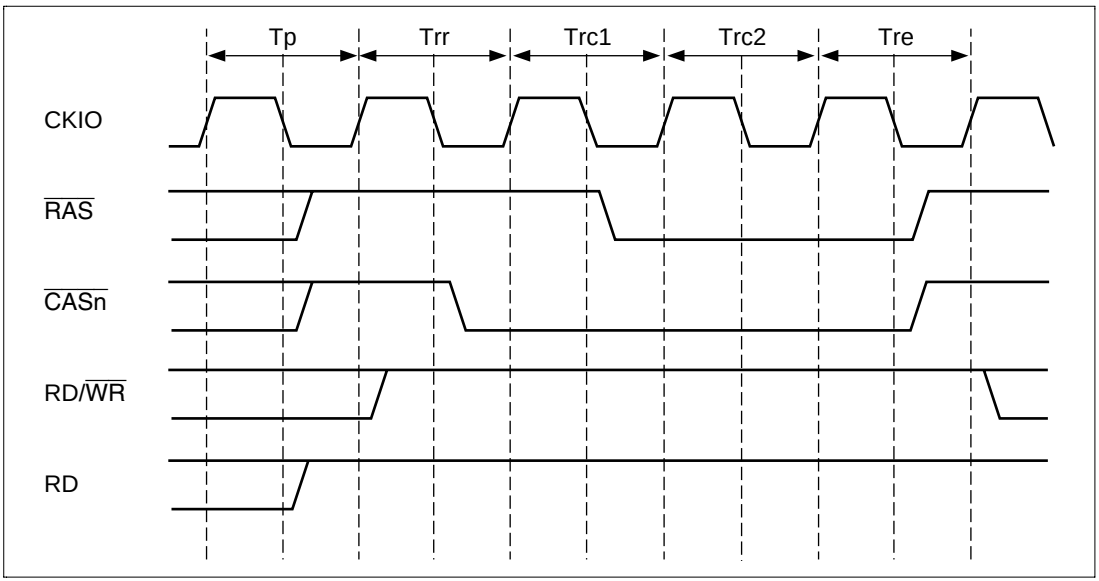



**Figure 7.47 DRAM CAS-before-RAS Refresh Cycle Timing**

**Self-Refreshing:** A self-refresh is started by setting both the RMODE bit and the RFSH bit to 1. During the self-refresh, DRAM cannot be accessed. Also, bus arbitration is not performed in the self-refresh state. Self-refreshing is cleared by clearing the RMODE bit to 0. Self-refresh timing is shown in figure 7.48. Settings must be made so that self-refresh clearing and data retention are performed correctly, and CAS-before-RAS refreshing is immediately performed at the correct intervals. When self-refreshing is started from the state in which CAS-before-RAS refreshing is set, or when exiting standby mode by means of a manual reset or NMI, auto-refreshing is restarted if RFSH is set to 1 and RMODE is cleared to 0 when self-refresh mode is cleared. If the transition from clearing of self-refresh mode to starting auto-refresh takes time, this time should be taken into consideration when setting the initial value of RTCNT. When the RTCNT value is set to RTCOR-1, the refresh can be started immediately.

After self-refreshing has been set, the self-refresh state continues even if the chip standby state is entered using the chip's standby function. The self-refresh state is also maintained even after recovery from standby mode by means of NMI input.

In the case of a power-on reset, the bus state controller's registers are initialized, and therefore the self-refresh state is cleared.

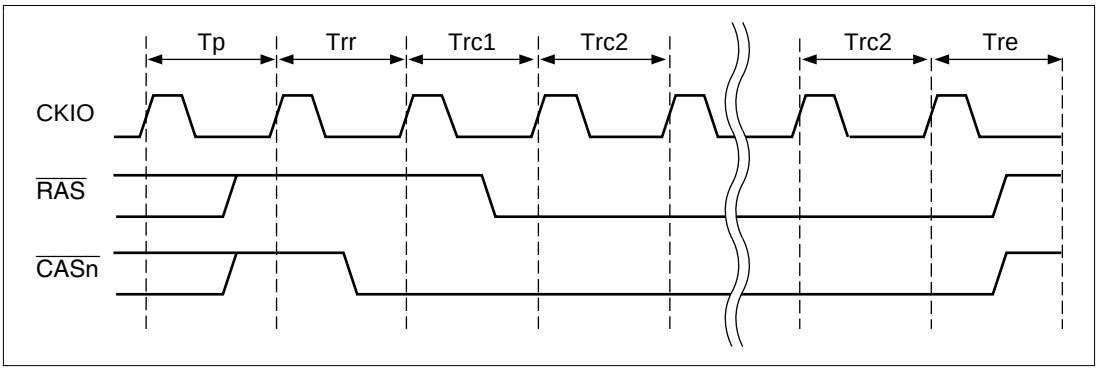



**Figure 7.48 DRAM Self-Refresh Cycle Timing**

### 7.6.9 Power-On Sequence

When DRAM is used after the power is turned on, there is a requirement for a waiting period during which accesses cannot be performed (100 µs or 200 µs minimum) followed by at least the prescribed number of dummy CAS-before-RAS refresh cycles (usually 8). The bus state controller (BSC) does not perform any special operations for the power-on reset, so the required power-on sequence must be implemented by the initialization program executed after a power-on reset.

## 7.7 Burst ROM Interface

Set the BSTROM bit in BCR1 to set the CS0 space for connection to burst ROM. The burst ROM interface is used to permit fast access to ROMs that have the nibble access function. Figure 7.50 shows the timing of nibble accesses to burst ROM. Set for two wait cycles. The access is basically the same as an ordinary access, but when the first cycle ends, only the address is changed. The CS0 signal is not negated, enabling the next access to be conducted without the T1 cycle required for ordinary space access. From the second time on, the T1 cycle is omitted, so access is 1 cycle faster than ordinary accesses. Currently, the nibble access can only be used on 4-address ROM. This function can only be utilized for word or longword reads to 8-bit ROM and longword reads to 16-bit ROM. Mask ROMs have slow access speeds and require 4 instruction fetches for 8-bit widths and 16 accesses for cache filling. Limited support of nibble access was thus added to alleviate this problem. When connecting to an 8-bit width ROM, a maximum of 4 consecutive accesses are performed; when connecting to a 16-bit width ROM, a maximum of 2 consecutive accesses are performed. Figure 7.49 shows the relationship between data width and access size. For cache filling and DMAC 16-byte transfers, longword accesses are repeated 4 times.

When one or more wait states are set for a burst ROM access, the $\overline{\text{WAIT}}$ pin is sampled. When the burst ROM is set and 0 specified for waits, there are 2 access cycles from the second time on. Figure 7.51 shows the timing.

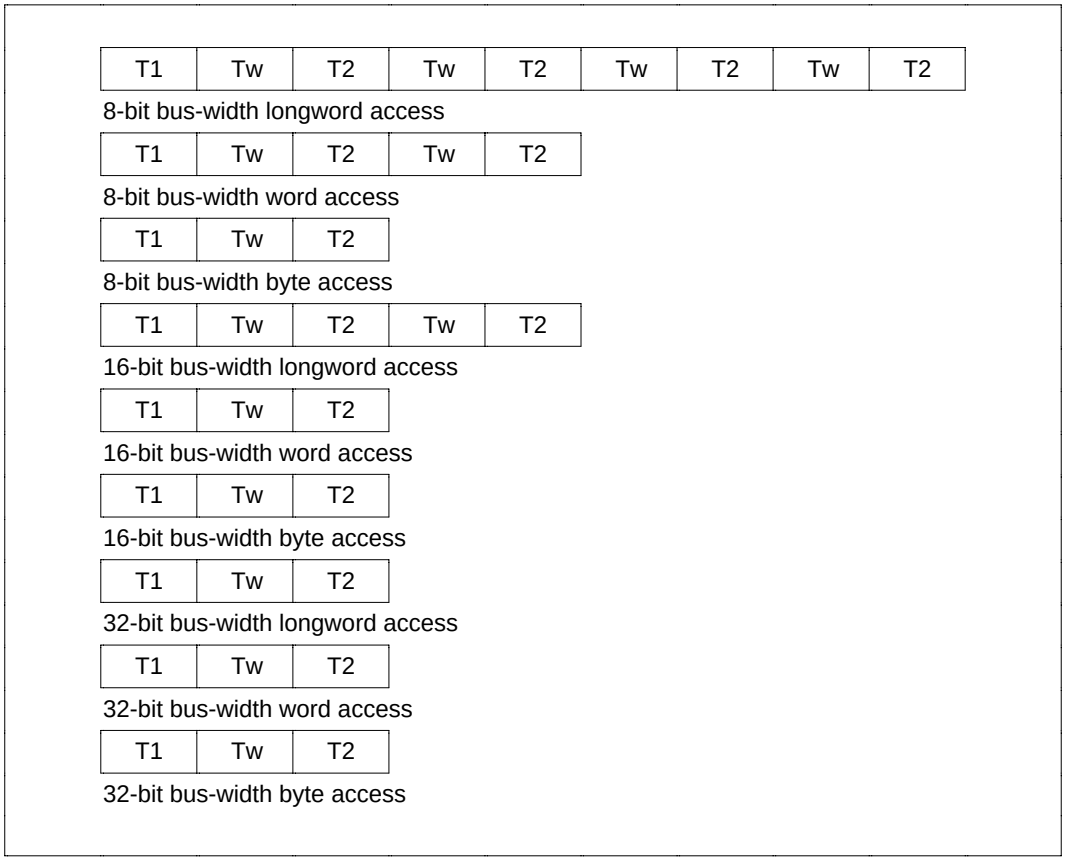
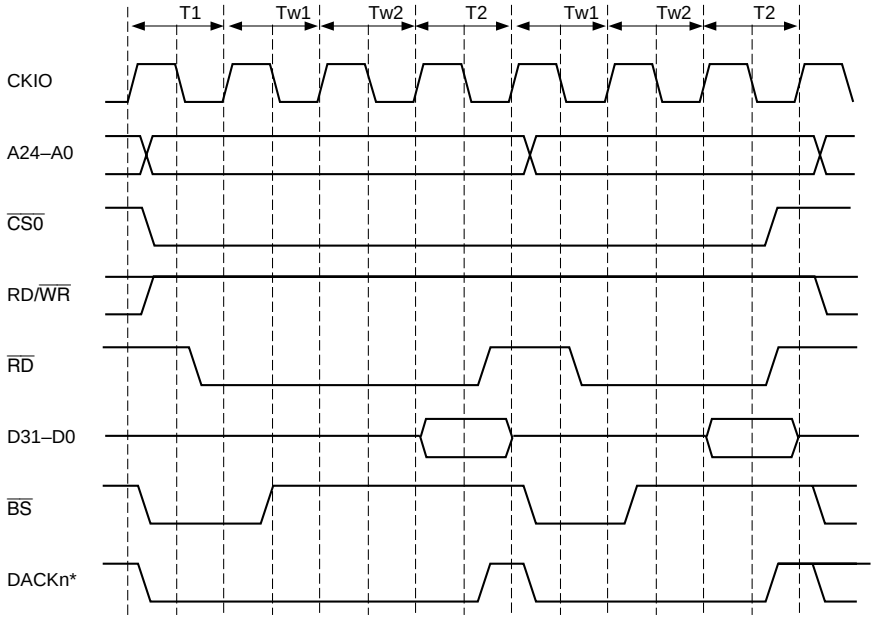
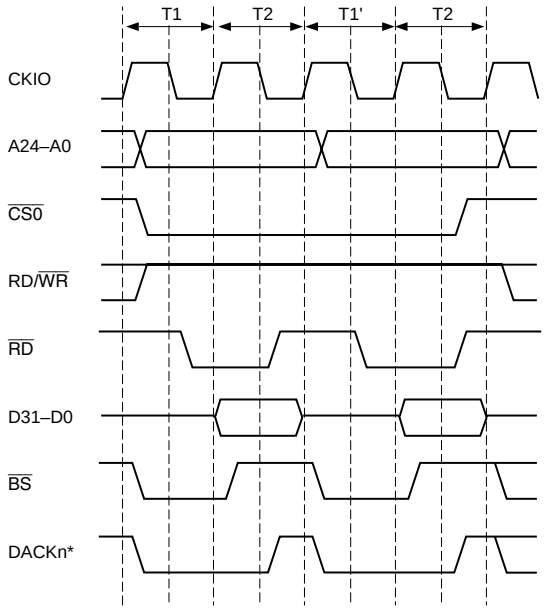



**Figure 7.49 Data Width and Burst ROM Access (1 Wait State)**




Note: * DACKn waveform when active-low is specified.

**Figure 7.50 Burst ROM Nibble Access (2 Wait States)**




Note: * DACKn waveform when active-low is specified.

**Figure 7.51 Burst ROM Nibble Access (No Wait States)**

## 7.8 Idles between Cycles

Because operating frequencies have become high, when a read from a slow device is completed, data buffers may not go off in time to prevent data conflicts with the next access. This lowers device reliability and causes errors. To prevent this, a function has been added to avoid data conflicts that memorizes the space and read/write state of the preceding access and inserts an idle cycle in the access cycle for those cases in which problems are found to occur when the next access starts up. The BSC checks whether a wait is to be inserted in two cases: if a read cycle is followed immediately by a read access to a different CS space, and if a read access is followed immediately by a write from the chip. When the chip is writing continuously, the data direction is always from the chip to other memory, and there are no particular problems. Neither is there any particular problem if the following read access is to the same CS space, since data is output from the same data buffer. The number of idle cycles to be inserted into the access cycle when reading from another CS space, or performing a write, after a read from the CS3 space, is specified by the IW31 and IW30 bits in WCR1. Likewise, IW21 and IW20 specify the number of idle cycles after CS2 reads, IW11 and IW10 specify the number after CS1 reads, and IW01 and IW00 specify the number after CS0 reads. From 0, 1, 2, or 4 cycles can be specified. When there is already a gap between accesses, the number of empty cycles is subtracted from the number of idle cycles before insertion. When a write cycle is performed immediately after a read access, 1 idle cycle is inserted even when 0 is specified for waits between access cycles.

When the chip shifts to a read cycle immediately after a write, the write data becomes high impedance when the clock rises, but the $\overline{RD}$ signal, which indicates read cycle data output enable, is not asserted until the clock falls. The result is that no idles are inserted into the cycle.

When bus arbitration is being performed, an empty cycle is inserted for arbitration, so no is inserted between cycles.

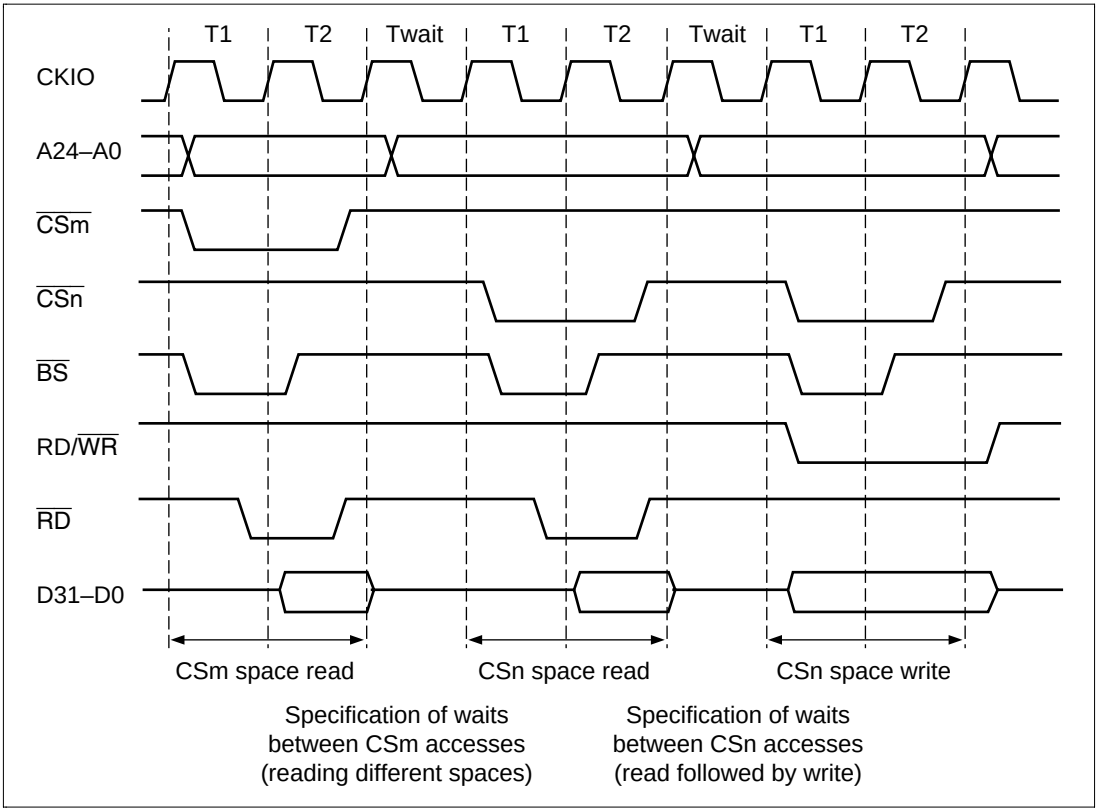



**Figure 7.52 Idles between Cycles**

## 7.9 Bus Arbitration

The chip has a bus arbitration function that, when a bus release request is received from an external device, releases the bus to that device after the bus cycle being executed is completed.

The chip keeps the bus under normal conditions and permits other devices to use the bus by releasing it when they request its use.

In the following explanation, external devices requesting the bus are called slaves.

The chip has two internal bus masters, the CPU and the DMAC. When synchronous DRAM or DRAM is connected and refresh control is performed, the refresh request becomes a third master. In addition to these, there are also bus requests from external devices. The priority for bus requests when they occur simultaneously is, highest to lowest, refresh requests, bus requests from external devices, DMAC and CPU.

When a bus request from the CPU is pending, the CPU has priority.

When the bus is being passed between slave and master, all bus control signals are negated before the bus is released to prevent erroneous operation of the connected devices. When the bus is transferred, also, the bus control signals begin bus driving from the negated state. The master and slave passing the bus between them drive the same signal values, so output buffer conflict is avoided. A pull-up resistance is required for the bus control signals to prevent malfunction caused by external noise when they are at high impedance.

Bus permission is granted at the end of the bus cycle. When the bus is requested, the bus is released immediately if there is no ongoing bus cycle. If there is a current bus cycle, the bus is not released until the bus cycle ends. Even when a bus cycle does not appear to be in progress when viewed from off-chip, it is not possible to determine immediately whether the bus has been released by looking at $\overline{\text{CS}}$n or other control signals, since a bus cycle (such as wait insertion between access cycles) may have been started internally. The bus cannot be released during burst transfers for cache filling or 16-byte DMAC block transfers. Likewise, the bus cannot be released between the read and write cycles of a TAS instruction. Arbitration is also not performed between multiple bus cycles produced by a data width smaller than the access size, such as a longword access to an 8-bit data width memory. Bus arbitration is not performed in the self-refresh state in synchronous DRAM and the DRAM interface. Bus arbitration is performed between external vector fetch, PC save, and SR save cycles during interrupt handling, which are all independent accesses.

Because the CPU is connected to cache memory by a dedicated internal bus, cache memory can be read even when the bus is being used by another bus master on the chip or externally. When writing from the CPU, an external write cycle is produced. Since the internal bus that connects the CPU, DMAC, and on-chip peripheral modules can operate in parallel to the external bus, both read and write accesses from the CPU to on-chip peripheral modules and from the DMAC to on-chip peripheral modules are possible even if the external bus is not held.

### 7.9.1 Master Mode

The chip keeps the bus unless it receives a bus request. When a bus release request ($\overline{\text{BRLS}}$) assertion (low level) is received from an external device, buses are released and a bus grant ($\overline{\text{BGR}}$) is asserted (low level) as soon as the bus cycle being executed is completed. When it receives a negated (high level) $\overline{\text{BRLS}}$ signal, indicating that the slave has released the bus, it negates the $\overline{\text{BGR}}$ (to high level) and begins using the bus. When the bus is released, all output and I/O signals related to the bus interface are changed to high impedance, except for the CKE signal for the synchronous DRAM interface, the $\overline{\text{BGR}}$ signal for bus arbitration, and DMA transfer control signals DACK0 and DACK1.

When DRAM has finished precharging, the bus is released. With synchronous DRAM, too, the bus is released after issuing an all-banks precharge command for the active bank, and after issuing an NOP command in other cases.

The specific bus release sequence is as follows. First, the address bus and data bus become high impedance synchronously with a rise of the clock. Half a cycle later, the bus use enable signal is asserted synchronously with a fall of the clock. Thereafter the bus control signals ($\overline{BS}$, $\overline{CSn}$, $\overline{RAS}$, $\overline{CAS}$, $\overline{WEn}$, $\overline{RD}$, RD/$\overline{WR}$) become high impedance at a rise of the clock. These bus control signals are driven high at least 2 cycles before they become high impedance. Sampling for bus request signals occurs at the clock fall.

The sequence when the bus is taken back from the slave is as follows. When the negation of $\overline{BRLS}$ is detected at a clock fall, high-level driving of the bus control signals starts half a cycle later. The bus use enable signal is then negated at the next clock fall. The address bus and data bus are driven starting at the next clock rise. The bus control signals are asserted and the bus cycle actually starts from the same clock rise at which the address and data signals are driven, at the earliest. Figure 7.52 shows the timing of bus arbitration.

When DRAM or synchronous DRAM is accessed on the slave side, the design must provide for the bus to be released after completion of precharging for DRAM or synchronous DRAM on the slave side.

To reduce the overhead due to arbitration with a user-designed slave, a number of consecutive bus accesses may be attempted. In this case, to insure dependable refreshing, the design must provide for the slave to release the bus before it has held it for a period exceeding the refresh cycle. The SH7612 is provided with the REFOUT pin to send a signal requesting the bus while refresh execution is being kept waiting. REFOUT is asserted while refresh execution is being kept waiting until the bus is acquired. When the external slave device receives this signal and releases the bus, the bus is returned to the chip and refreshing can be executed.

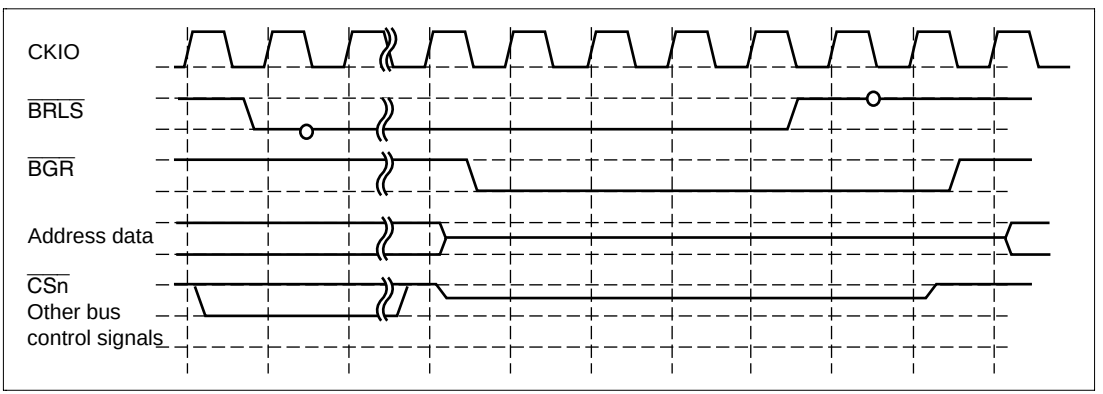



**Figure 7.53 Bus Arbitration**

## 7.10 Additional Items

### 7.10.1 Resets

The bus state controller is completely initialized only in a power-on reset. All signals are immediately negated, regardless of whether or not the chip is in the middle of a bus cycle. Signal negation is simultaneous with turning the output buffer off. All control registers are initialized. In standby mode, sleep mode, and a manual reset, no bus state controller control registers are initialized. When a manual reset is performed, the currently executing bus cycle only is completed, and then the chip waits for an access. When a cache filling or 16-byte DMAC transfer is executing, the CPU or DMAC that is the bus master ends the access in a longword unit, since the access request is canceled by the manual reset. This means that when a manual reset is executed during a cache filling, the cache contents can no longer be guaranteed. During a manual reset, the RTCNT does not count up, so no refresh request is generated, and a refresh cycle is not initiated. To preserve the data of the DRAM and synchronous DRAM, the pulse width of the manual reset must be shorter than the refresh interval. Master mode chips accept arbitration requests even when a manual reset signal is asserted. When a reset is executed only for the chip in master mode while the bus is released, the $\overline{\text{BGR}}$ signal is negated to indicate this. If the $\overline{\text{BRLS}}$ signal is continuously asserted, the bus release state is maintained.

### 7.10.2 Access as Viewed from CPU or DMAC

The chip is internally divided into three buses: cache, internal, and peripheral. The CPU and cache memory are connected to the cache bus, the DMAC and bus state controller are connected to the internal bus, and the low-speed peripheral devices and mode registers are connected to the peripheral bus. On-chip memory other than cache memory and the user break controller are connected to both the cache bus and the internal bus. The internal bus can be accessed from the cache bus, but not the other way around. The peripheral bus can be accessed from the internal bus, but not the other way around. This results in the following.

The DMAC can access on-chip memory other than cache memory, but cannot access cache memory. When the DMAC causes a write to external memory, the external memory contents and the cache contents may be different. When external memory contents are rewritten by a DMA transfer, the cache memory must be purged by software if there is a possibility that the data for that address is present in the cache.

When the CPU starts a read access, if the access is to a cache area, a cache search is first performed. This takes one cycle. If there is data in the cache, it fetches it and completes the access. If there is no data in the cache, a cache filling is performed via the internal bus, so four consecutive longword reads occur. For misses that occur when byte or word operands are accessed or branches occur to odd word boundaries (4n + 2 addresses), the filling is always performed by longword accesses on the chip-external interface. In the cache-through area, the access is to the actual access address. When the access is an instruction fetch, the access size is always longword.

For cache-through areas and on-chip peripheral module read cycles, after an extra cycle is added to determine the cycle, the read cycle is started through the internal bus. Read data is sent to the CPU through the cache bus.

When write cycles access the cache area, the cache is searched. When the data of the relevant address is found, it is written here. The actual write occurs in parallel to this via the internal bus. When the right to use the internal bus is held, the CPU is notified that the write is completed without waiting for the end of the actual off-chip write. When the right to use the internal bus is not held, as when it is being used by the DMAC or the like, there is a wait until the bus is acquired before the CPU is notified of completion.

Accesses to cache-through areas and on-chip peripheral modules work the same as in the cache area, except for the cache search and write.

Because the bus state controller has one level of write buffer, the internal bus can be used for another access even when the chip-external bus cycle has not ended. After a write has been performed to low-speed memory off the chip, performing a read or write with an on-chip peripheral module enables an access to the on-chip peripheral module without having to wait for the completion of the write to low-speed memory.

During reads, the CPU always has to wait for the end of the operation. To immediately continue processing after checking that the write to the device of actual data has ended, perform a dummy read access to the same address consecutively to check that the write has ended.

The bus state controller’s write buffer functions in the same way during accesses from the DMAC. A dual-address DMA transfer thus starts in the next read cycle without waiting for the end of the write cycle. When both the source address and destination address of the DMA are external spaces to the chip, however, it must wait until the completion of the previous write cycle before starting the next read cycle.

### 7.10.3 On-Chip Peripheral Module Access

When accessing peripheral module registers, the number of access cycles depends on the register.

| Peripheral Module Register Address | Cycles |
|---|---|
| H'FFFFFC00 to H'FFFFFCFF | 3—4 Pø cycles* |
| H'FFFFFE00 to H'FFFFFEFF | 4—5 Pø cycles* |
| H'FFFFFF00 to H'FFFFFFFF | 2—3 Iø cycles* |

Note: * Pø: Peripheral module clock
Iø: CPU/DSP core clock

### 7.10.4 Notes on Memory Access when Using the DMAC

**Note on synchronous DRAM single write mode when clock ratio Iø:Eø = 1:1**

Conditions: Clock ratio Iø:Eø = 1:1 (Iø: CPU clock, Eø: external memory clock (CKIO))
SDRAM single write mode
DMAC dual address transfer
Read from SDRAM immediately after write to SDRAM (see table 7.8)

**Table 7.8 Access Sequence**

| Write to SDRAM | Read from SDRAM |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur when a read is performed by the CPU immediately after a write, because the accesses are not consecutive internally.

Problem: When an SDRAM read access is performed, the SH7612 drives the data bus.

Remedy: When accessing SDRAM using the DMAC in dual address mode with clock ratio Iø:Eø = 1:1, the SDRAM should be used in burst write mode.

**Note on writing to normal space immediately after a synchronous DRAM write when clock ratio Iø:Eø = 1:1**

Conditions: Clock ratio Iø:Eø = 1:1 (Iø: CPU clock, Eø: external memory clock (CKIO))
Write to normal space immediately after write to SDRAM (see table 7.9)

**Table 7.9 Access Sequence**

| Write to SDRAM | Write to Normal Space |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur when a write is performed by the CPU immediately after a write, because the accesses are not consecutive internally.

Problem: When a write to normal space is performed, the data bus is not driven in the first cycle (figures 7.54 to 7.56).

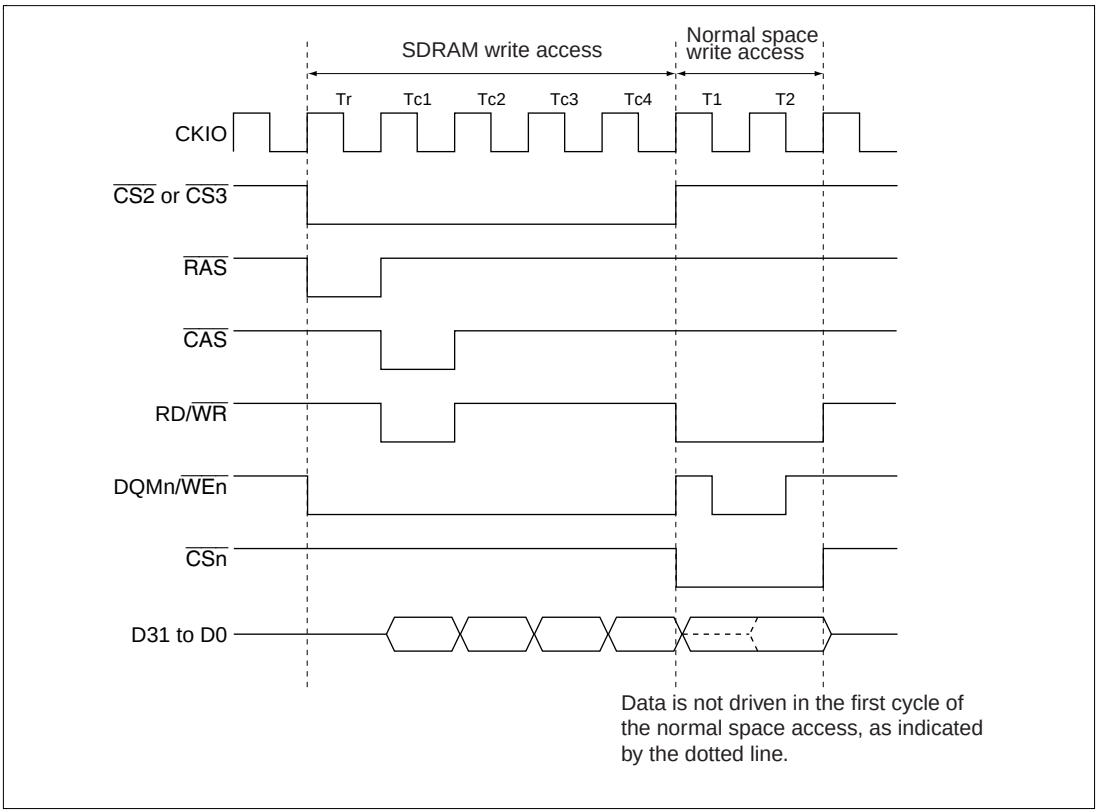



**Figure 7.54 Normal Space Access (No Wait)**

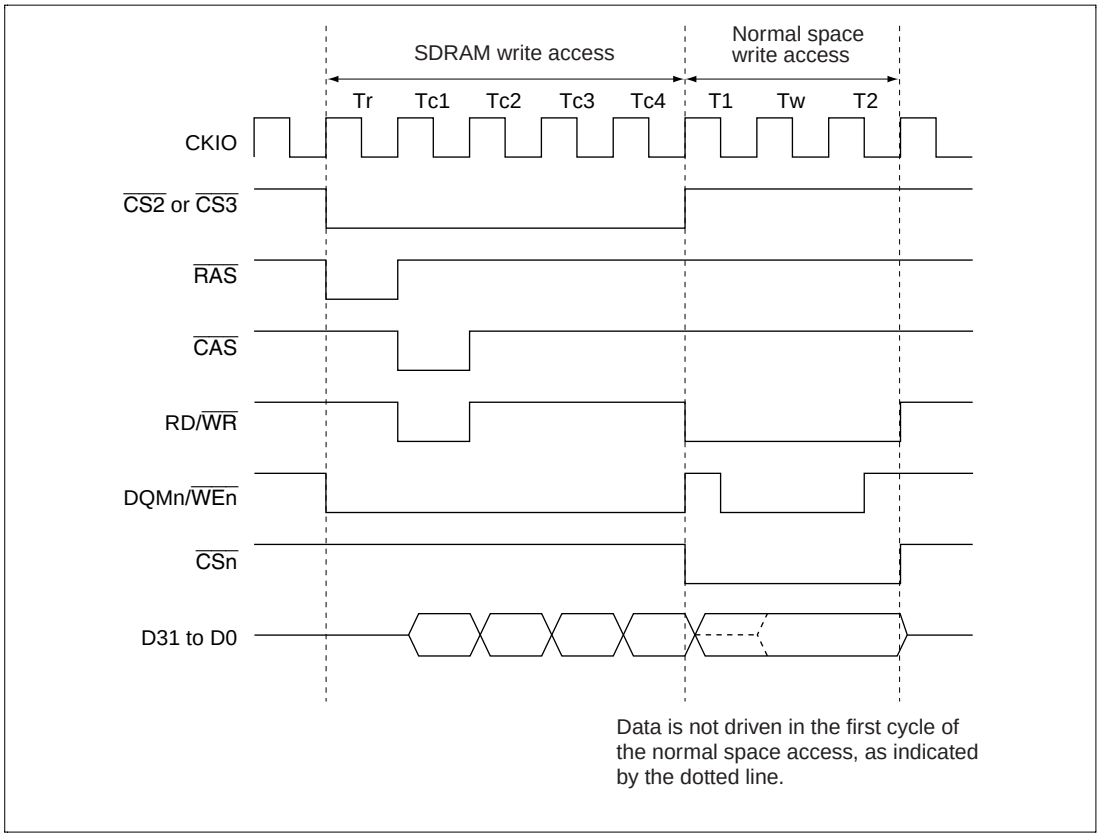



**Figure 7.55 Normal Space Access (One Wait)**

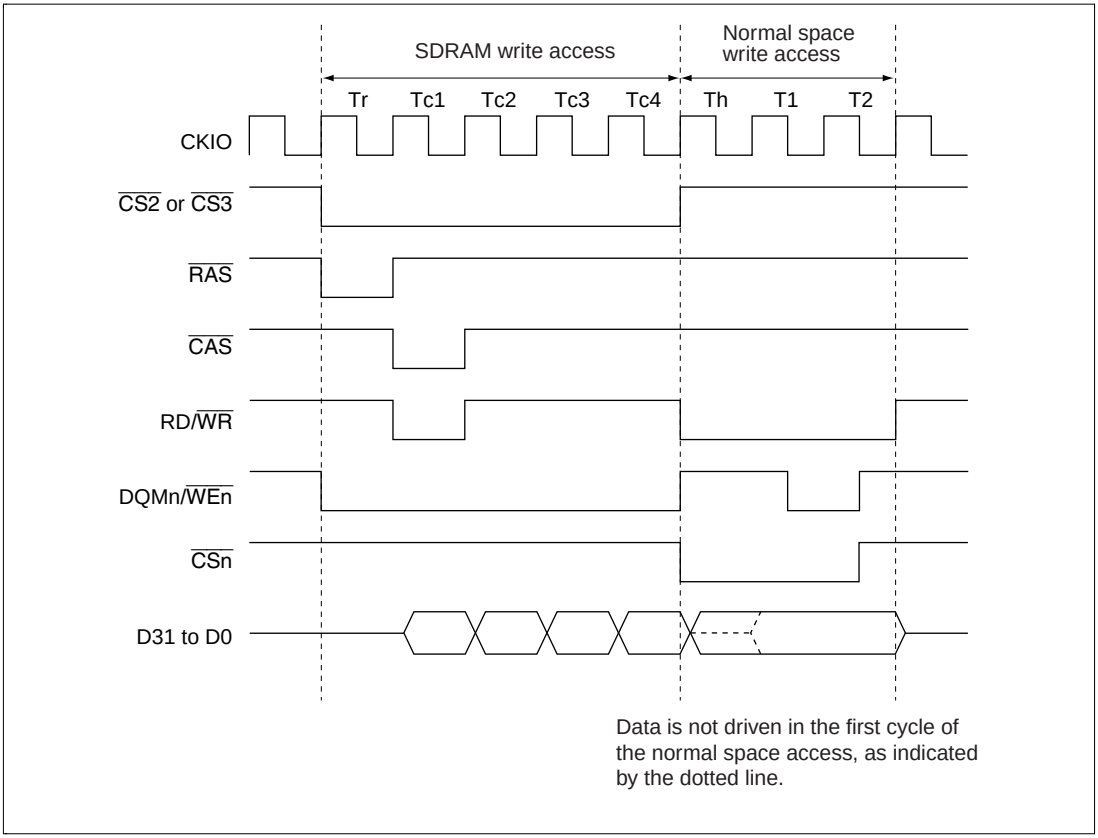



**Figure 7.56 Normal Space Access (One Setup Wait Cycle)**

Remedy: When the above conditions apply, in order to secure the normal space write data setup time, either increase the normal space wait by one cycle, or increase the number of cycles from address and $\overline{CSn}$ output to $\overline{RD}$ and $\overline{WEn}$ assertion by one.

**Note on synchronous DRAM bank-active mode**

Conditions: SDRAM bank-active mode
SDRAM access by DMAC
Consecutive accesses to SDRAM (see table 7.10)

**Table 7.10 Access Sequence**

| Access to SDRAM | Access to SDRAM |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur with consecutive accesses by the CPU, because the accesses are not consecutive internally.

Problem: The second SDRAM access is not performed correctly.

Remedy: Do not use bank-active mode when accessing SDRAM using the DMAC.

**Note on EDO access in DMAC single address mode**

Conditions:

- DMAC operating mode
  - — Single address transfer mode
  - — Transfer from memory to device with DACK (AM: CHRC0, CHRC1 bit 8 = 1)
- BSC setting
  - — EDO RAS down mode
  - — W31—W30 = 00

Problem: Although RAS down mode is set, $\overline{RAS}$ is negated for one cycle. The next access to EDO may not be performed correctly.

Remedy: When performing transfer from memory to a device with DACK (AM: CHRC0, CHRC1 bit 8 = 1) for EDO in DMAC single address transfer mode, either do not use RAS down mode, or else set bits W31—W30 to a value other than 00.

**Note on normal space access immediately after a synchronous DRAM write:** The DQMn/$\overline{WEn}$ signal negation for an SDRAM write and the $\overline{CSn}$ assertion for an immediately following normal space access occur at the same rising edge of CKIO (figures 7.57 and 7.58). The access sequences when an SDRAM write and normal space access are consecutive are shown in table 7.11.

**Table 7.11 Access Sequence**

| Write to SDRAM | Read from SDRAM |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: When a write is performed by the CPU immediately after a write, the accesses are not consecutive internally.

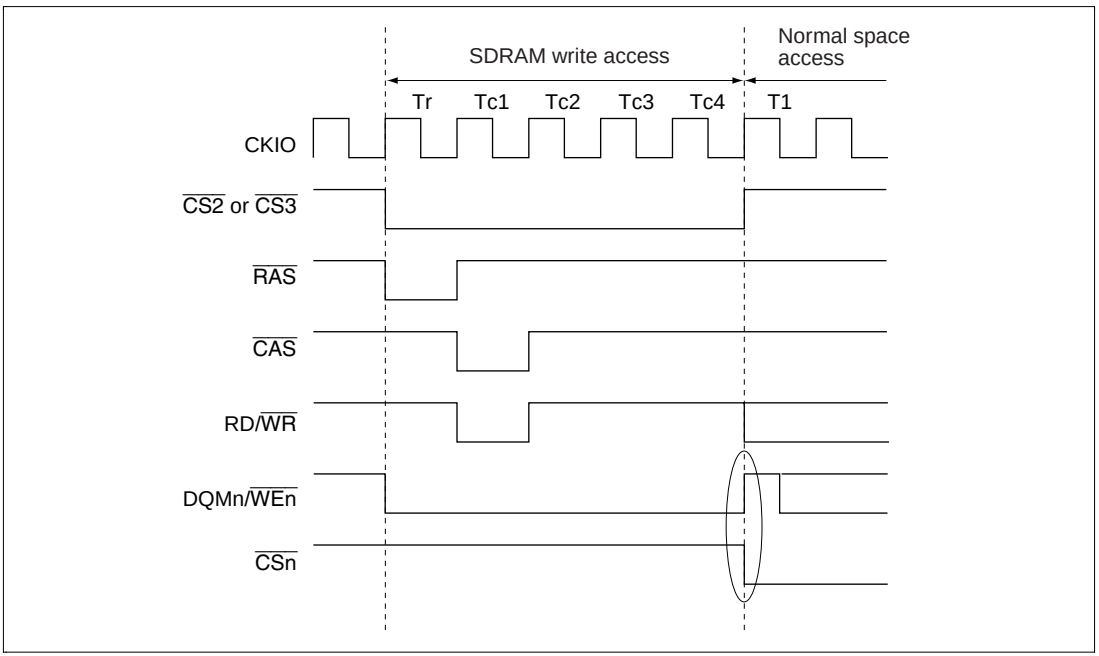



**Figure 7.57 Normal Space Access after SDRAM Write (Burst Write Mode)**

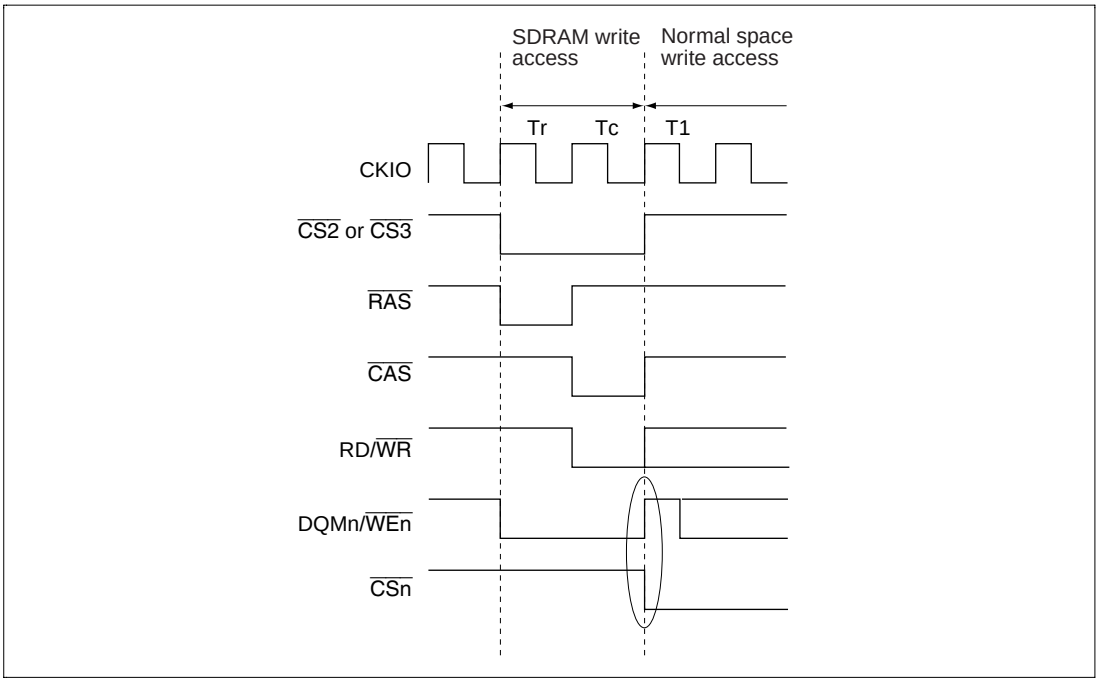



**Figure 7.58 Normal Space Access after SDRAM Write (Single Write Mode)**

**Note on DACK output when synchronous DRAM is accessed in DMAC single address mode:** In the case of consecutive accesses to synchronous DRAM by the DMAC, DACK is not negated in the Trwl and Tap cycles.

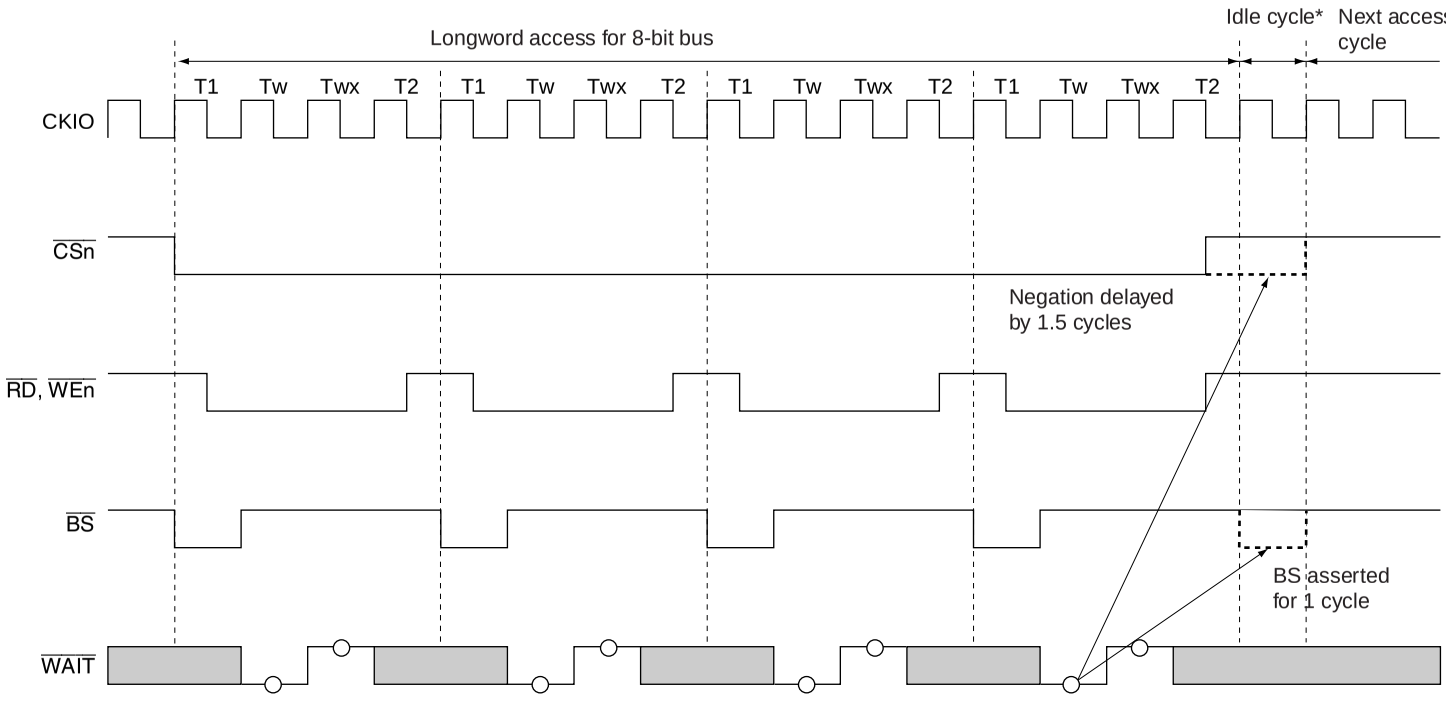
### 7.10.5 Additional Note

**Bug occurring with clock ratio Iø:Eø = 1:1, 8-bit bus width, and external wait input**

Conditions: Clock ratio Iø:Eø = 1:1 (Iø: CPU clock, Eø: external memory clock (CKIO))
8-bit external bus width
0.5 address hold cycle (AnSHW1—AnSHW0 = 00, A4HW1—A4HW0 = 00)
External wait input

Problem: When an external wait is inserted in the fourth burst transfer, $\overline{\text{CSn}}$ negation is delayed by 1.5 cycles. $\overline{\text{BS}}$ is asserted for one cycle, one-half cycle after $\overline{\text{RD}}$ or $\overline{\text{WEn}}$ negation (figure 7.59).

Remedy: When using an 8-bit bus width with clock ratio Iø:Eø = 1:1, one or more hold waits should be set. Specifically, set AnSHW1—AnSHW0 > 00 and A4HW1—A4HW0 > 00 for the relevant space.




Conditions: Clock ratio Iø:Eø = 1:1
8-bit bus width
0.5 address hold cycle
External wait input

Note: * When clock ratio Iø:Eø = 1:1, one idle cycle is inserted between cycles.

**Figure 7.59 External Wait Input**

## 8.1 Introduction

This chip incorporates 4 kbytes of four-way, mixed instruction/data type cache memory. This memory can also be used as 2-kbyte RAM and 2 kbyte mixed instruction/data type cache memory by making a setting in the cache control register (CCR) (two-way cache mode). CCR can specify that either instructions or data do not use cache.

Each line of cache memory consists of 16 bytes. Cache memory is always updated in line units. Four 32-bit accesses are required to update a line. Since the number of entries is 64, the six bits (A9 to A4) in each address determine the entry. A four-way set associative configuration is used, so up to four different instructions/data can be stored in the cache even when entry addresses match. To efficiently use four ways having the same entry address, replacement is provided based on a pseudo-LRU (least-recently used) replacement algorithm.

The cache configuration is shown in figure 8.1, and addresses in figure 8.2.

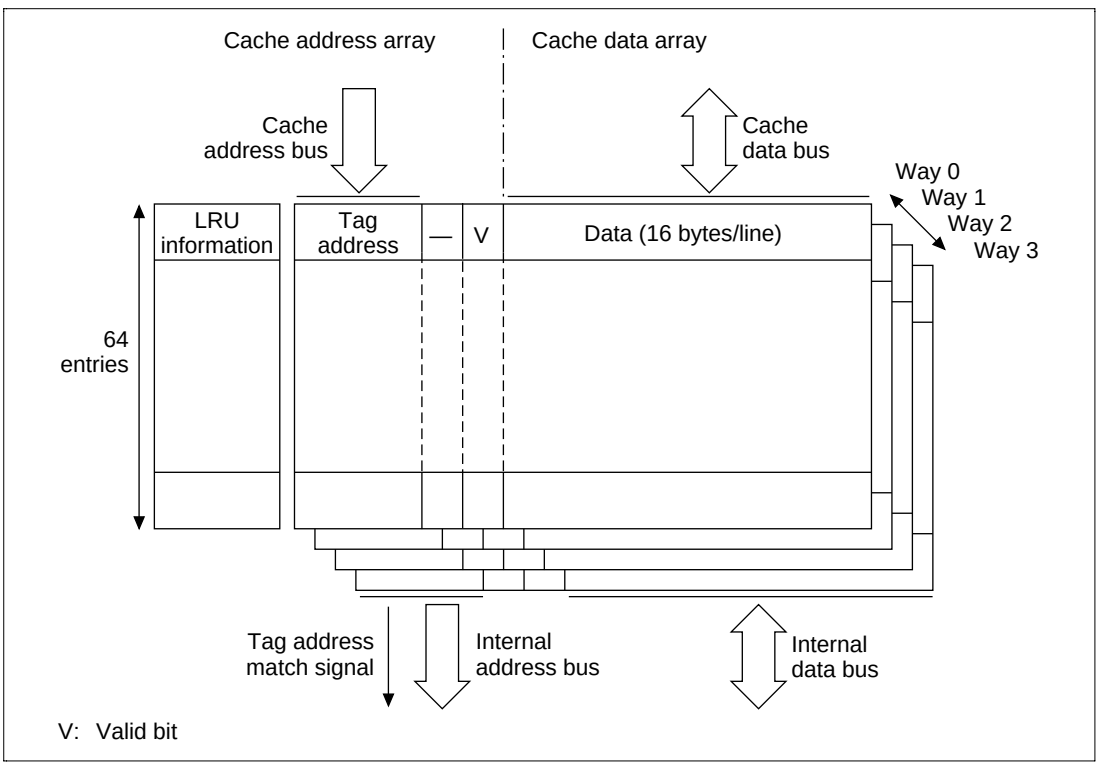



**Figure 8.1 Cache Configuration**

**Figure 8.2 Address**

### 8.1.1 Register Configuration

Table 8.1 shows the cache register configuration.

**Table 8.1 Register Configuration**

| Name | Abbrev. | R/W | Initial Value | Address |
|---|---|---|---|---|
| Cache control register | CCR | R/W | H'00 | H'FFFFFE92 |

## 8.2 Register Description

### 8.2.1 Cache Control Register (CCR)

The cache control register (CCR) is used for cache control. CCR must be set and the cache must be initialized before use. CCR is initialized to H'00 by a power-on reset or manual reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | W1 | W0 | — | CP | TW | OD | ID | CE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

Bits 7 and 6—Way Specification Bit (W1, W0): W1 and W0 specify the way when an address array is directly accessed by address specification.

| Bit 7: W1 | Bit 6: W0 | Description | |
|---|---|---|---|
| 0 | 0 | Way 0 | (Initial value) |
| | 1 | Way 1 | |
| 1 | 0 | Way 2 | |
| | 1 | Way 3 | |

Bit 5—Reserved: This bit always reads 0. The write value should always be 0.

| Bit 5: WB | Description | |
|---|---|---|
| 0 | Write-through | (Initial value) |
| 1 | Write-back | |

Bit 4—Cache Purge Bit (CP): When 1 is written to the CP bit, all cache entries and the valid bits, and LRU information of all ways are initialized to 0. After initialization is completed, the CP bit reverts to 0. The CP bit always reads 0.

| Bit 4: CP | Description | |
|---|---|---|
| 0 | Normal operation | (Initial value) |
| 1 | Cache purge | |

Note: Always read 0.

Bit 3—Two-Way Mode (TW): TW is the two-way mode bit. The cache operates as a four-way set associative cache when TW is 0 and as a two-way set associative cache and 2-kbyte RAM when TW is 1. In the two-way mode, ways 2 and 3 are cache and ways 0 and 1 are RAM. Ways 0 and 1 are read or written by direct access of the data array according to address space specification.

| Bit 3: TW | Description | |
|---|---|---|
| 0 | Four-way mode | (Initial value) |
| 1 | Two-way mode | |

Bit 2—Data Replacement Disable Bit (OD): OD is the bit for disabling data replacement. When this bit is 1, data fetched from external memory is not written to the cache even if there is a cache miss. Cache data is, however, read or updated during cache hits. OD is valid only when CE is 1.

| Bit 2: OD | Description | |
|---|---|---|
| 0 | Normal operation | (Initial value) |
| 1 | Data not replaced even when cache miss occurs in data access | |

Bit 1—Instruction Replacement Disable Bit (ID): ID is the bit for disabling instruction replacement. When this bit is 1, an instruction fetched from external memory is not written to the cache even if there is a cache miss. Cache data is, however, read or updated during cache hits. ID is valid only when CE is 1.

| Bit 1: ID | Description | |
|---|---|---|
| 0 | Normal operation | (Initial value) |
| 1 | Data not replaced even when cache miss occurs in instruction fetch | |

Bit 0—Cache Enable Bit (CE): CE is the cache enable bit. Cache can be used when CE is set to 1.

| Bit 0: CE | Description | |
|---|---|---|
| 0 | Cache disabled | (Initial value) |
| 1 | Cache enabled | |

## 8.3 Address Space and the Cache

The address space is divided into six partitions. The cache access operation is specified by addresses. Table 8.2 lists the partitions and their cache operations. For more information on address spaces, see section 7, Bus State Controller. Note that the spaces of the cache area and cache-through area are the same.

**Table 8.2 Address Space and Cache Operation**

| Addresses A31–A28 | Partition | Cache Operation |
|---|---|---|
| 0000 | Cache area | Cache is used when the CE bit in CCR is 1 |
| 0001* | On-chip memory area | Cache is not used |
| 001* | Cache-through area | Cache is not used |
| 010* | Associative purge area | Cache line of the specified address is purged (disabled) |
| 011* | Address array read/write area | Cache address array is accessed directly |
| 110* | Data array read/write area | Cache data array is accessed directly |
| 111* | Synchronous DRAM mode setting area and on-chip peripheral module area | Cache is not used |

Note: * A28 has no effect.

## 8.4 Cache Operation

### 8.4.1 Cache Reads

This section describes cache operation when the cache is enabled and data is read from the CPU. One of the 64 entries is selected by the entry address part of the address output from the CPU on the cache address bus. The tag addresses of ways 0 through 3 are compared to the tag address parts of the addresses output from the CPU. When there is a way for which the tag address matches, this is called a cache hit (when any one of the way tag addresses and the tag address of the address output from the CPU match). In proper use, the tag addresses of each way differ from each other, and the tag address of only one way will match. When none of the way tag addresses match, it is called a cache miss. Tag addresses of entries with valid bits of 0 will not match in any case.

When a cache hit occurs, data is read from the data array of the way that was matched according to the entry address, the byte address within the line, and the access data size, and is sent to the CPU. The address output on the cache address bus is calculated in the CPU's instruction execution phase and the results of the read are written during the CPU's write-back stage. The cache address bus and cache data bus both operate as pipelines in concert with the CPU's pipeline structure. From address comparison to data read requires 1 cycle; since the address and data operate as a pipeline, consecutive reads can be performed at each cycle with no waits (figure 8.3).

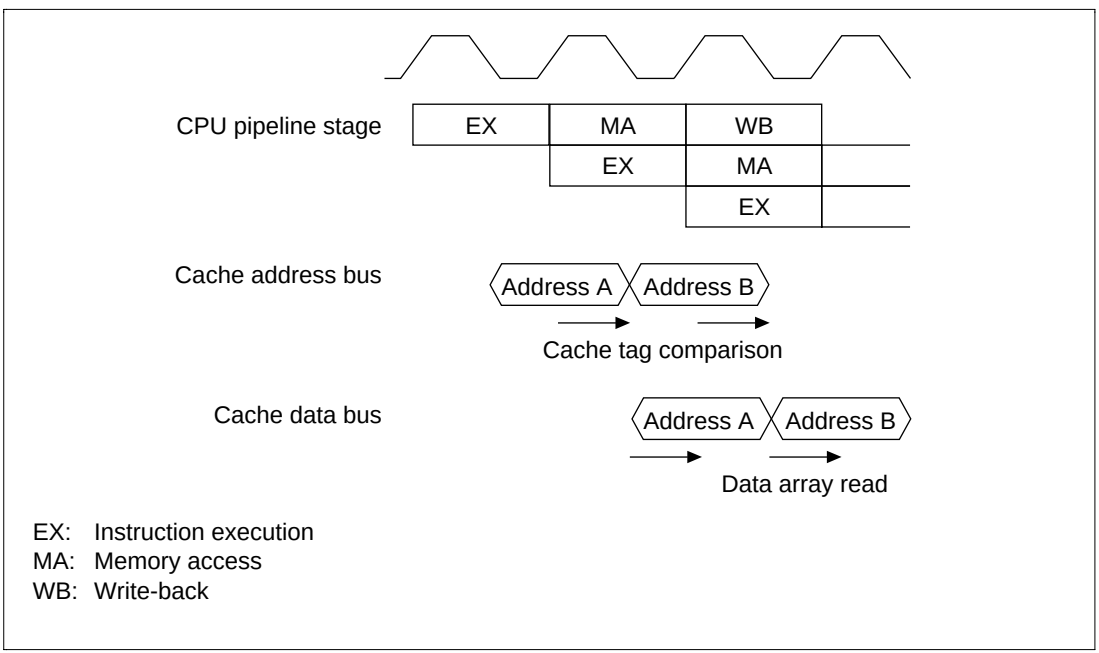



**Figure 8.3 Read Access in Case of a Cache Hit**

When a cache miss occurs, the way for replacement is determined using the LRU information, and the read address from the CPU is written in the address array for that way. Simultaneously, the valid bit is set to 1. Since the 16 bytes of data for replacing the data array are simultaneously read, the address on the cache address bus is output to the internal address bus and 4 longwords are read consecutively. The access order is such that, for the address output to the internal address, the byte address within the line is sequentially incremented by 4, so that the longword that contains the address to be read from the cache comes last. The read data on the internal data bus is written sequentially to the cache data array. One cycle after the last data is written to the cache data array, it is also output to the cache data bus and the read data is sent to the CPU.

The internal address bus and internal data bus also function as pipelines, just like the cache bus (figure 8.4).

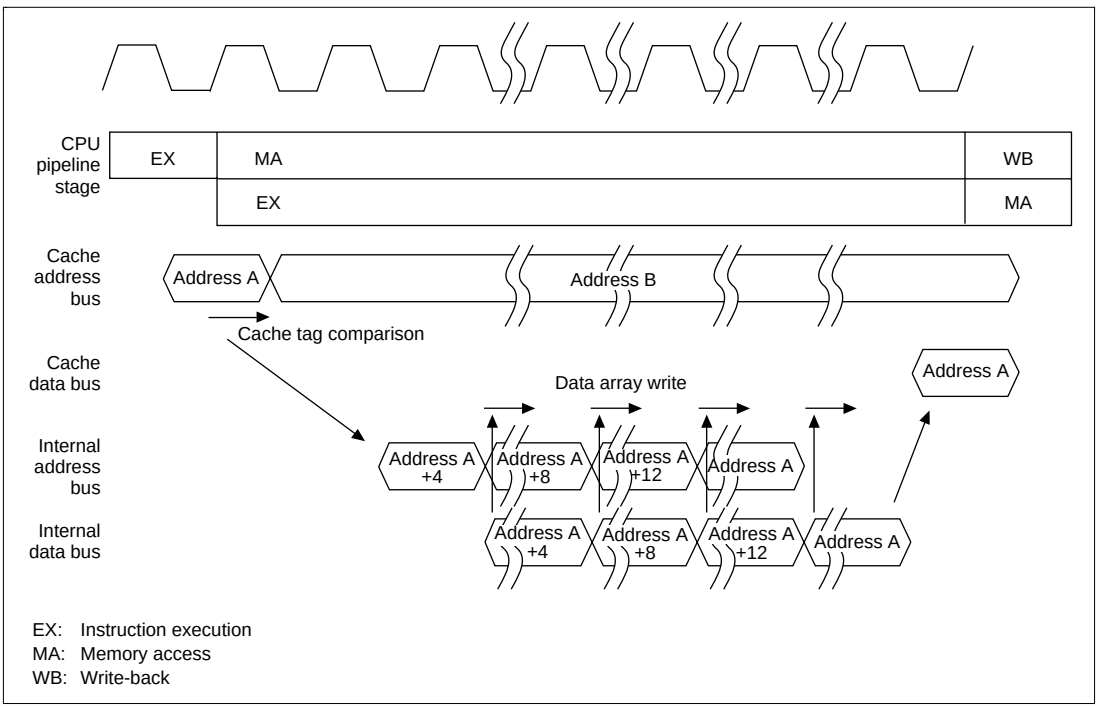



**Figure 8.4 Read Access in Case of a Cache Miss**

### 8.4.2 Write Access

**Write-Through Mode:** Writing to external memory is performed regardless of whether or not there is a cache hit. The write address output to the cache address bus is used for comparison to the tag address of the cache's address array. If they match, the write data output to the cache data bus in the following cycle is written to the cache data array. If they do not match, nothing is written to the cache data array. The write address is output to the internal address bus 1 cycle later than the cache address bus. The write data is similarly output to the internal data bus 1 cycle later than the cache data bus. The CPU waits until the writes on the internal buses are completed (figure 8.5).

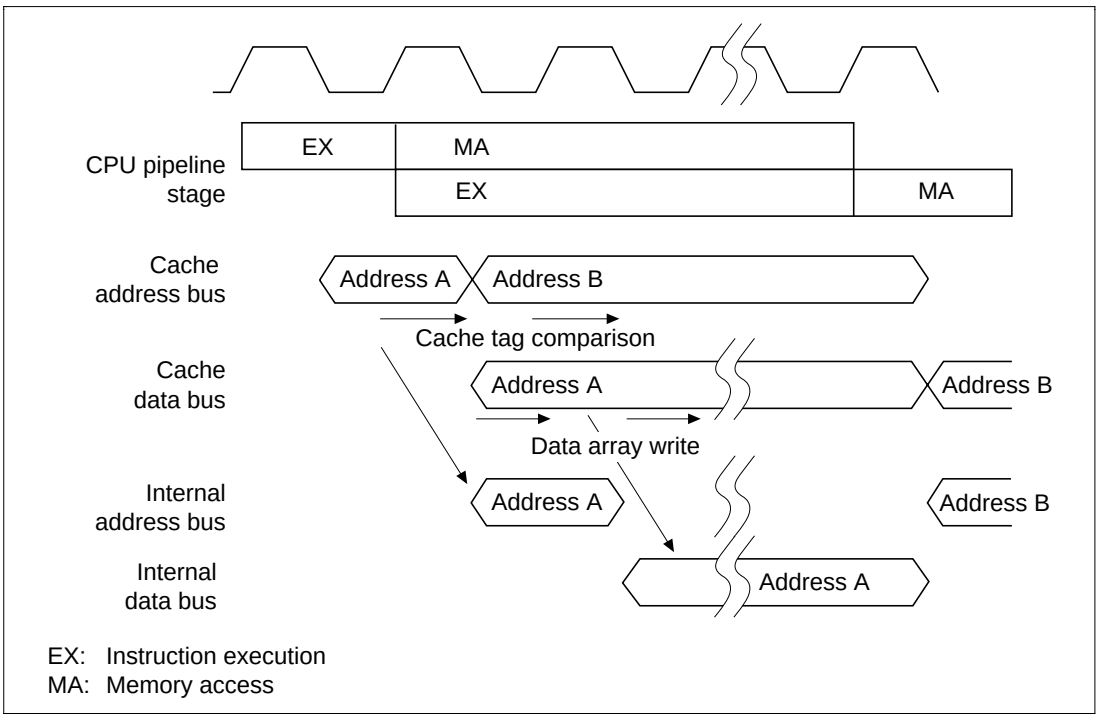



**Figure 8.5 Write Access (Write-Through)**

### 8.4.3 Cache-Through Access

When reading or writing a cache-through area, the cache is not accessed. Instead, the cache address value is output to the internal address bus. For read operations, the read data output to the internal data bus is fetched and output to the cache data bus, as shown in figure 8.6. The read of the cache-through area is only performed on the address in question. For write operations, the write data on the cache data bus is output to the internal data bus. Writes on the cache through area are compared to the address tag; except for the fact that nothing is written to the data array, the operation is the same as the write shown in figure 8.5.

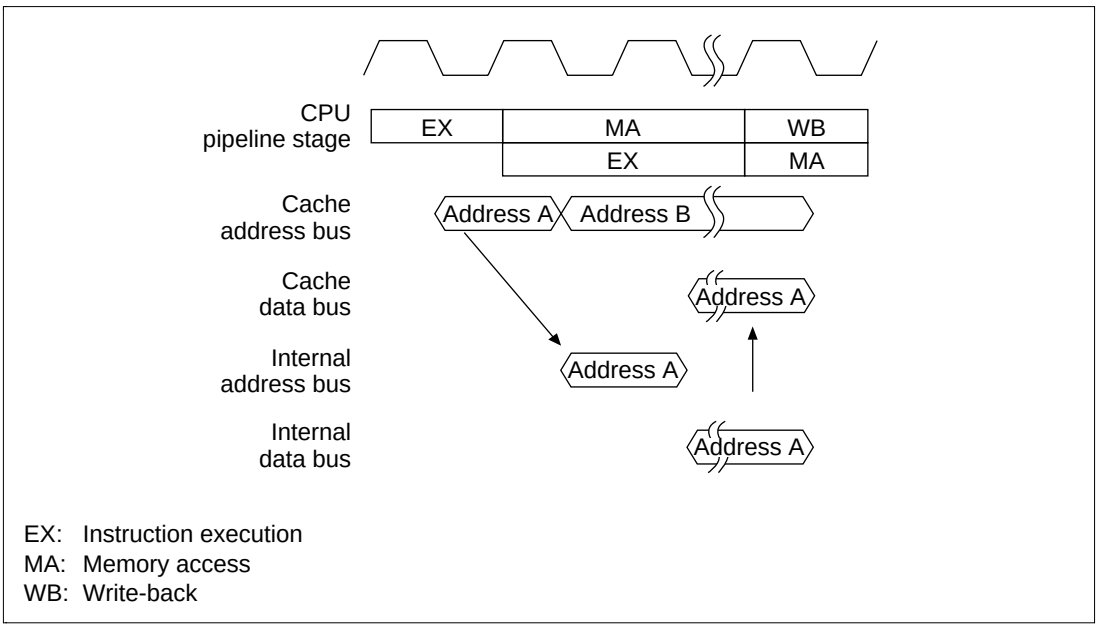



**Figure 8.6 Reading Cache-Through Areas**

### 8.4.4 The TAS Instruction

The TAS instruction reads data from memory, compares it to 0, reflects the result in the T bit of the status register, and sets the most significant bit to 1. It is an instruction that writes to the same address. Accesses to the cache area are handled in the same way as ordinary data accesses.

### 8.4.5 Pseudo-LRU and Cache Replacement

When a cache miss occurs during a read, the data of the missed address is read from 1 line (16 bytes) of memory and replaced. It is therefore necessary to decide which of the four ways is to be replaced. It can generally be expected that a way that has been infrequently used recently is also unlikely to be used next. This algorithm for replacing ways is called the least recently used replacement algorithm, or LRU. The hardware to implement it, however, is complex. For that

reason, this cache uses a pseudo-LRU replacement algorithm that keeps track of the order of way access and replaces the oldest way.

Six bits of data are used as the LRU information. The bits indicate the access order for 2 ways, as shown in figure 8.7. When the value is 1, access occurred in the direction of the appropriate arrow in the figure. The direction of the arrow can be determined by reading the bit. Access to the way to which all the arrows are pointing is the oldest, and that way becomes subject to replacement. The access order is recorded in the LRU information bits, so the LRU information is rewritten when a cache hit occurs during a read, when a cache hit occurs during a write, and when replacement occurs after a cache miss. Table 8.3 shows the rewrite values; table 8.4 shows how the way to be replaced is selected.

After a cache purge by means of the CP bit in CCR, all the LRU information is zeroized, so the initial order of use is way 3 → way 2 → way 1 → way 0. Thereafter, the way is selected according to the order of access in the program. Since the replacement will not be correct if the LRU gets an inappropriate value, the address array write function can be used to rewrite. When this is done, be sure not to write a value other than 0 as the LRU information.

When the OD bit or ID bit in CCR is 1, cache replacement is not performed even if a cache miss occurs during data read or instruction fetch. Instead of replacing, the missed address data is read and directly transferred to the CPU.

The two-way mode of the cache set by CCR's TW bit can only be implemented by replacing ways 2 and 3. Comparisons of address array tag addresses are carried out on all four ways even in two-way mode, so the valid bits of ways 1 and 0 must be cleared to 0 before beginning operation in two-way mode.

Writing for the tag address and valid bit for cache replacement does not wait for the read from memory to be completed. If a memory access is aborted due to a reset, etc., during replacement, there will be a discrepancy between the cache contents and memory contents, so a purge must be performed.

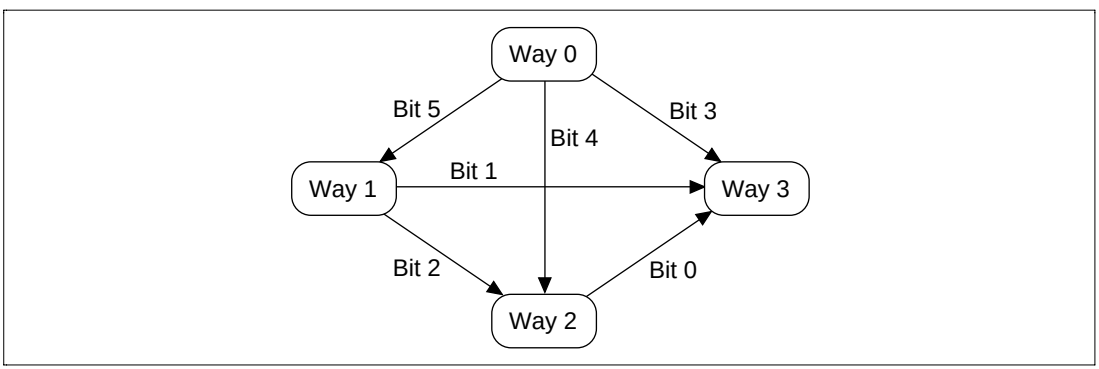



**Figure 8.7 LRU Information and Access Sequence**

**Table 8.3 LRU Information after Update**

| | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|
| Way 0 | 0 | 0 | 0 | — | — | — |
| Way 1 | 1 | — | — | 0 | 0 | — |
| Way 2 | — | 1 | — | 1 | — | 0 |
| Way 3 | — | — | 1 | — | 1 | 1 |

Note: —: Holds the value before update.

**Table 8.4 Selection Conditions for Replaced Way**

| | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|
| Way 0 | 1 | 1 | 1 | — | — | — |
| Way 1 | 0 | — | — | 1 | 1 | — |
| Way 2 | — | 0 | — | 0 | — | 1 |
| Way 3 | — | — | 0 | — | 0 | 0 |

Note: —: Don’t care.

### 8.4.6 Cache Initialization

Purges of the entire cache area can only be carried out by writing 1 to the CP bit in CCR. Writing 1 to the CP bit initializes the valid bit of the address array, and all bits of the LRU information, to 0. Cache purges are completed in 1 cycle, but additional time is required for writing to CCR. Always initialize the valid bit and LRU before enabling the cache.

When the cache is enabled, instructions are read from the cache even during writing to CCR. This means that the prefetched instructions are read from the cache. To do a proper purge, write 0 to CCR’s CE bit, then disable the cache and purge. Since CCR’s CE bit is cleared to 0 by a power-on reset or manual reset, the cache can be purged immediately by a reset.

### 8.4.7 Associative Purges

Associative purges invalidate 1 line (16 bytes) corresponding to specific address contents when the contents are in the cache.

When the contents of a shared address are rewritten by one CPU in a multiprocessor configuration, that address must be invalidated in the cache of the other CPU if it is present there.

When writing to or reading the address obtained by adding H'40000000 to the address to be purged, the valid bit of the entry storing the address prior to addition are initialized to 0.

16 bytes are purged in each write, so a purge of 256 bytes of consecutive areas can be accomplished in 16 writes. Access sizes when associative purges are performed should be longword. A purge of 1 line requires 2 cycles.

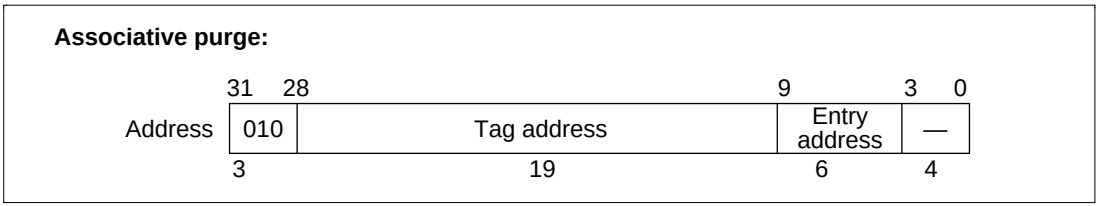



**Figure 8.8 Associative Purge Access**

### 8.4.8 Data Array Access

The cache data array can be read or written directly via the data array read/write area. Byte, word, or longword access can be used on the data array. Data array accesses are completed in 1 cycle for a read and 2 cycles for a write. Since only the cache bus is used, the operation can proceed in parallel even when another master, such as the DMAC, is using the bus. The data array of way 0 is mapped on H'C0000000 to H'C00003FF, way 1 on H'C0000400 to H'C00007FF, way 2 on H'C0000800 to H'C0000BFF and way 3 on H'C0000C00 to H'C0000FFF. When the two-way mode is being used, the area H'C0000000 to H'C00007FF is accessed as 2 kbytes of on-chip RAM. When the cache is disabled, the area H'C0000000 to H'C0000FFF can be used as 4 kbytes of on-chip RAM.

When the contents of the way being used as cache are rewritten using a data array access, the contents of external memory and cache will not match, so this operation should be avoided.

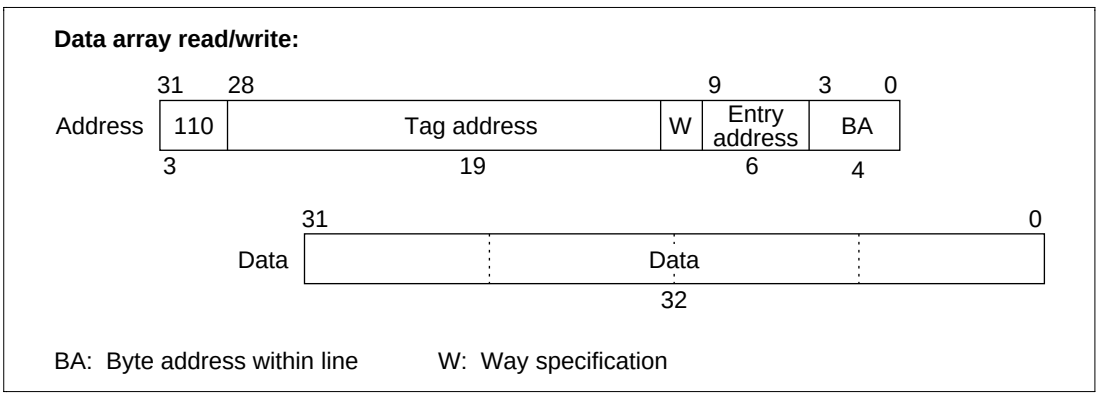



**Figure 8.9 Data Array Access**

### 8.4.9 Address Array Access

The address array of the cache can be accessed so that the contents fetched to the cache can be checked for purposes of program debugging or the like. The address array is mapped on H'60000000 to H'600003FF. Since all of the ways are mapped to the same addresses, ways are selected by rewriting the W1 and W0 bits in CCR. The address array can only be accessed in longwords.

When the address array is read, the tag address, LRU information, and valid bit are output as data. When the address array is written to, the tag address, and valid bit are written from the cache address bus. The write address must therefore be calculated before the write is performed. LRU information is written from the cache data bus, but 0 must always be written to prevent malfunctions.

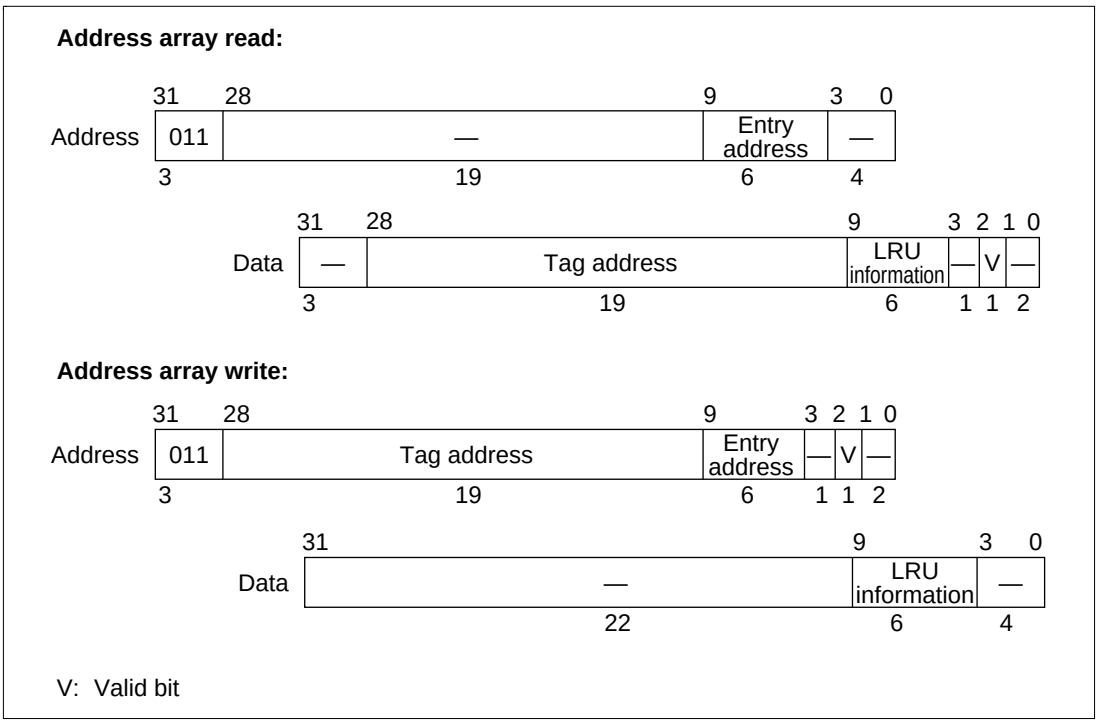



**Figure 8.10 Address Array Access**

## 8.5 Cache Use

### 8.5.1 Initialization

Cache memory is not initialized in a reset. Therefore, the cache must be initialized by software before use. The cache is initialized by zeroizing all address array valid bits and LRU information. The address array write function can be used to initialize each line, but it is simpler to initialize it once by writing 1 to the CP bit in CCR. Figure 8.11 shows how to initialize the cache.

```
MOV.W  #H'FE92, R1
MOV.B  @R1, R0     ;
AND    #H'FE, R0   ;
MOV.B  #R0, @R1    ; Cache disable
OR     #H'10, R0
MOV.B  R0, @R1     ; Cache purge
OR     #H'01, R0
MOV.B  R0, R1      ; Cache enable
```

**Figure 8.11 Cache Initialization**

### 8.5.2 Purge of Specific Lines

There is no snoop function (for monitoring data rewrites), so specific lines of cache must be purged when the contents of cache memory and external memory differ as a result of an operation. For instance, when a DMA transfer is performed to the cache area, cache lines corresponding to the rewritten address area must be purged. All entries of the cache can be purged by setting the CP bit in CCR to 1. However, it is efficient to purge only specific lines if only a limited number of entries are to be purged.

An associative purge is used to purge specific lines. Since cache lines are 16 bytes long, purges are performed in a 16-byte units. The four ways are checked simultaneously, and only lines holding data corresponding to specified addresses are purged. When addresses do not match, the data at the specified address is not fetched to the cache, so no purge occurs.

```
; Purging 32 bytes from address R3
MOV.L       #H'40000000, R0
XOR         R1, R1
MOV.L       R1, @(R0, R3)
ADD         #16, R3
MOV.L       R1, @(R0, R3)
```

**Figure 8.12 Purging Specific Addresses**

When it is troublesome to purge the cache after every DMA transfer, it is recommended that the OD bit in CCR be set to 1 in advance. When the OD bit is 1, the cache operates as cache memory only for instructions. However, when data is already fetched into cache memory, specific lines of cache memory must be purged for DMA transfers.

### 8.5.3 Cache Data Coherency

The cache memory does not have a snoop function. This means that when data is shared with a bus master other than the CPU, software must be used to ensure the coherency of data. For this purpose, the cache-through area can be used, or a cache purge can be performed with program logic.

When the cache-through area is to be used, the data shared by the bus masters is placed in the cache-through area. This makes it easy to maintain data coherency, since access of the cache-through area does not fetch data into the cache. When the shared data is accessed repeatedly and the frequency of data rewrites is low, a lower access speed can adversely affect performance.

To purge the cache using program logic, the data updates are detected by the program flow and the cache is then purged. For example, if the program inputs data from a disk, whenever reading of a unit (such as a sector) is completed, the buffer address used for reading or the entire cache is purged, thereby maintaining coherency. When data is to be exchanged between two processors, only flags that provide mutual notification of completion of data preparation or completion of a fetch are placed in the cache-through area. The data actually to be transferred is placed in the cache area and the cache is purged before the first data read to maintain the coherency of the data. When semaphores are used as the means of communication, data coherency can be maintained even when the cache is not purged by utilizing the TAS instruction. Direct external access must always be used for a TAS instruction read.

When the update unit is small, specific addresses can be purged, so only the relevant addresses are purged. When the update unit is larger, it is faster to purge the entire cache rather than purging all the addresses in order, and then read the data that previously existed in the cache again from external memory.

### 8.5.4 Two-Way Cache Mode

The 4-kbyte cache can be used as 2-kbyte RAM and 2-kbyte mixed instruction/data cache memory by setting the TW bit in CCR to 1. Ways 2 and 3 become cache, and ways 0 and 1 become RAM.

Initialization is performed by writing 1 to the CP bit in CCR, in the same way as with 4 ways. The valid bit, and LRU bits are cleared to 0.

When LRU information is initialized to zero, the initial order of use is way 3 → way 2. Thereafter, way 3 or way 2 is selected for replacement in accordance with the LRU information. The conditions for updating the LRU information are the same as for four-way mode, except that the number of ways is two.

When designated as 2-kbyte RAM, ways 0 and 1 are accessed by data array access. Figure 8.13 shows the address mapping.

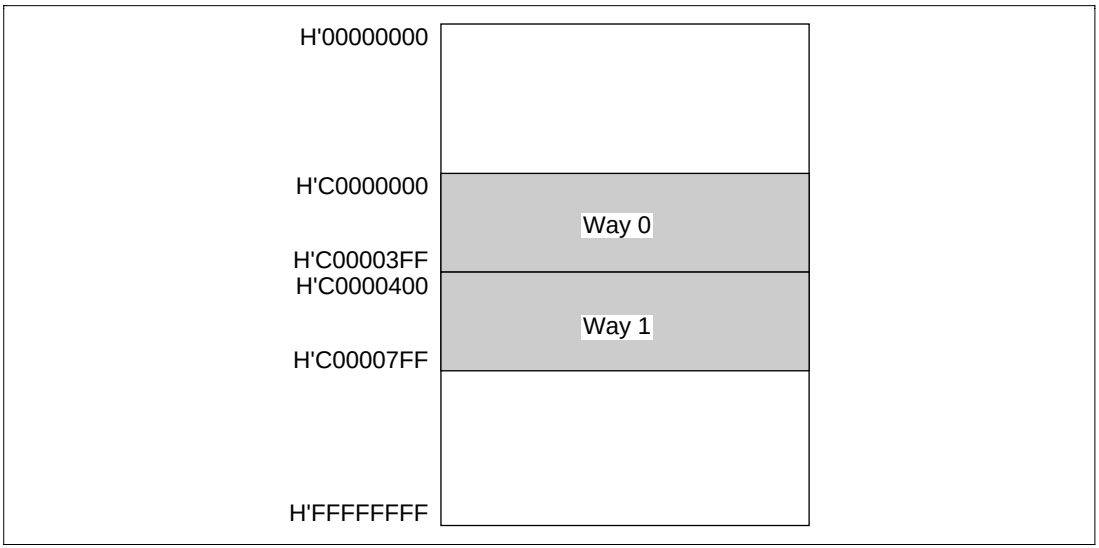



**Figure 8.13 Address Mapping of 2-kbyte RAM in the Two-Way Mode**

## 8.6 Usage Notes

### 8.6.1 Standby

Disable the cache before entering the standby mode for power-down operation. After returning from standby, initialize the cache before use.

### 8.6.2 Cache Control Register

Changing the contents of CCR also changes cache operation. The chip makes full use of pipeline operations, so it is difficult to synchronize access. For this reason, change the contents of the cache control register simultaneously when disabling the cache or after the cache is disabled. After changing the CCR contents, perform a CCR read.

### 8.6.3 Cache Initialization

The program that initializes the cache must not be located in the cache.

# Section 9 Direct Memory Access Controller (DMAC)

## 9.1 Overview

A two-channel direct memory access controller (DMAC) is included on-chip. The DMAC can be used in place of the CPU to perform high-speed data transfers between external devices equipped with DACK (transfer request acknowledge signal), external memories, on-chip memory, and memory-mapped external devices. Using the DMAC reduces the burden on the CPU and increases the operating efficiency of the chip as a whole.

### 9.1.1 Features

The DMAC has the following features:

- Two channels
- Address space: Architecturally 4 Gbytes
- Choice of data transfer unit: Byte, word (2-byte), longword (4-byte) or 16-byte unit (In a 16-byte transfer, four longword reads are executed, followed by four longword writes.)
- Maximum of 16,777,216 (16M) transfers
- In the event of a cache hit, CPU instruction processing and DMA operation can be executed in parallel
- Single address mode transfers: Either the transfer source or transfer destination (peripheral device) is accessed by a DACK signal (selectable) while the other is accessed by address. One transfer unit of data is transferred in one bus cycle.

  Possible transfer devices: External devices with DACK and memory-mapped external devices (including external memory)
- Dual address mode transfer: Both the transfer source and transfer destination are accessed by address. One transfer unit of data is transferred in two bus cycles.

  Possible transfer devices:
  - Two external memories
  - External memory and memory-mapped external device
  - Two memory-mapped external devices
  - On-chip memory and memory-mapped external device
  - Two on-chip memories
  - On-chip memory and external memory
- Transfer requests
  - External request: from the DREQ pins. Edge or level detection, and active-low or active-high mode, can be specified for DREQ.
  - On-chip peripheral module requests: 16-bit timer pulse unit (TPU)

— Auto-request: the transfer request is generated automatically within the DMAC

- Choice of bus mode
  - — Cycle steal mode
  - — Burst mode
- Choice of channel priority order
  - — Fixed mode
  - — Round robin mode
- An interrupt request can be sent to the CPU on completion of data transfer

### 9.1.2 Block Diagram

Figure 9.1 shows the DMAC block diagram.

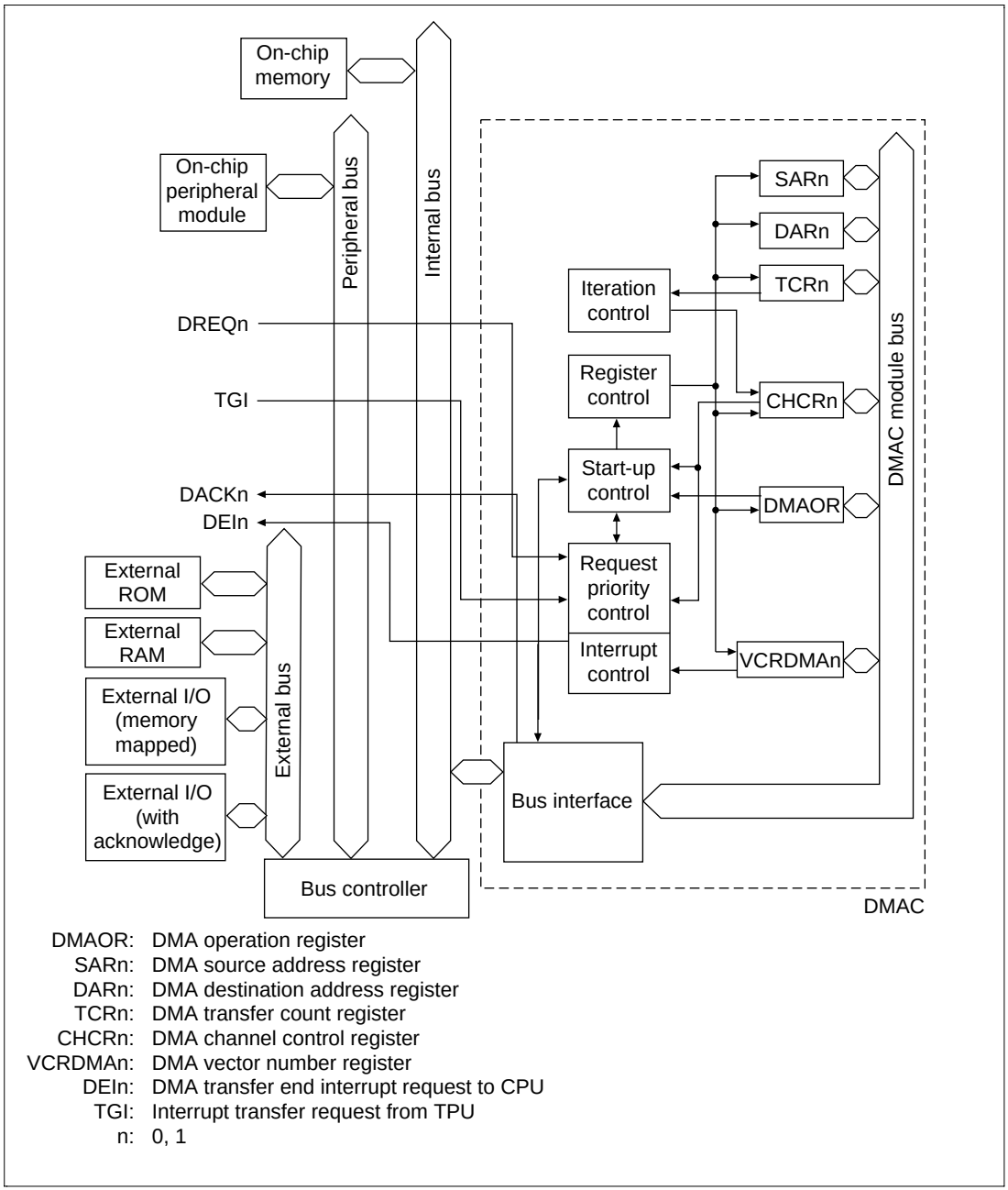



**Figure 9.1 DMAC Block Diagram**

### 9.1.3 Pin Configuration

Table 9.1 shows the DMAC pin configuration.

**Table 9.1 Pin Configuration**

| Channel | Name | Symbol | I/O | Function |
|---|---|---|---|---|
| 0 | DMA transfer request | DREQ0 | I | DMA transfer request input from external device to channel 0 |
| | DMA transfer request acknowledge | DACK0 | O | DMA transfer request acknowledge output from channel 0 to external device |
| 1 | DMA transfer request | DREQ1 | I | DMA transfer request input from external device to channel 1 |
| | DMA transfer request acknowledge | DACK1 | O | DMA transfer request acknowledge output from channel 1 to external device |

### 9.1.4 Register Configuration

Table 9.2 summarizes the DMAC registers. The DMAC has a total of 13 registers. Each channel has six control registers. One control register is shared by both channels.

**Table 9.2 Register Configuration**

| Channel | Name | Abbr. | R/W | Initial Value | Address | Access Size[*3] |
|---|---|---|---|---|---|---|
| 0 | DMA source address register 0 | SAR0 | R/W | Undefined | H'FFFFFF80 | 32 |
| | DMA destination address register 0 | DAR0 | R/W | Undefined | H'FFFFFF84 | 32 |
| | DMA transfer count register 0 | TCR0 | R/W | Undefined | H'FFFFFF88 | 32 |
| | DMA channel control register 0 | CHCR0 | R/(W)[*1] | H'00000000 | H'FFFFFF8C | 32 |
| | DMA vector number register 0 | VCRDMA0 | R/W | Undefined | H'FFFFFFA0 | 32 |
| | DMA request/response selection control register 0 | DRCR0 | R/W | H'00 | H'FFFFFE71 | 8[*3] |
| 1 | DMA source address register 1 | SAR1 | R/W | Undefined | H'FFFFFF90 | 32 |
| | DMA destination address register 1 | DAR1 | R/W | Undefined | H'FFFFFF94 | 32 |
| | DMA transfer count register 1 | TCR1 | R/W | Undefined | H'FFFFFF98 | 32 |
| | DMA channel control register 1 | CHCR1 | R/(W)[*1] | H'00000000 | H'FFFFFF9C | 32 |
| | DMA vector number register 1 | VCRDMA1 | R/(W) | Undefined | H'FFFFFFA8 | 32 |
| | DMA request/response selection control register 1 | DRCR1 | R/(W) | H'00 | H'FFFFFE72 | 8[*3] |
| All | DMA operation register | DMAOR | R/(W)[*2] | H'00000000 | H'FFFFFFB0 | 32 |

Notes: 1. Only 0 can be written to bit 1 of CHCR0 and CHCR1, after reading 1, to clear the flags.
2. Only 0 can be written to bits 1 and 2 of the DMAOR, after reading 1, to clear the flags.
3. Access DRCR0 and DRCR1 in byte units. Access all other registers in longword units.

## 9.2 Register Descriptions

### 9.2.1 DMA Source Address Registers 0 and 1 (SAR0, SAR1)

| Bit: | 31 | 30 | 29 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

DMA source address registers 0 and 1 (SAR0 and SAR1) are 32-bit read/write registers that specify the source address of a DMA transfer. During a DMA transfer, these registers indicate the next source address. (In single-address mode, SAR is ignored in transfers from external devices with DACK to memory-mapped external devices or external memory). In 16-byte unit transfers, always set the value of the source address to a 16-byte boundary (16n address). Operation results

cannot be guaranteed if other values are used. Values are retained in a reset, in standby mode, and when the module standby function is used.

### 9.2.2 DMA Destination Address Registers 0 and 1 (DAR0, DAR1)

| Bit: | 31 | 30 | 29 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

DMA destination address registers 0 and 1 (DAR0 and DAR1) are 32-bit read/write registers that specify the destination address of a DMA transfer. During a DMA transfer, these registers indicate the next destination address. (In single-address mode, DAR is ignored in transfers from memory-mapped external devices or external memory to external devices with DACK). Values are retained in a reset, in standby mode, and when the module standby function is used.

When performing 16-byte-unit transfer, a 16-byte boundary (address 16n) value must be set for the destination address.

### 9.2.3 DMA Transfer Count Registers 0 and 1 (TCR0, TCR1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 23 | 22 | 21 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

DMA transfer count registers 0 and 1 (TCR0 and TCR1) are 32-bit read/write registers that specify the DMA transfer count. The lower 24 of the 32 bits are valid. The value is written as 32 bits, including the upper eight bits. The number of transfers is 1 when the setting is H'00000001, 16,777,215 when the setting is H'00FFFFFF and 16, 777,216 (the maximum) when H'00000000 is set. During a DMA transfer, these registers indicate the remaining transfer count.

If the amount of transfer data is n bytes, the number of transfers should be set as follows.

| Transfer Unit | Set Number of Transfers |
|---|---|
| Byte | n |
| Word | n/2 |
| Longword | n/4 |
| 16 bytes | n/4 |

Set the initial value as the write value in the upper eight bits. These bits always read 0. Values are retained in a reset, in standby mode, and when the module standby function is used.

### 9.2.4 DMA Channel Control Registers 0 and 1 (CHCR0, CHCR1)

| Bit: | 31 | 30 | 29 | … | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | … | — | — | — | — |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | DM1 | DM0 | SM1 | SM0 | TS1 | TS0 | AR | AM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | AL | DS | DL | TB | TA | IE | TE | DE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/(W)* | R/W |

Note: Only 0 can be written, to clear the flag.

DMA channel control registers 0 and 1 (CHCR0 and CHCR1) are 32-bit read/write registers that control the DMA transfer mode. They also indicate the DMA transfer status. Only the lower 16 of the 32 bits are valid. They should be read and written as 32-bit values, including the upper 16 bits. Write the initial values to the upper 16 bits. These bits always read 0. The registers are initialized to H'00000000 by a reset and in standby mode. Values are retained during a module standby.

Bits 15 and 14—Destination Address Mode Bits 1, 0 (DM1, DM0): Select whether the DMA destination address is incremented, decremented or left fixed (in single address mode, DM1 and DM0 are ignored when transfers are made from a memory-mapped external device, or external

memory to an external device with DACK). DM1 and DM0 are initialized to 00 by a reset and in standby mode. Values are retained during a module standby.

| Bit 15: DM1 | Bit 14: DM0 | Description |
|---|---|---|
| 0 | 0 | Fixed destination address (Initial value) |
| | 1 | Destination address is incremented (+1 for byte transfer size, +2 for word transfer size, +4 for longword transfer size, +16 for 16-byte transfer size) |
| 1 | 0 | Destination address is decremented (–1 for byte transfer size, –2 for word transfer size, –4 for longword transfer size, –16 for 16-byte transfer size) |
| | 1 | Reserved (setting prohibited) |

Bits 13 and 12—Source Address Mode Bits 1, 0 (SM1, SM0): Select whether the DMA source address is incremented, decremented or left fixed. (In single address mode, SM1 and SM0 are ignored when transfers are made from an external device with DACK to a memory-mapped external device, or external memory.) For a 16-byte transfer, the address is incremented by +16 regardless of the SM1 and SM0 values. SM1 and SM0 are initialized to 00 by a reset and in standby mode. Values are retained during a module standby.

| Bit 13: SM1 | Bit 12: SM0 | Description |
|---|---|---|
| 0 | 0 | Fixed source address (+16 for 16-byte transfer size) (Initial value) |
| | 1 | Source address is incremented (+1 for byte transfer size, +2 for word transfer size, +4 for longword transfer size, +16 for 16-byte transfer size) |
| 1 | 0 | Source address is decremented (–1 for byte transfer size, –2 for word transfer size, –4 for longword transfer size, +16 for 16-byte transfer size) |
| | 1 | Reserved (setting prohibited) |

Bits 11 and 10—Transfer Size Bits (TS1, TS0): Select the DMA transfer size. TS1 and TS0 are initialized to 00 by a reset and in standby mode. Values are retained during a module standby.

| Bit 11: TS1 | Bit 10: TS0 | Description |
|---|---|---|
| 0 | 0 | Byte unit (initial value) |
| | 1 | Word (2-byte) unit |
| 1 | 0 | Longword (4-byte) unit |
| | 1 | 16-byte unit (4 longword transfers) |

Bit 9—Auto Request Mode Bit (AR): Selects either auto-request mode (in which transfer requests are generated automatically within the DMAC) or a mode using external requests or requests from on-chip peripheral modules. The AR bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 9: AR | Description | |
|---|---|---|
| 0 | External/on-chip peripheral module request mode | (Initial value) |
| 1 | Auto-request mode | |

Bit 8—Acknowledge/Transfer Mode Bit (AM): In dual address mode, this bit selects whether the DACK signal is output during the data read cycle or write cycle. In single-address mode, it selects whether to transfer data from memory to device or from device to memory. The AM bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 8: AM | Description | |
|---|---|---|
| 0 | DACK output in read cycle (dual address mode)/transfer from memory to device (single address mode) | (Initial value) |
| 1 | DACK output in write cycle (dual address mode)/transfer from device to memory (single address mode) | |

Bit 7—Acknowledge Level Bit (AL): Selects whether the DACK signal is an active-high signal or an active-low signal. The AL bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 7: AL | Description | |
|---|---|---|
| 0 | DACK is an active-low signal | (Initial value) |
| 1 | DACK is an active-high signal | |

Bit 6—DREQ Select Bit (DS): Selects the DREQ input detection used. The DS bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 6: DS | Description | |
|---|---|---|
| 0 | Detected by level | (Initial value) |
| 1 | Detected by edge | |

Bit 5—DREQ Level Bit (DL): Selects the DREQ input detection level. The DL bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 5: DL | Description |
|---|---|
| 0 | When DS is 0, DREQ is detected by low level; when DS is 1, DREQ is detected at falling edge (Initial value) |
| 1 | When DS is 0, DREQ is detected by high level; when DS is 1, DREQ is detected at rising edge |

Bit 4—Transfer Bus Mode Bit (TB): Selects the bus mode for DMA transfers. The TB bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 4: TB | Description |
|---|---|
| 0 | Cycle-steal mode* (Initial value) |
| 1 | Burst mode |

Note: * For 16-byte transfer in dual address mode, a burst-style operation is used.

Bit 3—Transfer Address Mode Bit (TA): Selects the DMA transfer address mode. The TA bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 3: TA | Description |
|---|---|
| 0 | Dual address mode (Initial value) |
| 1 | Single address mode |

Bit 2—Interrupt Enable Bit (IE): Determines whether or not to request a CPU interrupt at the end of a DMA transfer. When the IE bit is set to 1, an interrupt (DEI) request is sent to the CPU when the TE bit is set. The IE bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 2: IE | Description |
|---|---|
| 0 | Interrupt request disabled (Initial value) |
| 1 | Interrupt request enabled |

Bit 1—Transfer-End Flag Bit (TE): Indicates that the transfer has ended. When the value in the DMA transfer count register (TCR) becomes 0, the DMA transfer ends normally and the TE bit is set to 1. When TCR is not 0, the TE bit is not set if the transfer ends because of an NMI interrupt or DMA address error, or because the DME bit in the DMA operation register (DMAOR) or the DE bit was cleared. To clear the TE bit, read 1 from it and then write 0. When the TE bit is set, setting the DE bit to 1 will not enable a transfer. The TE bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 1: TE | Description | |
|---|---|---|
| 0 | DMA has not ended or was aborted | (Initial value) |
| | Cleared by reading 1 from the TE bit and then writing 0 | |
| 1 | DMA has ended normally (by TCR = 0) | |

Bit 0—DMA Enable Bit (DE): Enables or disables DMA transfers. In auto-request mode, the transfer starts when this bit or the DME bit in DMAOR is set to 1. The NMIF and AE bits in DMAOR and the TE bit must be all set to 0. In external request mode or on-chip peripheral module request mode, the transfer begins when the DMA transfer request is received from the relevant device or on-chip peripheral module, provided this bit and the DME bit are set to 1. As with the auto-request mode, the TE bit and the NMIF and AE bits in DMAOR must all be set to 0. The transfer can be stopped by clearing this bit to 0. The DE bit is initialized to 0 by a reset and in standby mode. Its value is retained during a module standby.

| Bit 0: DE | Description | |
|---|---|---|
| 0 | DMA transfer disabled | (Initial value) |
| 1 | DMA transfer enabled | |

### 9.2.5 DMA Vector Number Registers 0 and 1 (VCRDMA0, VCRDMA1)

| Bit: | 31 | 30 | 29 | … | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | … | — | — | — | — |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | VC7 | VC6 | VC5 | VC4 | VC3 | VC2 | VC1 | VC0 |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

DMA vector number registers 0 and 1 (VCRDMA0, VCRDMA1) are 32-bit read/write registers that set the DMAC transfer-end interrupt vector number. Only the lower eight bits of the 32 are valid. They are written as 32-bit values, including the upper 24 bits. Write the initial values to the upper 24 bits. These bits return 0 if read. Values are retained in a reset, in standby mode, and when the module standby function is used.

Bits 31 to 8—Reserved: These bits always read 0. The write value should always be 0.

Bits 7 to 0—Vector Number Bits 7–0 (VC7–VC0): Set the interrupt vector numbers at the end of a DMAC transfer. Interrupt vector numbers of 0–127 can be set. When a transfer-end interrupt

occurs, the vector number is fetched and control is transferred to the specified interrupt handling routine. The VC7–VC0 bits retain their values in a reset and in standby mode. As the maximum vector number is 127, 0 must always be written to VC7.

### 9.2.6 DMA Request/Response Selection Control Registers 0 and 1 (DRCR0, DRCR1)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | RS4 | RS3 | RS2 | RS1 | RS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W |

DMA request/response selection control registers 0 and 1 (DRCR0, DRCR1) are 8-bit read/write registers that set the DMAC transfer request source. They are written as 8-bit values. They are initialized to H'00 by a reset, but retain their values in standby mode and a module standby.

Bits 7 to 5—Reserved: These bits always read 0. The write value should always be 0.

Bits 4 to 0—Resource Select Bits 4 to 0 (RS4–RS0): Specify which transfer request to input to the DMAC. Changing the transfer request source must be done when the DMA enable bit (DE) is 0. The RS4–RS0 bits are initialized to 00 by a reset.

| Bit 4: RS4 | Bit 3: RS3 | Bit 2: RS2 | Bit 1: RS1 | Bit 0: RS0 | Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | DREQ (external request) (Initial value) |
| 0 | 1 | 1 | 0 | 0 | TPU TGI0A (on-chip TPU input capture channel 0A interrupt request)* |
| | | | | 1 | TPU TGI0B (on-chip TPU input capture channel 0B interrupt request)* |
| | | | 1 | 0 | TPU TGI0C (on-chip TPU input capture channel 0C interrupt request)* |
| | | | | 1 | TPU TGI0D (on-chip TPU input capture channel 0D interrupt request)* |

Note: * When a transfer request is generated by an on-chip module, select cycle-steal as the bus mode, dual transfer as the transfer mode, and falling edge detection for the DREQ setting. Do not use settings other than those shown above.

### 9.2.7 DMA Operation Register (DMAOR)

| Bit: | 31 | 30 | 29 | … | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | … | — | — | — | — |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | PR | AE | NMIF | DME |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/(W)* | R/(W)* | R/W |

Note: * Only 0 can be written, to clear the flag.

The DMA operation register (DMAOR) is a 32-bit read/write register that controls the DMA transfer mode. It also indicates the DMA transfer status. Only the lower four of the 32 bits are valid. DMAOR is written as a 32-bit value, including the upper 28 bits. Write the initial values to the upper 28 bits. These bits always read 0. DMAOR is initialized to H'00000000 by a reset and in standby mode. It retains its value when the module standby function is used.

Bits 31 to 4—Reserved: These bits always read 0. The write value should always be 0.

Bit 3—Priority Mode Bit (PR): Specifies whether a fixed channel priority order or round-robin mode is to be used there are simultaneous transfer requests for multiple channels. It is initialized to 0 by a reset and in standby mode. It retains its value when the module standby function is used.

| Bit 3: PR | Description |
|---|---|
| 0 | Fixed priority (channel 0 > channel 1) (Initial value) |
| 1 | Round-robin (Top priority shifts to bottom after each transfer. The priority for the first DMA transfer after a reset is channel 1 > channel 0) |

Bit 2—Address Error Flag Bit (AE): This flag indicates that an address error has occurred in the DMAC. When the AE bit is set to 1, DMA transfer cannot be enabled even if the DE bit in the DMA channel control register (CHCR) is set to 1. To clear the AE bit, read 1 from it and then write 0. Operation is performed up to the DMAC transfer being executed when the address error occurred. AE is initialized to 0 by a reset and in standby mode. It retains its value when the module standby function is used.

| Bit 2: AE | Description | |
|---|---|---|
| 0 | No DMAC address error<br>To clear the AE bit, read 1 from it and then write 0 | (Initial value) |
| 1 | Address error by DMAC | |

Bit 1—NMI Flag Bit (NMIF): This flag indicates that an NMI interrupt has occurred. When the NMIF bit is set to 1, DMA transfer cannot be enabled even if the DE bit in the DMA channel control register (CHCR) and the DME bit are set to 1. To clear the NMIF bit, read 1 from it and then write 0. Operation is completed up to the end of the DMAC transfer being executed when NMI was input. When the NMI interrupt is input while the DMAC is not operating, the NMIF bit is set to 1. The NMIF bit is initialized to 0 by a reset or in the standby mode. It retains its value when the module standby function is used.

| Bit 1: NMIF | Description | |
|---|---|---|
| 0 | No NMIF interrupt<br>To clear the NMIF bit, read 1 from it and then write 0 | (Initial value) |
| 1 | NMIF interrupt has occurred | |

Bit 0—DMA Master Enable Bit (DME): Enables or disables DMA transfers on all channels. A DMA transfer becomes enabled when the DE bit in the CHCR and the DME bit are set to 1. For this to be effective, the TE bit in CHCR and the NMIF and AE bits must all be 0. When the DME bit is cleared, all channel DMA transfers are aborted. DME is initialized to 0 by a reset and in standby mode. It retains its value when the module standby function is used.

| Bit 0: DME | Description | |
|---|---|---|
| 0 | DMA transfers disabled on all channels | (Initial value) |
| 1 | DMA transfers enabled on all channels | |

## 9.3 Operation

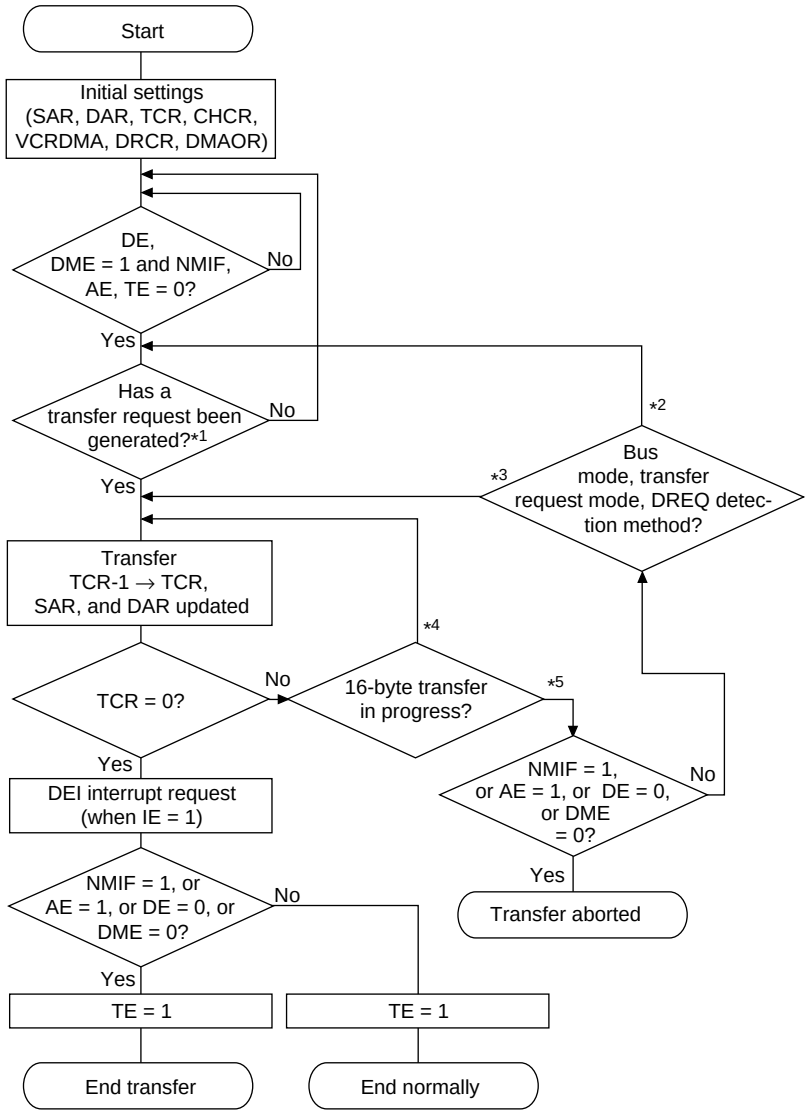
When there is a DMA transfer request, the DMAC starts the transfer according to the predetermined channel priority; when the transfer-end conditions are satisfied, it ends the transfer. Transfers can be requested in three modes: auto-request, external request, and on-chip module request. A transfer can be in either single address mode or dual address mode. The bus mode can be either burst or cycle-steal.

### 9.3.1 DMA Transfer Flow

After the DMA source address registers (SAR), DMA destination address registers (DAR), DMA transfer count registers (TCR), DMA channel control registers (CHCR), DMA vector number registers (VCRDMA), DMA request/response selection control registers (DRCR), and DMA operation register (DMAOR) are initialized (initializing sets each register so that ultimately the condition (DE = 1, DME = 1, TE = 0, NMIF = 0, AE = 0) is satisfied), the DMAC transfers data according to the following procedure:

1. Checks to see if transfer is enabled (DE = 1, DME = 1, TE = 0, NMIF = 0, AE = 0)
2. When a transfer request occurs and transfer is enabled, the DMAC transfers 1 transfer unit of data. (In auto-request mode, the transfer begins automatically after register initialization. The TCR value will be decremented by 1.) The actual transfer flows vary depending on the address mode and bus mode.
3. When the specified number of transfers have been completed (when TCR reaches 0), the transfer ends normally. If the IE bit in CHCR is set to 1 at this time, a DEI interrupt request is sent to the CPU.
4. When an address error occurs in the DMAC or an NMI interrupt is generated, the transfer is aborted. Transfers are also aborted when the DE bit in CHCR or the DME bit in DMAOR is changed to 0.

Figure 9.2 shows a flowchart illustrating this procedure.




Notes: 1. In auto-request mode, the transfer will start when the NMIF, AE, and TE bits are all 0 and the DE and DME bits are then set to 1.
2. In burst mode, DREQ = level detection (external request), or cycle-steal mode.
3. In burst mode, DREQ = edge detection (external request), or auto-request mode in burst mode.
4. 16-byte transfer cycle in progress.
5. End of a 16-byte transfer cycle.

**Figure 9.2 DMA Transfer Flow**

### 9.3.2 DMA Transfer Requests

DMA transfer requests are usually generated in either the data transfer source or destination, but they can also be generated by devices that are neither the source nor the destination. Transfers can be requested in three modes: auto-request, external request, and on-chip peripheral module request. The request mode is selected with the AR bit in DMA channel control registers 0 and 1 (CHCR0, CHCR1) and the RS0, RS1, RS2, RS3 and RS4 bits in DMA request/response selection control registers 0 and 1 (DRCR0, DRCR1).

**Table 9.3 Selecting the DMA Transfer Request Using the AR and RS Bits**

| CHCR | DRCR | | | | | | |
|---|---|---|---|---|---|---|---|
| **AR** | **RS4** | **RS3** | **RS2** | **RS1** | **RS0** | **Request Mode** | **Resource Selection** |
| 0 | 0 | 0 | 0 | 0 | 0 | External request | DREQ |
| | 0 | 0 | 1 | 0 | 0 | On-chip peripheral module request | TPU TGI0A |
| | | | | | 1 | | TPU TGI0B |
| | | | | 1 | 0 | | TPU TGI0C |
| | | | | | 1 | | TPU TGI0D |
| 1 | * | * | * | * | * | Auto-request mode | |

Note: * Don't care

**Auto-Request Mode:** When there is no transfer request signal from an external source (as in a memory-to-memory transfer, the auto-request mode allows the DMAC to automatically generate a transfer request signal internally. When the DE bits in CHCR0 and CHCR1 and the DME bit in the DMA operation register (DMAOR) are set to 1, the transfer begins (so long as the TE bits in CHCR0 and CHCR1 and the NMIF and AE bits in DMAOR are all 0).

**External Request Mode:** In this mode a transfer is started by a transfer request signal (DREQ) from an external device. Choose one of the modes shown in table 9.4 according to the application system. When DMA transfer is enabled (DE = 1, DME = 1, TE = 0, NMIF = 0, AE = 0), a transfer is performed upon input of a DREQ signal.

**Table 9.4 Selecting External Request Modes with the TA and AM Bits**

| CHCR | | Transfer | | | |
|---|---|---|---|---|---|
| **TA** | **AM** | **Address Mode** | **Acknowledge Mode** | **Source** | **Destination** |
| 0 | 0 | Dual address mode | DACK output in read cycle | Any* | Any* |
| | 1 | Dual address mode | DACK output in write cycle | Any* | Any* |
| 1 | 0 | Single address mode | Data transferred from memory to device | External memory or memory-mapped external device | External device with DACK |
| | 1 | Single address mode | Data transferred from device to memory | External device with DACK | External memory or memory-mapped external device |

Note: External memory, memory-mapped external device.

Choose to detect DREQ either by the falling edge or by level using the DS and DL bits in CHCR0 and CHCR1 (DS = 0 is level detection, DS = 1 is edge detection; DL = 0 is active-low, DL = 1 is active-high). The source of the transfer request does not have to be the data transfer source or destination.

**Table 9.5 Selecting the External Request Signal with the DS and DL Bits**

| CHCR | | |
|---|---|---|
| **DS** | **DL** | **External Request** |
| 0 | 0 | Level (active-low) |
| | 1 | Level (active-high) |
| 1 | 0 | Edge (falling) |
| | 1 | Edge (rising) |

**On-Chip Module Request Mode:** In this mode, transfers are started by the transfer request signal (interrupt request signal) of an on-chip peripheral module. Transfer request signals include interrupts from the TPU (table 9.6). If DMA transfer is enabled (DE = 1, DME = 1, TE = 0, NMIF = 0, AE = 0), DMA transfer starts upon the input of a transfer request signal.

**Table 9.6 Selecting On-Chip Peripheral Module Request Mode with the AR and RS Bits**

| AR | RS4 | RS3 | RS2 | RS1 | RS0 | DMA Transfer Request Source | DMA Transfer Request Signal | Transfer* Source | Transfer* Destination | Bus Mode | DREQ Setting |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | TPU channel 0A | TGI0A | Any | Any | Cycle-steal | Edge, active-low |
| | | | | | 1 | TPU channel 0B | TGI0B | Any | Any | Cycle-steal | Edge, active-low |
| | | | | 1 | 0 | TPU channel 0C | TGI0C | Any | Any | Cycle-steal | Edge, active-low |
| | | | | | 1 | TPU channel 0D | TGI0D | Any | Any | Cycle-steal | Edge, active-low |

Note: * Do not perform transfers between on-chip peripheral modules.

For outputting transfer request from the TPU, the corresponding interrupt enable bits must be set to output the interrupt signals. Note that transfer request signals from on-chip peripheral modules (interrupt request signals) are sent not just to the DMAC but to the CPU as well. When an on-chip peripheral module is specified as the transfer request source, set the priority level values in the interrupt priority level registers (IPRC–IPRE) of the interrupt controller (INTC) at or below the levels set in the I3–I0 bits of the CPU's status register so that the CPU does not accept the interrupt request signal.

With the DMA transfer request signals in table 9.6, when DMA transfer is performed a DMA transfer request (interrupt request) from any module will be cleared at the first transfer.

### 9.3.3 Channel Priorities

When the DMAC receives simultaneous transfer requests on two channels, it selects a channel according to a predetermined priority order. There is a choice of two priority modes, fixed or round-robin. The mode is selected by the priority bit, PR, in the DMA operation register (DMAOR).

**Fixed Priority Mode:** In this mode, the relative channel priority levels are fixed. When PR is set to 0, channel 0 has higher priority than channel 1. Figure 9.3 shows an example of a transfer in burst mode.

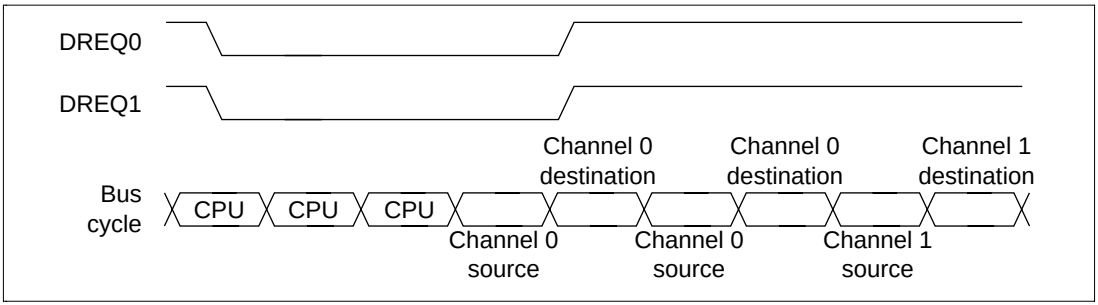



**Figure 9.3 Fixed Mode Burst DMA Transfer (Dual Address, Active-Low DREQ Level)**

In cycle-steal mode, once a channel 0 request is accepted, channel 1 requests are also accepted until the next request is accepted, which makes more effective use of the bus cycle. If requests come simultaneously for channel 0 and channel 1 when DMA operation is starting, the first is transmitted with channel 0, and thereafter channel 1 and channel 0 transfers are performed alternately.

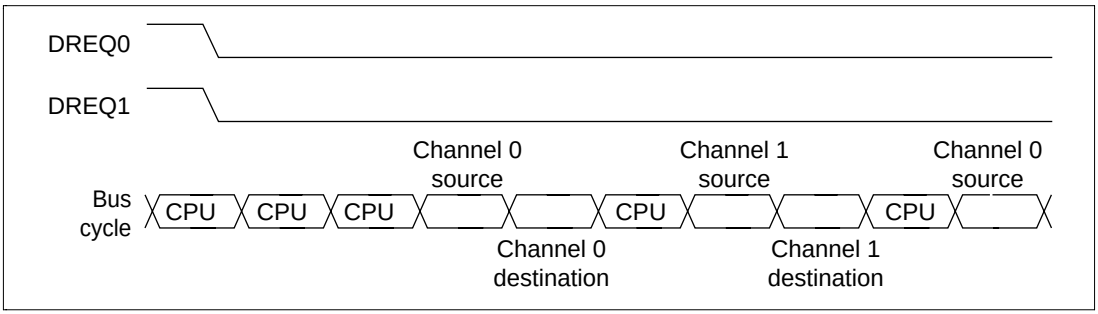



**Figure 9.4 Fixed Mode Cycle-Steal DMA Transfer (Dual Address, Active-Low DREQ Level)**

**Round-Robin Mode:** Switches the priority of channel 0 and channel 1, shifting their ability to receive transfer requests. Each time one transfer ends on one channel, the priority shifts to the other channel. The channel on which the transfer just finished is assigned low priority. After reset, channel 1 has higher priority than channel 0.

Figure 9.5 shows how the priority changes when channel 0 and channel 1 transfers are requested simultaneously and another channel 0 transfer is requested after the first two transfers end. The DMAC operates as follows:

1. Transfer requests are generated simultaneously to channels 1 and 0.
2. Channel 1 has the higher priority, so the channel 1 transfer begins first (channel 0 waits for transfer).
3. When the channel 1 transfer ends, channel 1 becomes the lower-priority channel.

4. The channel 0 transfer begins.
5. When the channel 0 transfer ends, channel 0 becomes the lower-priority channel.
6. A channel 0 transfer is requested.
7. The channel 0 transfer begins.
8. When the channel 0 transfer ends, channel 0 is already the lower-priority channel, so the order remains the same.

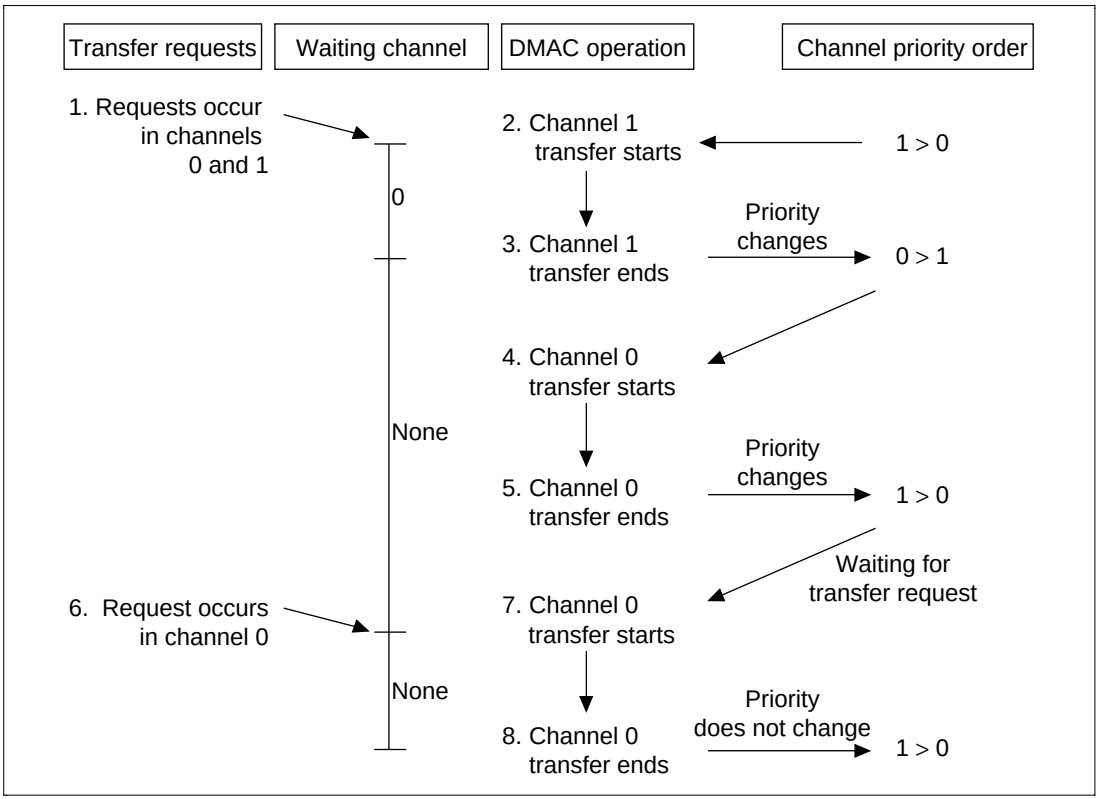



**Figure 9.5 Channel Priority in Round-Robin Mode**

### 9.3.4 DMA Transfer Types

It can operate in single address mode or dual address mode, as defined by how many bus cycles the DMAC takes to access the transfer source and transfer destination. The actual transfer operation timing varies with the DMAC bus mode used: cycle-steal mode or burst mode. The DMAC supports all the transfers shown in table 9.7.

**Table 9.7 Supported DMA Transfers**

| | Destination | | | |
|---|---|---|---|---|
| **Source** | **External Device with DACK** | **External Memory** | **Memory-Mapped External Device** | **On-Chip Memory** |
| External device with DACK | Not available | Single | Single | Not available |
| External memory | Single | Dual | Dual | Dual |
| Memory-mapped external device | Single | Dual | Dual | Dual |
| On-chip memory | Not available | Dual | Dual | Dual |

Single: Single address mode

Dual: Dual address mode

**Address Modes:**

- Single Address Mode

  In single address mode, both the transfer source and destination are external; one (selectable) is accessed by a DACK signal while the other is accessed by address. In this mode, the DMAC performs the DMA transfer in one bus cycle by simultaneously outputting a transfer request acknowledge DACK signal to one external device to access it, while outputting an address to the other end of the transfer. Figure 9.6 shows an example of a transfer between external memory and external device with DACK. That data is written in external memory in the same bus cycle while the external device outputs data to the data bus.

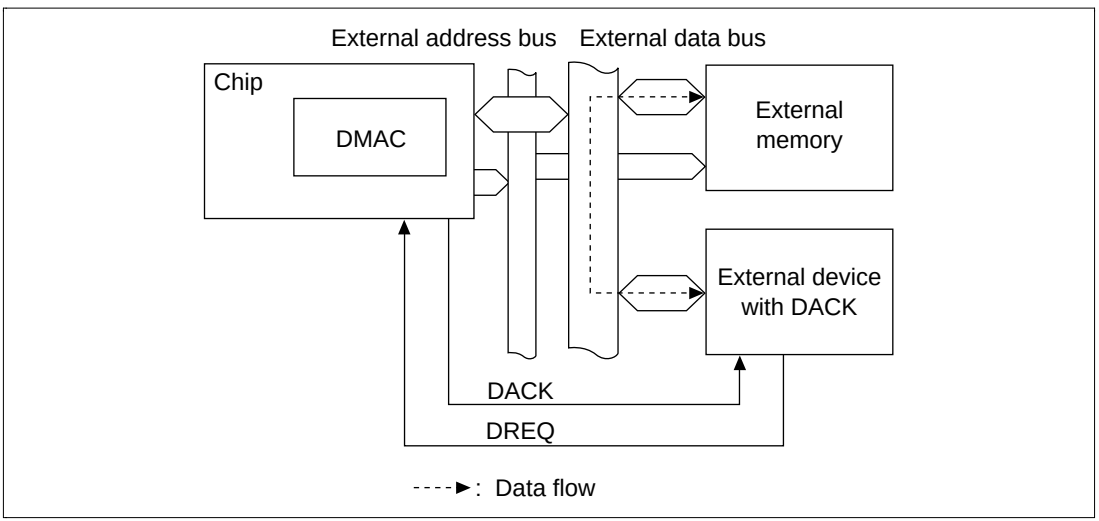



**Figure 9.6 Data Flow in Single Address Mode**

Two types of transfers are possible in single address mode: 1) transfers between external devices with DACK and memory-mapped external devices; and 2) transfers between external devices with DACK and external memory. For both of them, transfer must be requested by the external request signal (DREQ). Figure 9.7 shows the DMA transfer timing for single address mode.

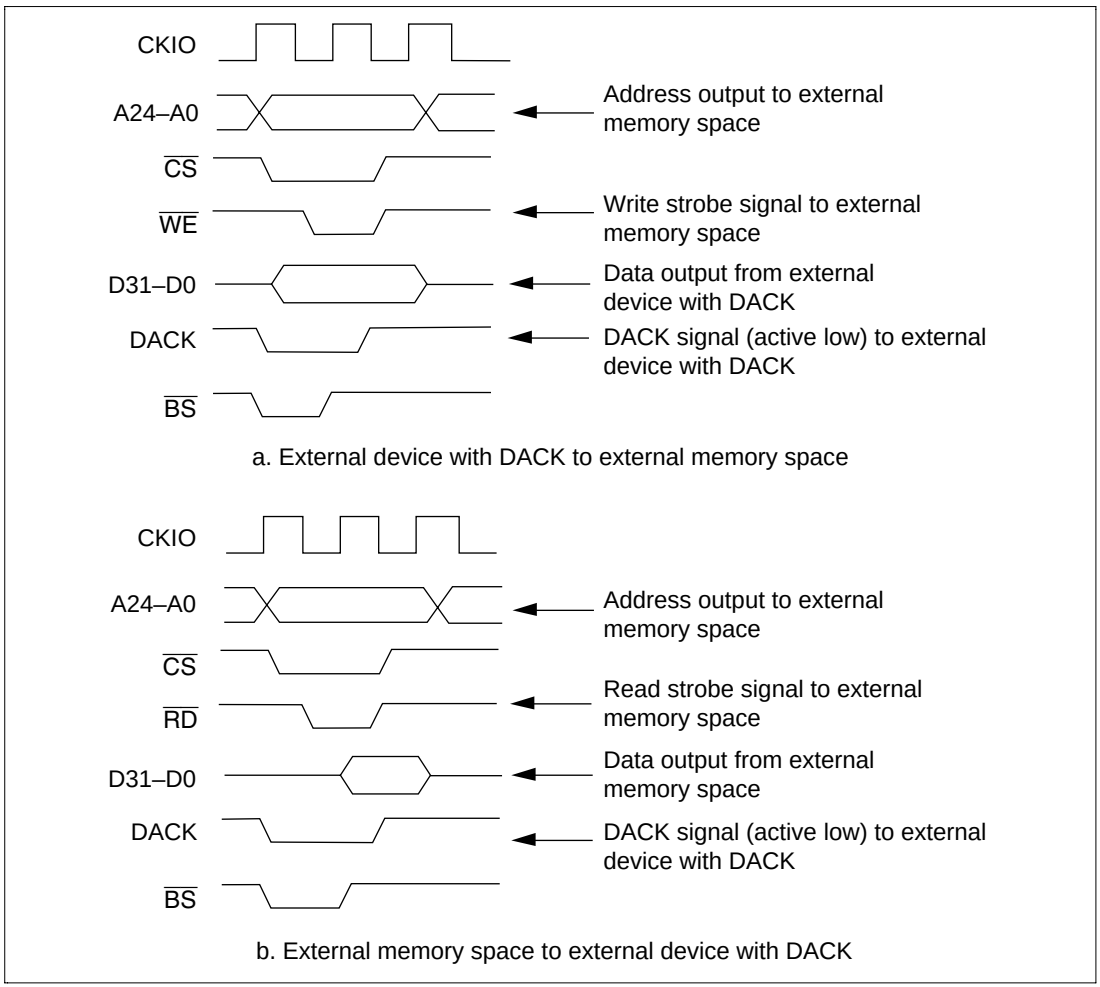



**Figure 9.7 DMA Transfer Timing in Single Address Mode**

- Dual Address Mode

  In dual address mode, both the transfer source and destination are accessed (selectable) by address. The source and destination can be located externally or internally. The DMAC accesses the source in the read cycle and the destination in the write cycle, so the transfer is performed in two separate bus cycles. The transfer data is temporarily stored in the DMAC. Figure 9.8 shows an example of a transfer between two external memories in which data is read from one external memory in the read cycle and written to the other external memory in the following write cycle.

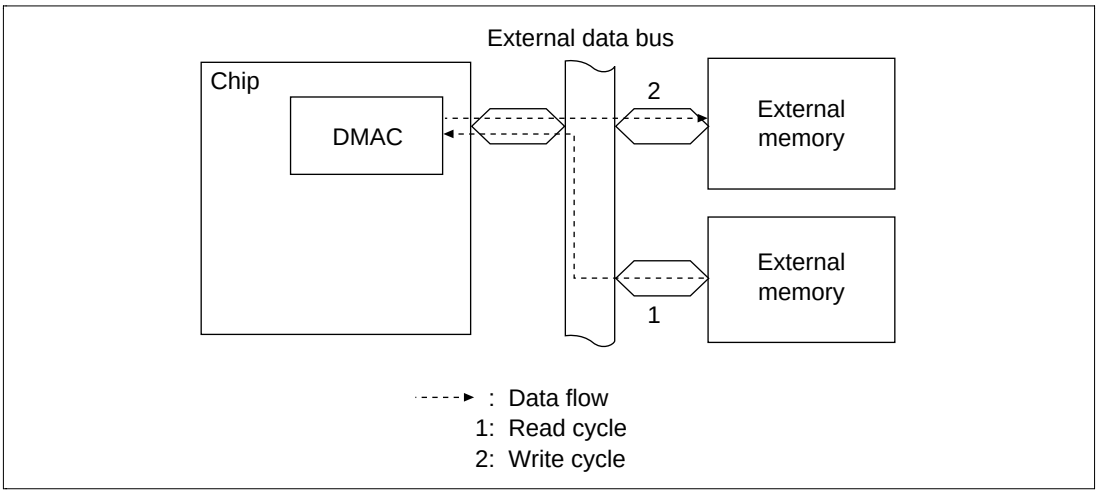



**Figure 9.8 Data Flow in Dual Address Mode**

In dual address mode transfers, external memory and memory-mapped external devices can be mixed without restriction. Specifically, this enables transfers between the following:

1. External memory and external memory.
2. External memory and memory-mapped external devices.
3. Memory-mapped external devices and memory-mapped external devices.
4. On-chip memory*[2] and on-chip memory.*[2]
5. On-chip memory*[2] and memory-mapped external devices.
6. On-chip memory*[2] and external memory.

Transfer requests can be auto-request, external requests, or on-chip peripheral module requests. (see table 9.6). Dual address mode outputs DACK in either the read cycle or write cycle. CHCR controls the cycle in which DACK is output.

Note: Specify word or byte as the on-chip memory access size.

Figure 9.9 shows the DMA transfer timing in dual address mode.

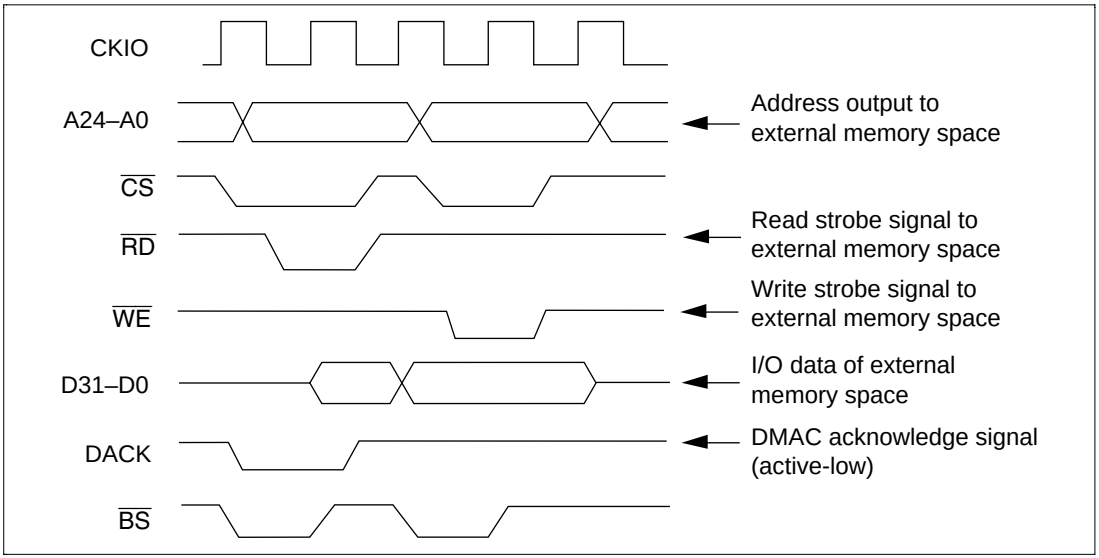



**Figure 9.9 DMA Transfer Timing in Dual Address Mode (External Memory Space → External Memory Space, DACK Output in Read Cycle)**

**Bus Modes:** There are two bus modes: cycle-steal and burst. Select the mode with the TB bits in CHCR0 and CHCR1.

- Cycle-Steal Mode

  In cycle-steal mode, the bus right is given to another bus master each time the DMAC completes one transfer. When another transfer request occurs, the bus right is retrieved from the other bus master and another transfer is performed for one transfer unit. When that transfer ends, the bus right is passed to the other bus master. This is repeated until the transfer end conditions are satisfied. (in the case of 16-byte transfer in dual address mode, the DMAC continues to hold the bus)

  Cycle-steal mode can be used with all categories of transfer destination, transfer source, and transfer request source. (with the exception of transfers between on-chip peripheral modules)

  The CPU may take the bus twice when an acknowledge signal is output during the write cycle or in single address mode. Figure 9.10 shows an example of DMA transfer timing in cycle-steal mode (dual address mode, DREQ level detection).

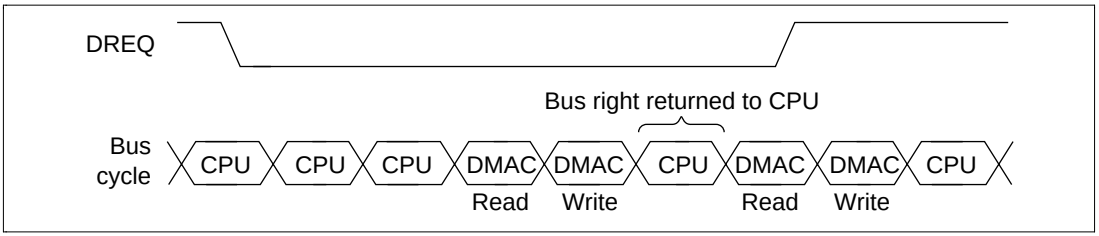



**Figure 9.10 DMA Transfer Timing in Cycle-Steal Mode (Dual Address Mode, DREQ Level Detection)**

- Burst Mode

  In burst mode, once the DMAC gets the bus, the transfer continues until the transfer end condition is satisfied. When external request mode is used with level detection of the DREQ pin, however, negating DREQ will pass the bus to the other bus master after completion of the bus cycle of the DMAC that currently has an acknowledged request, even if the transfer end conditions have not been satisfied. When the transfer request source is an on-chip peripheral module, however, cycle-steal mode is always used.

  Figure 9.11 shows an example of DMA transfer timing in burst mode (single address mode, DREQ level detection).

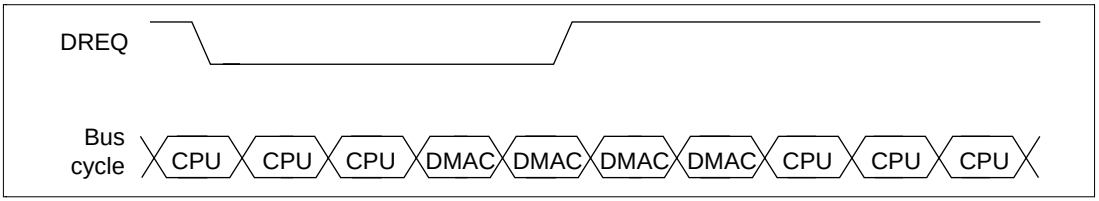



**Figure 9.11 DMA Transfer Timing in Burst Mode (Single Address Mode, DREQ Level Detection)**

Refreshes cannot be performed during a burst transfer, so ensure that the number of transfers satisfies the refresh request period when a memory requiring refreshing is used.

**Relationship of Request Modes and Bus Modes by DMA Transfer Category:** Table 9.8 shows the relationship between request modes, bus modes, etc., by DMA transfer category.

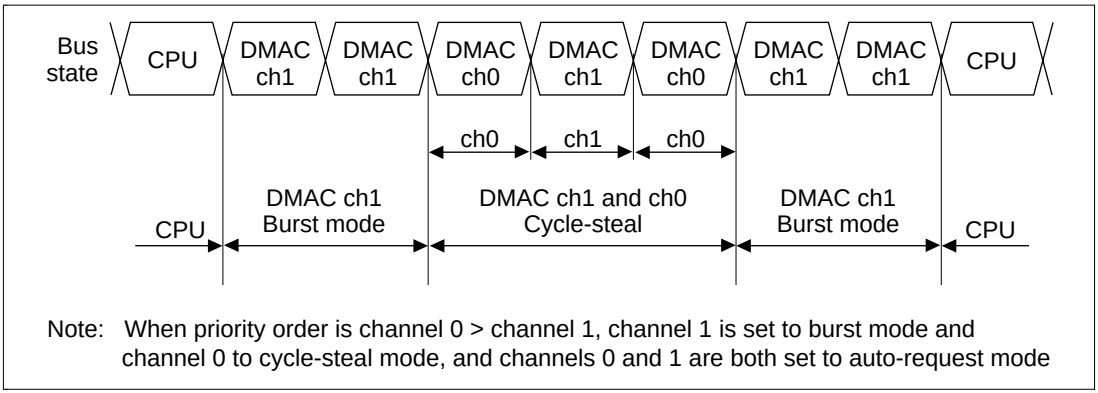
**Table 9.8 Relationship of Request Modes and Bus Modes by DMA Transfer Category**

| Address Mode | Transfer Category | Request Mode | Bus Mode | Transfer Size (Bytes) |
|---|---|---|---|---|
| Single | External device with DACK and external memory | External | B/C | 1/2/4/16 |
| | External device with DACK and memory-mapped external device | External | B/C | 1/2/4/16 |
| Dual | External memory and external memory | All | B/C | 1/2/4/16 |
| | External memory and memory-mapped external device | All | B/C | 1/2/4/16 |
| | Memory-mapped external device and memory-mapped external device | All | B/C | 1/2/4/16 |
| | On-chip memory and on-chip memory | Auto | B/C | 1/2 |
| | On-chip memory and memory-mapped external device | All*1 | B/C | 1/2 |
| | On-chip memory and external memory | All*2 | B/C | 1/2 |

B: Burst, C: Cycle-steal

Notes: 1. In transfer from on-chip memory to a memory-mapped external device, set DACK for write-time output.
In transfer from a memory-mapped external device to on-chip memory, set DACK for read-time output.
2. In transfer from on-chip memory to external memory, set DACK for write-time output.
In transfer from external memory to on-chip memory, set DACK for read-time output.

**Bus Mode and Channel Priority:** When a given channel (1) is transferring in burst mode and there is a transfer request to a channel (0) with a higher priority, the transfer of the channel with higher priority (0) will begin immediately. When channel 0 is also operating in the burst mode, the channel 1 transfer will continue as soon as the channel 0 transfer has completely finished. When channel 0 is in cycle-steal mode, channel 1 will begin operating again after channel 0 completes the transfer of one transfer unit, but the bus will then switch between the two in the order channel 1, channel 0, channel 1, channel 0. Since channel 1 is in burst mode, it will not give the bus to the CPU. If the channel in cycle-steal mode is using external requests, burst mode channel transfer may be executed twice in succession. This example is illustrated in Figure 9.12.




Note: When priority order is channel 0 > channel 1, channel 1 is set to burst mode and channel 0 to cycle-steal mode, and channels 0 and 1 are both set to auto-request mode

**Figure 9.12 Bus Status when Multiple Channels are Operating**

### 9.3.5 Number of Bus Cycles

The number of states in the bus cycle when the DMAC is the bus master is controlled by the bus state controller (BSC) just as it is when the CPU is the bus master. For details, see section 7, Bus State Controller (BSC).

### 9.3.6 DMA Transfer Request Acknowledge Signal Output Timing

DMA transfer request acknowledge signal DACKn is output synchronous to the DMAC address output specified by the channel control register AM bit. Normally, the acknowledge signal becomes valid when DMA address output begins, and becomes invalid 0.5 cycles before the address output ends. (See figure 9.13.) The output timing of the acknowledge signal varies with the settings of the connected memory space. The output timing of acknowledge signals in the memory spaces is shown in figure 9.13.

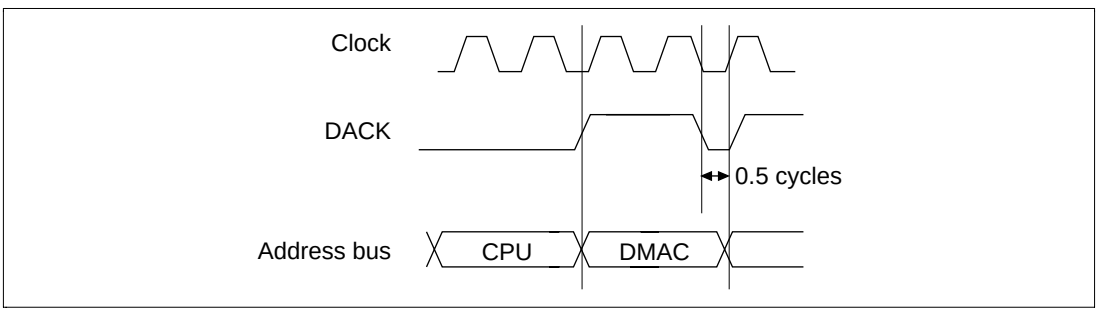



**Figure 9.13 Example of DACK Output Timing**

**Acknowledge Signal Output when External Memory Is Set as Ordinary Memory Space:** The timing at which the acknowledge signal is output is the same in the DMA read and write cycles specified by the AM bit (figures 9.14 and 9.15). When DMA address output begins, the acknowledge signal becomes valid; 0.5 cycles before address output ends, it becomes invalid. If a wait is inserted in this period and address output is extended, the acknowledge signal is also extended.

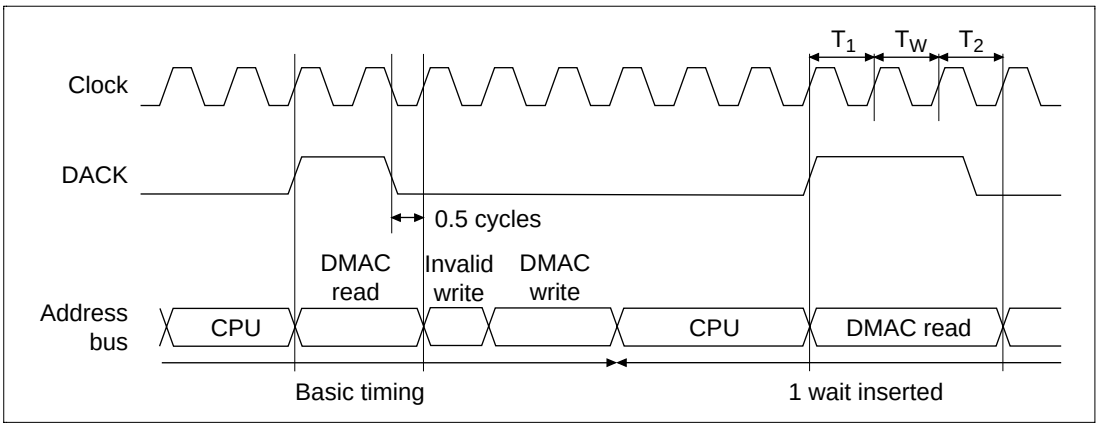



**Figure 9.14 DACK Output in Ordinary Space Accesses (AM = 0)**

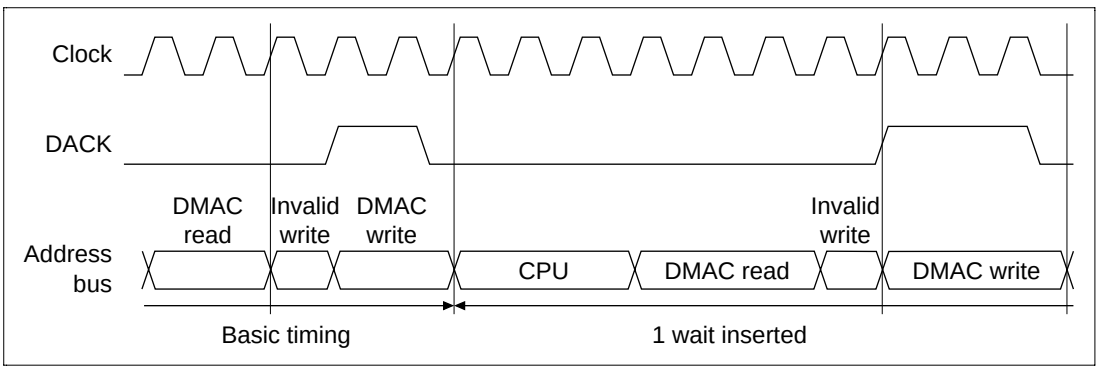



**Figure 9.15 DACK Output in Ordinary Space Accesses (AM = 1)**

In a longword access of a 16-bit external device (figure 9.16) or an 8-bit external device (figure 9.17), or a word access of an 8-bit external device (figure 9.18), the lower and upper addresses are output 2 and 4 times in each DMAC access in order to align the data. For all of these addresses, the acknowledge signal becomes valid simultaneous with the start of output and the signal becomes invalid 0.5 cycles before the address output ends. When multiple addresses are output in a single access to align data for synchronous DRAM, DRAM, or burst ROM, an acknowledge signal is output to those addresses as well.

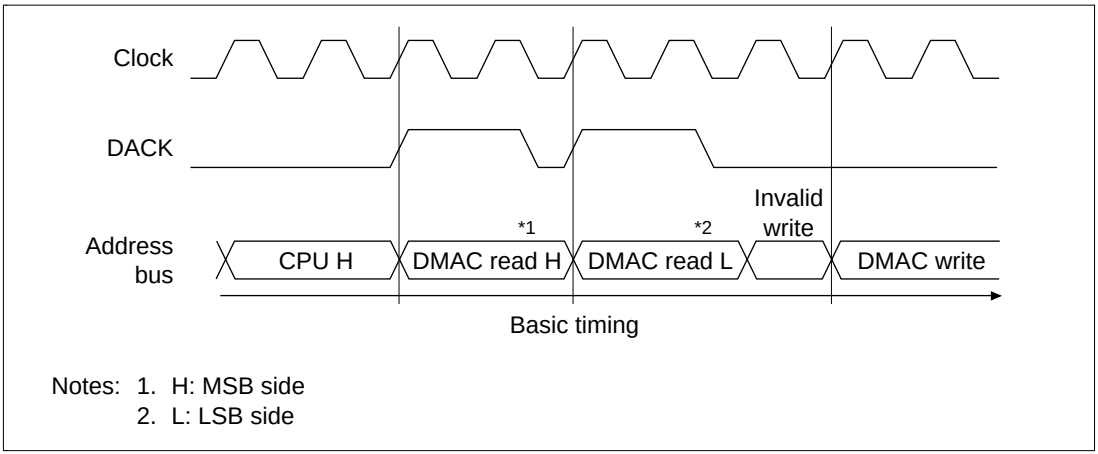



**Figure 9.16 DACK Output in Ordinary Space Accesses (AM = 0, Longword Access to 16-Bit External Device)**

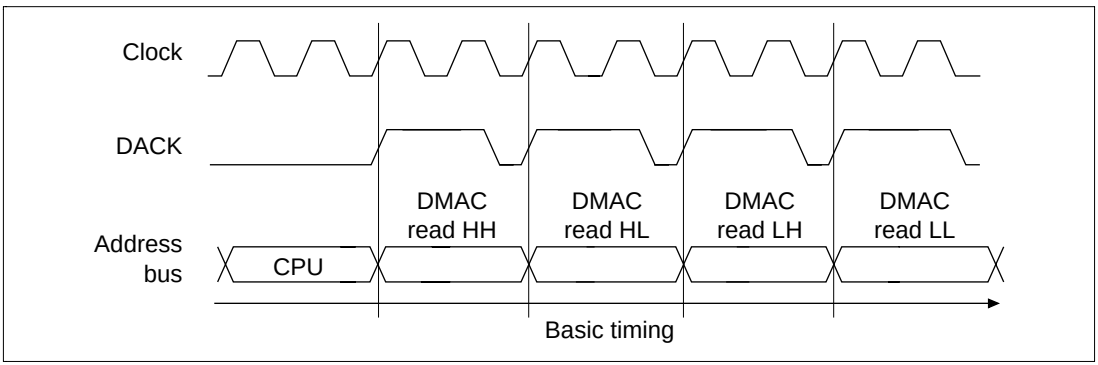



**Figure 9.17 DACK Output in Ordinary Space Accesses (AM = 0, Longword Access to 8-Bit External Device)**

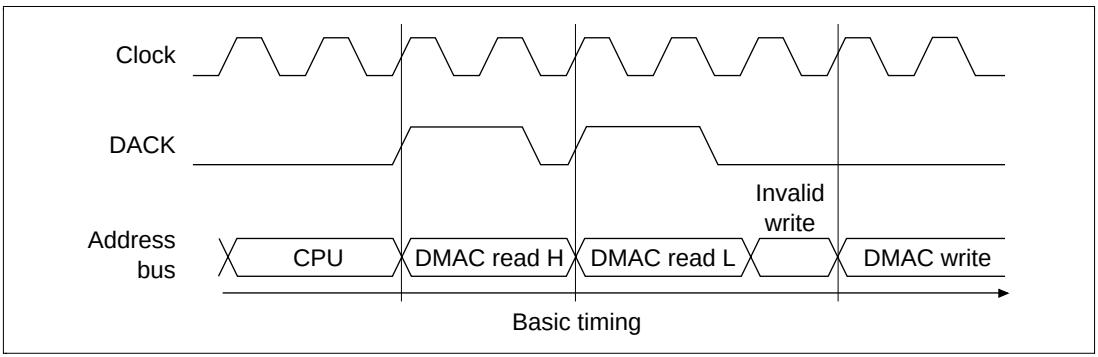



**Figure 9.18 DACK Output in Ordinary Space Accesses (AM = 0, Word Access to 8-Bit External Device)**

**Acknowledge Signal Output when External Memory Is Set as Synchronous DRAM:** When external memory is set as synchronous DRAM, DACK output becomes valid simultaneously with the start of the DMA address, and becomes invalid when the address output ends.

When external memory is set as synchronous DRAM auto-precharge and AM = 0, the acknowledge signal is output across the row address, read command, wait and read address of the DMAC read (figure 9.19). Since the synchronous DRAM read has only burst mode, during a single read an invalid address is output; the acknowledge signal, however, is output on the same timing (figure 9.20). At this time, the acknowledge signal is extended until the write address is output after the invalid read. When AM = 1, the acknowledge signal is output across the row address and column address of the DMAC write (figure 9.21).

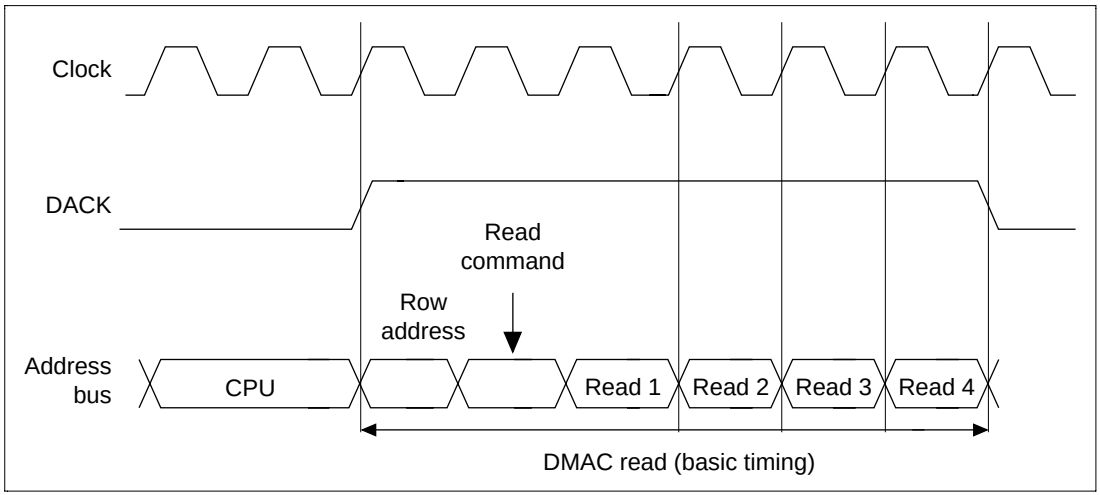



**Figure 9.19 DACK Output in Synchronous DRAM Burst Read (Auto-Precharge, AM = 0)**

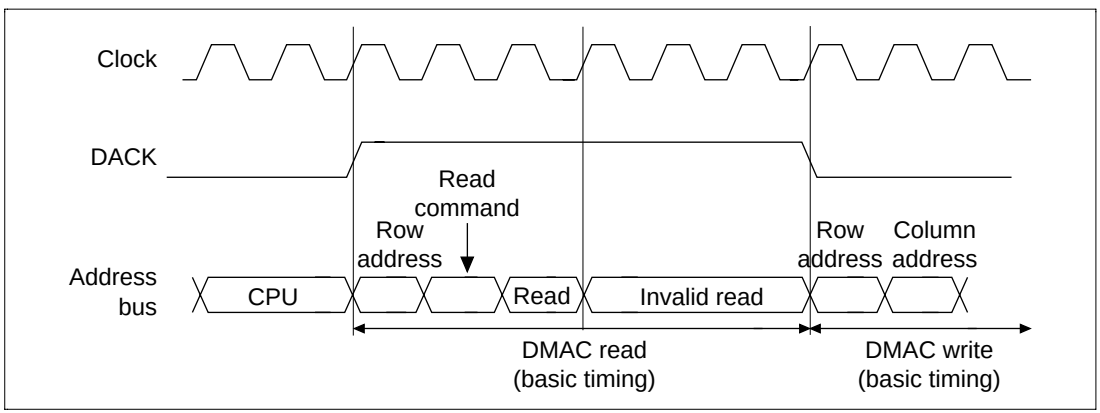



**Figure 9.20 DACK Output in Synchronous DRAM Single Read (Auto-Precharge, AM = 0)**

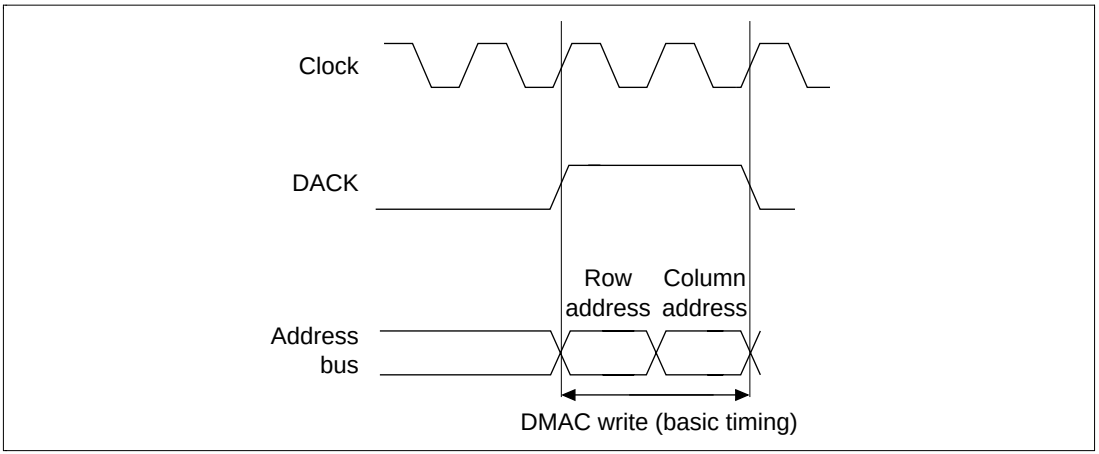



**Figure 9.21 DACK Output in Synchronous DRAM Write (Auto-Precharge, AM = 1)**

When external memory is set as bank active synchronous DRAM, during a burst read the acknowledge signal is output across the read command, wait and read address when the row address is the same as the previous address output (figure 9.22). When the row address is different from the previous address, the acknowledge signal is output across the precharge, row address, read command, wait and read address (figure 9.23).

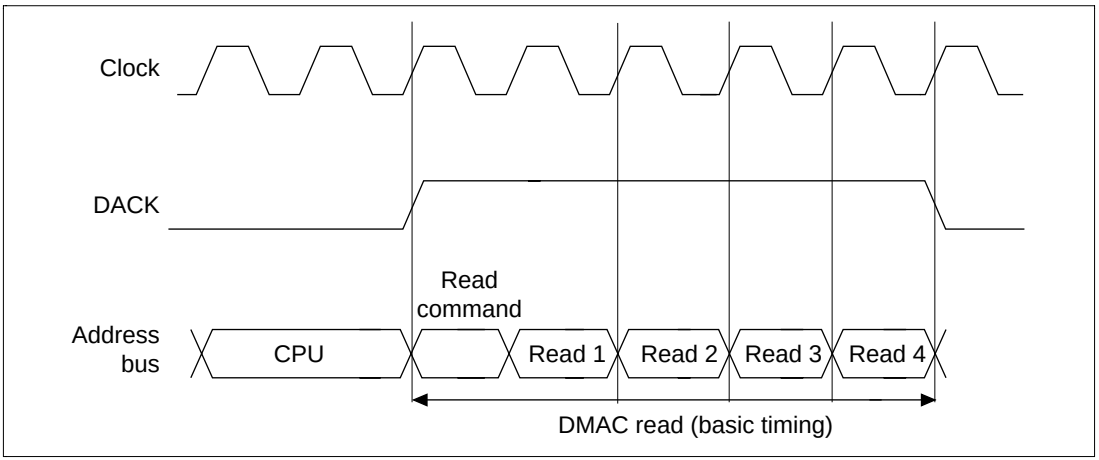



**Figure 9.22 DACK Output in Synchronous DRAM Burst Read (Bank Active, Same Row Address, AM = 0)**

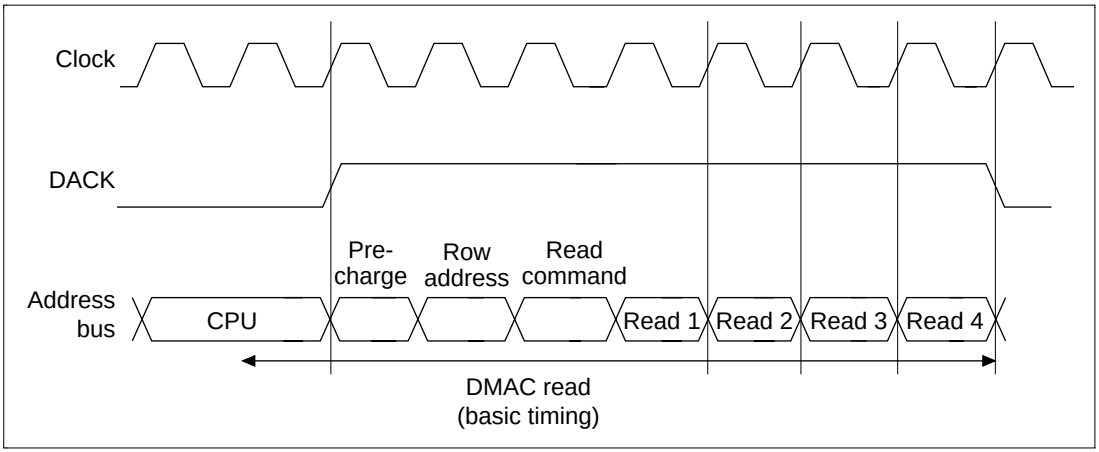



**Figure 9.23 DACK Output in Synchronous DRAM Burst Read (Bank Active, Different Row Address, AM = 0)**

When external memory is set as bank active synchronous DRAM, during a single read the acknowledge signal is output across the read command, wait and read address when the row address is the same as the previous address output (figure 9.24). When the row address is different from the previous address, the acknowledge signal is output across the precharge, row address, read command, wait and read address (figure 9.25). Since the synchronous DRAM read has only burst mode, during a single read an invalid address is output; the acknowledge signal is output on the same timing. At this time, the acknowledge signal is extended until the write address is output after the invalid read.

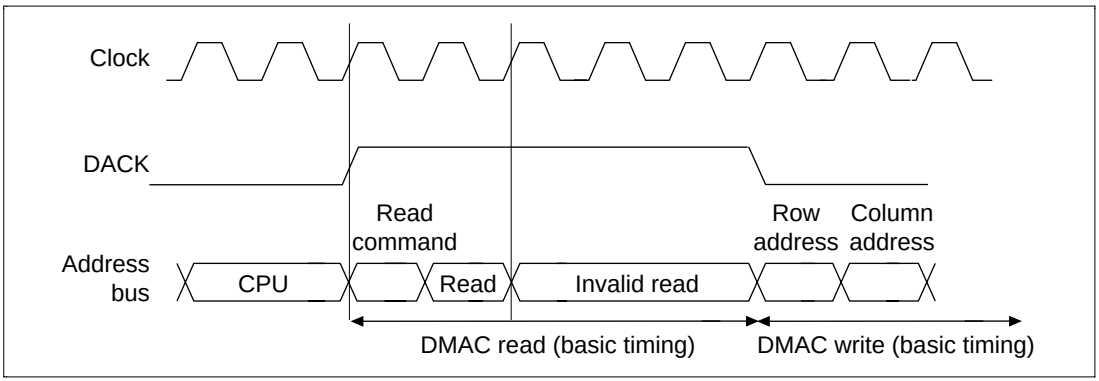



**Figure 9.24 DACK Output in Synchronous DRAM Single Read (Bank Active, Same Row Address, AM = 0)**

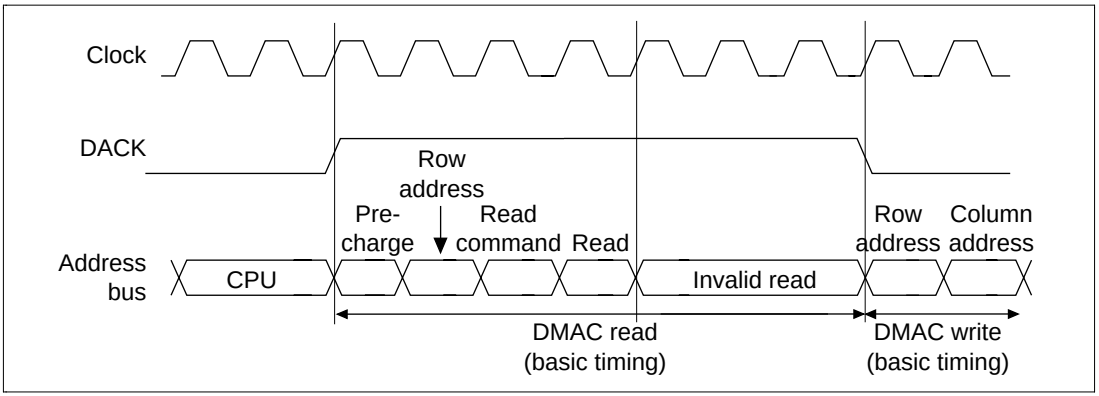



**Figure 9.25 DACK Output in Synchronous DRAM Single Read (Bank Active, Different Row Address, AM = 0)**

When external memory is set as bank active synchronous DRAM, during a write the acknowledge signal is output across the wait and column address when the row address is the same as the previous address output (figure 9.26). When the row address is different from the previous address, the acknowledge signal is output across the precharge, row address, wait and column address (figure 9.27).

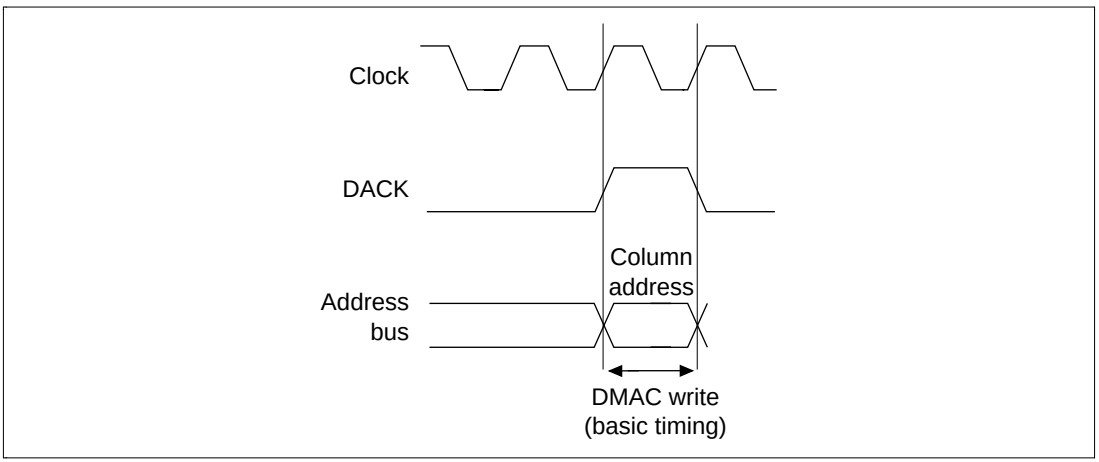



**Figure 9.26 DACK Output in Synchronous DRAM Write (Bank Active, Same Row Address, AM = 1)**

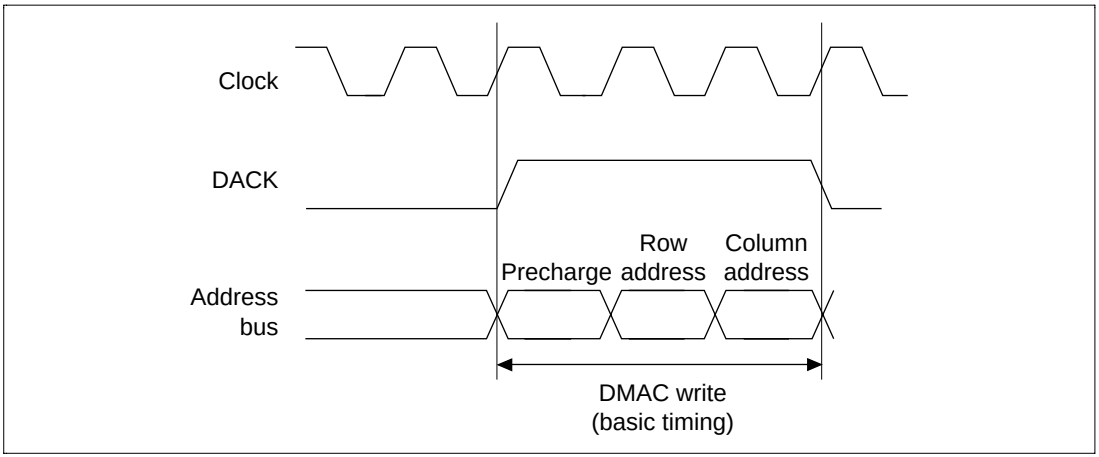



**Figure 9.27 DACK Output in Synchronous DRAM Write (Bank Active, Different Row Address, AM = 1)**

- SDRAM one-cycle write

  When a one-cycle write is performed to synchronous DRAM, the DACK signal is synchronized with the rising edge of the clock. A request by the request signal is accepted while the clock is high during DACK output.

| Transfer Width | Byte/Word/Longword Transfer | DREQ Detection Method | Level Detection |
|---|---|---|---|
| Transfer bus mode | Burst mode | DACK output timing | Write DACK |
| Transfer address mode | Single mode | Bus cycle | Basic bus cycle |

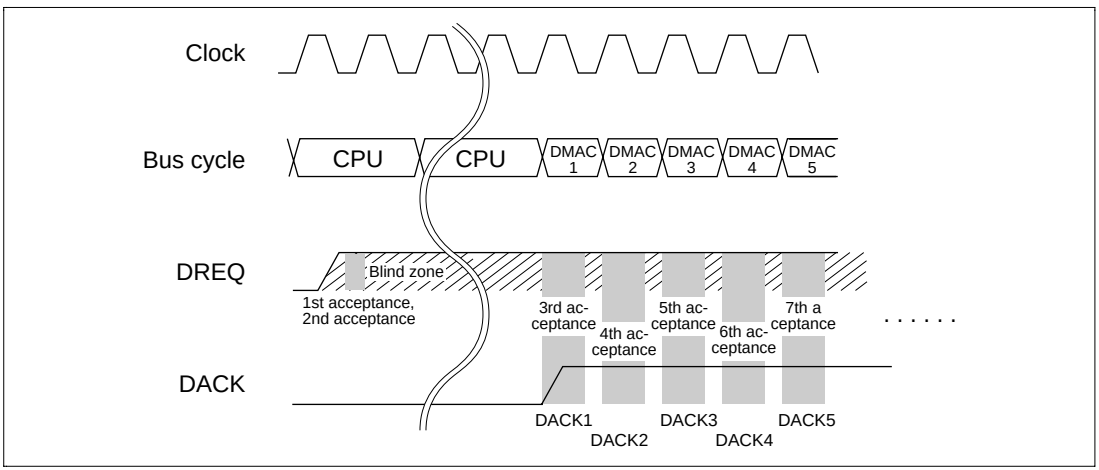



**Figure 9.28 SDRAM One-Cycle Write Timing**

**Acknowledge Signal Output when External Memory Is Set as DRAM:** When external memory is set as DRAM and a row address is output during a read or write, the acknowledge signal is output across the row address and column address (figures 9.29–9.31).

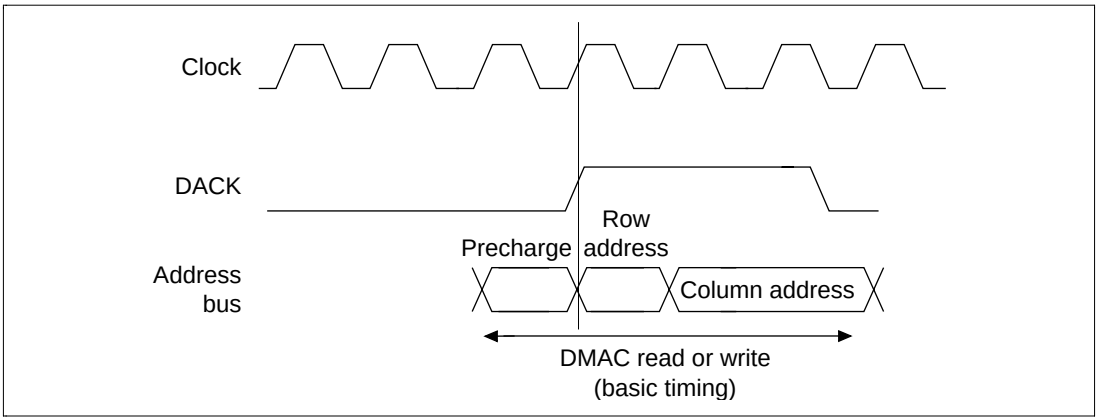



**Figure 9.29 DACK Output in Normal DRAM Accesses (AM = 0 or 1)**

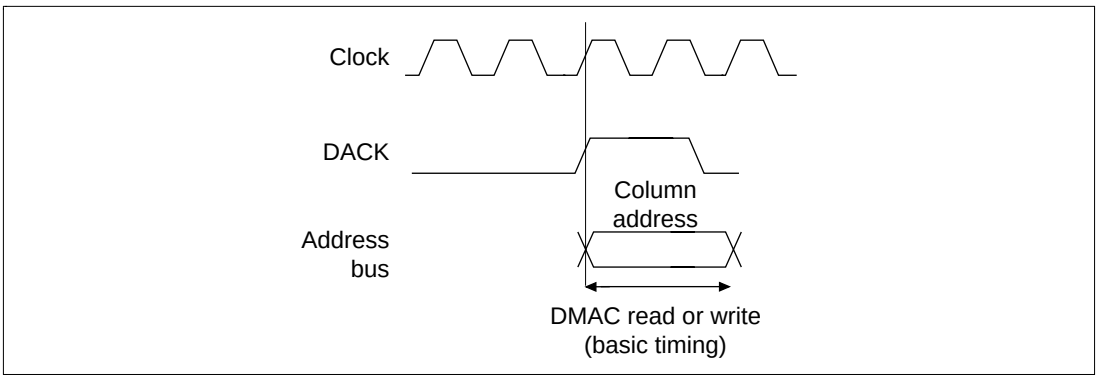



**Figure 9.30 DACK Output in DRAM Burst Accesses (Same Row Address, AM = 0 or 1)**

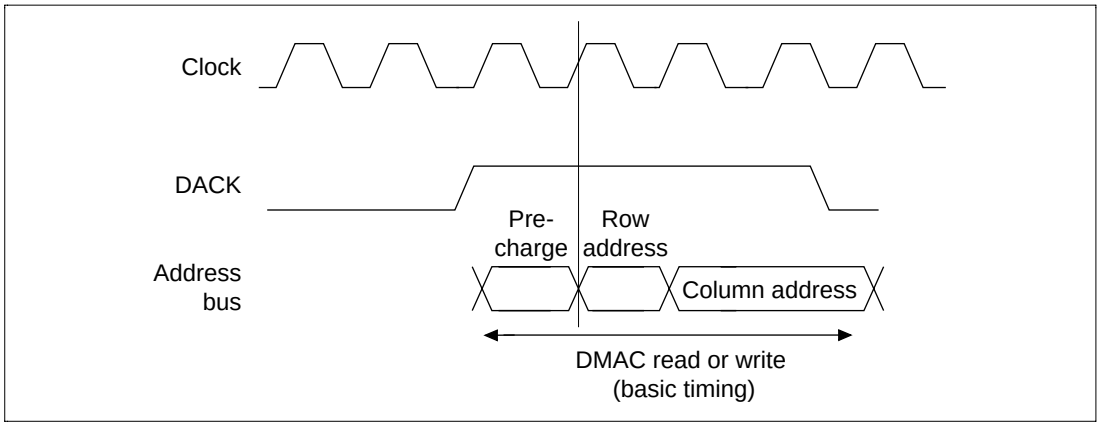



**Figure 9.31 DACK Output in DRAM Burst Accesses (Different Row Address, AM = 0 or 1)**

**Acknowledge Signal Output When External Memory Is Set as Burst ROM:** When external memory is set as burst ROM, the acknowledge signal is output synchronous to the DMAC address (no dual writes allowed) (figure 9.32).

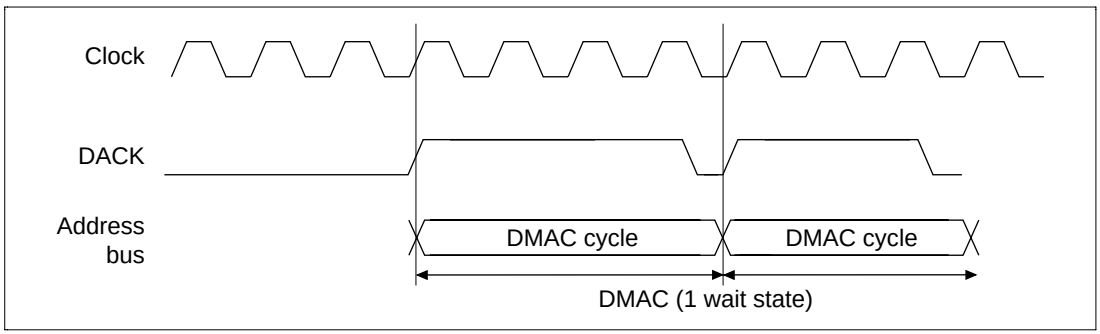



**Figure 9.32 DACK Output in Nibble Accesses of Burst ROM**

### 9.3.7 DREQ Pin Input Detection Timing

In external request mode, DREQ pin signals are usually detected at the falling edge of the clock pulse (CKIO). When a request is detected, a DMAC bus cycle is produced four cycles later at the earliest and a DMA transfer performed. After the request is detected, the timing of the next input detection varies with the bus mode, address mode, DREQ input detection, and the memory connected.

**DREQ Pin Input Detection Timing in Cycle-Steal Mode:** In cycle-steal mode, once a request is detected from the DREQ pin, the request signal is not detected until DACK signal output in the next external bus cycle. In cycle-steal mode, request detection is performed from DACK signal output until a request is detected.

In the case of a bus mode change or 16-byte transfer, the first DACK signal is the request acceptance start signal.

Once a request has been accepted, it cannot be canceled midway.

The timing from the detection of a request until the next time requests are detectable is shown below.

- Cycle-Steal Mode Edge Detection

| Transfer Width | Byte/Word/Longword | DREQ Detection Method | Edge Detection |
|---|---|---|---|
| Transfer bus mode | Cycle-steal mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |

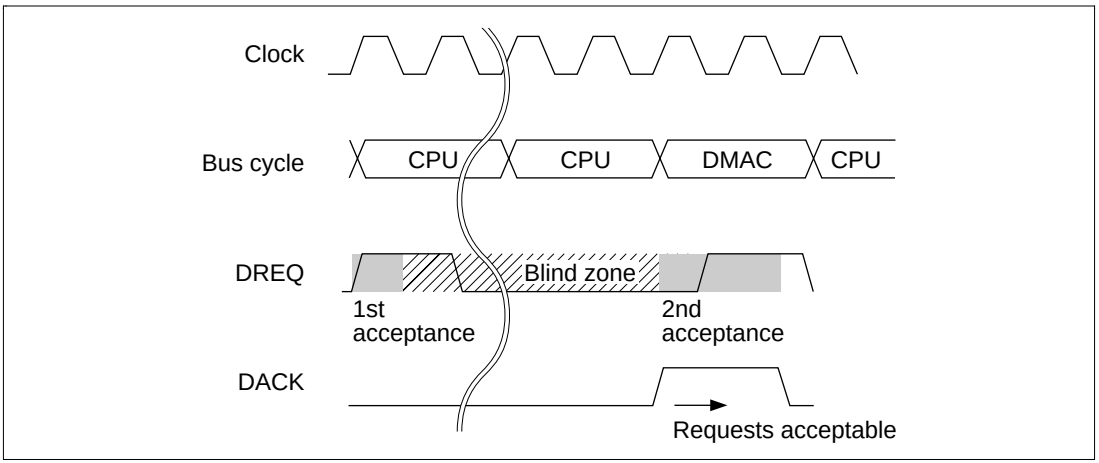



**Figure 9.33 DREQ Pin Input Detection Timing in Cycle-Steal Mode with Edge Detection**

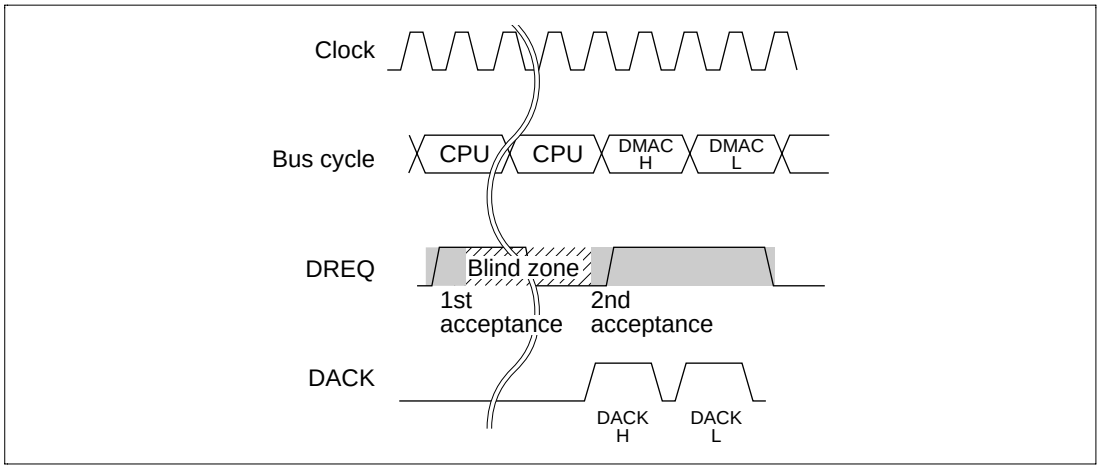



**Figure 9.34 When a16-Bit External Device is Connected (Edge Detection)**

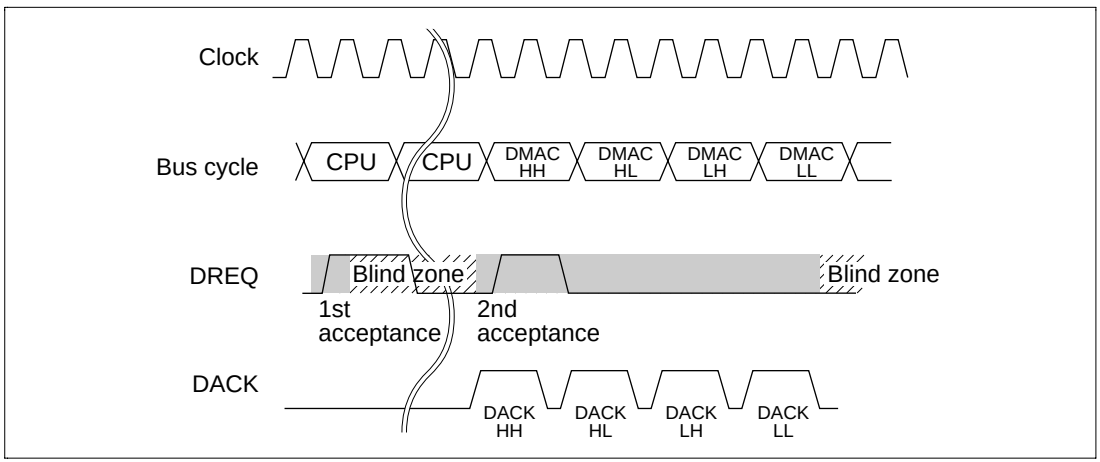
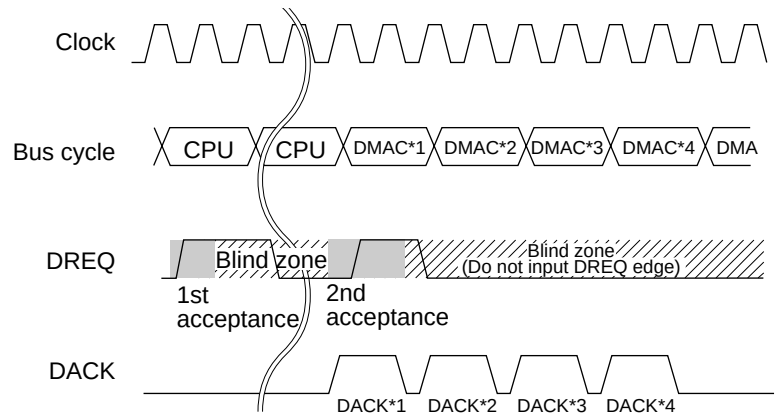



**Figure 9.35 When an 8-Bit External Device is Connected (Edge Detection)**

- Cycle-Steal Mode Edge Detection—16-Bit Transfer

  With 16-byte transfer, the first request signal is the first transfer request, and the second transfer request is accepted when the next request signal is accepted. The third and fourth requests are accepted in the same way. If DREQ is input continuously, the bus is not passed to the CPU.

| Transfer Width | 16-Byte Transfer | DREQ Detection Method | Edge Detection |
|---|---|---|---|
| Transfer bus mode | Cycle-steal mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |




Note: * n is the nth 16-byte transfer.

**Figure 9.36 DREQ Pin Input Detection Timing in Cycle-Steal Mode with Edge Detection (16-Byte Transfer Setting)**

- Cycle-Steal Mode Level Detection
  In level detection mode, too, a request cannot be canceled once accepted.

| Transfer Width | Byte/Word/Longword | DREQ Detection Method | Level Detection |
|---|---|---|---|
| Transfer bus mode | Cycle-steal mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |

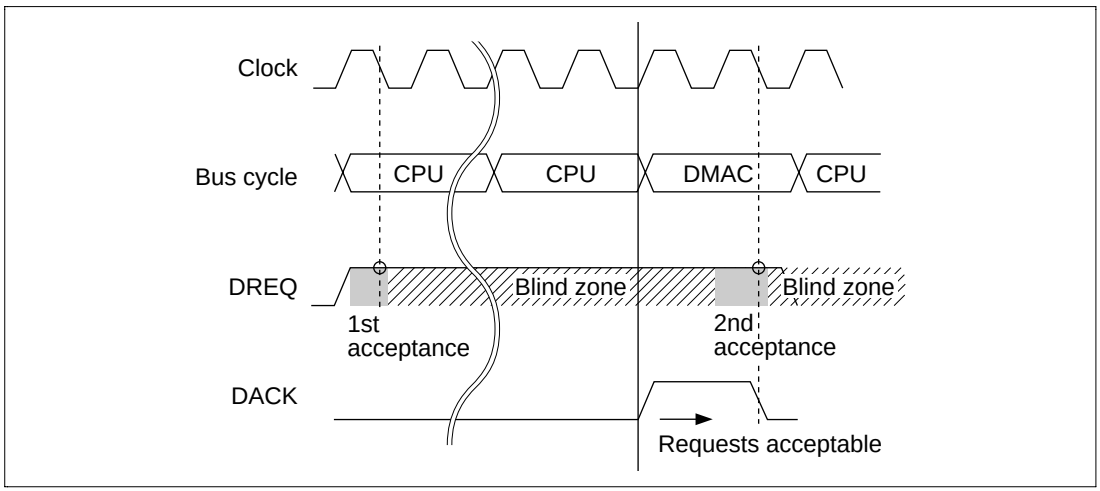



**Figure 9.37 DREQ Pin Input Detection Timing in Cycle-Steal Mode with Level Detection (Byte/Word/Longword Setting)**

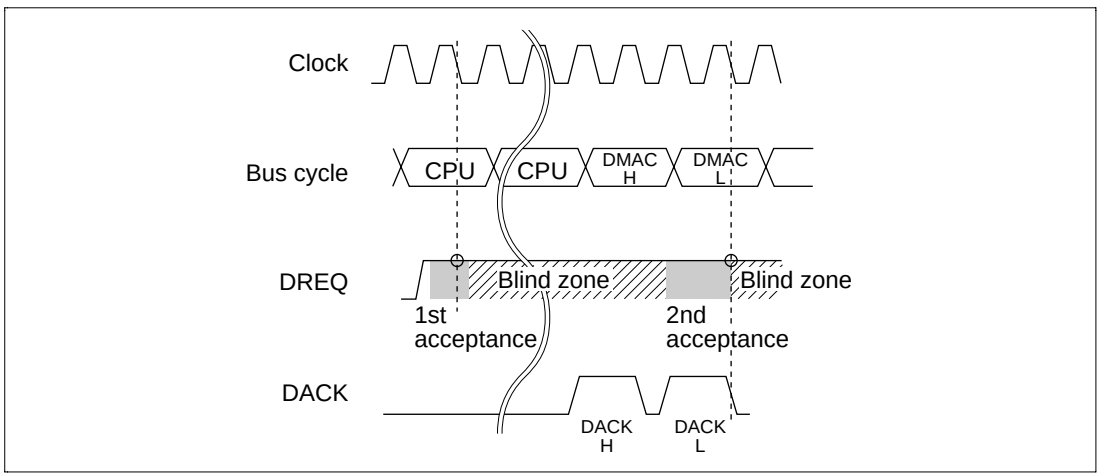



**Figure 9.38 When a 16-Bit External Device is Connected (Level Detection)**

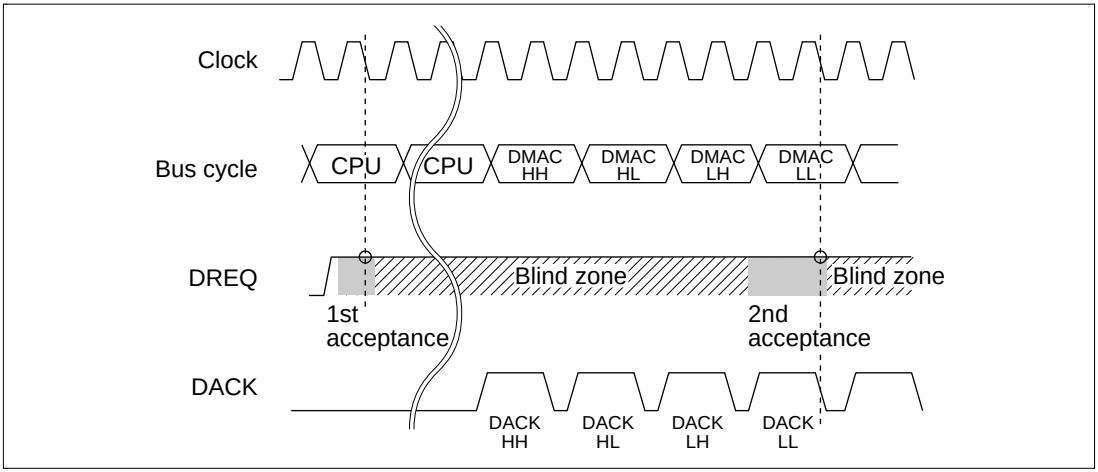



**Figure 9.39 When an 8-Bit External Device is Connected (Level Detection)**

- Cycle-Steal Mode Level Detection—16-Byte Transfer

  With 16-byte transfer, the first request signal is the first transfer request, and the second transfer request is accepted when the next request signal is accepted. The third and fourth requests are accepted in the same way. If DREQ is input continuously, the bus is not passed to the CPU.

| Transfer Width | 16-Byte Transfer | DREQ Detection Method | Level Detection |
|---|---|---|---|
| Transfer bus mode | Cycle-steal mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |

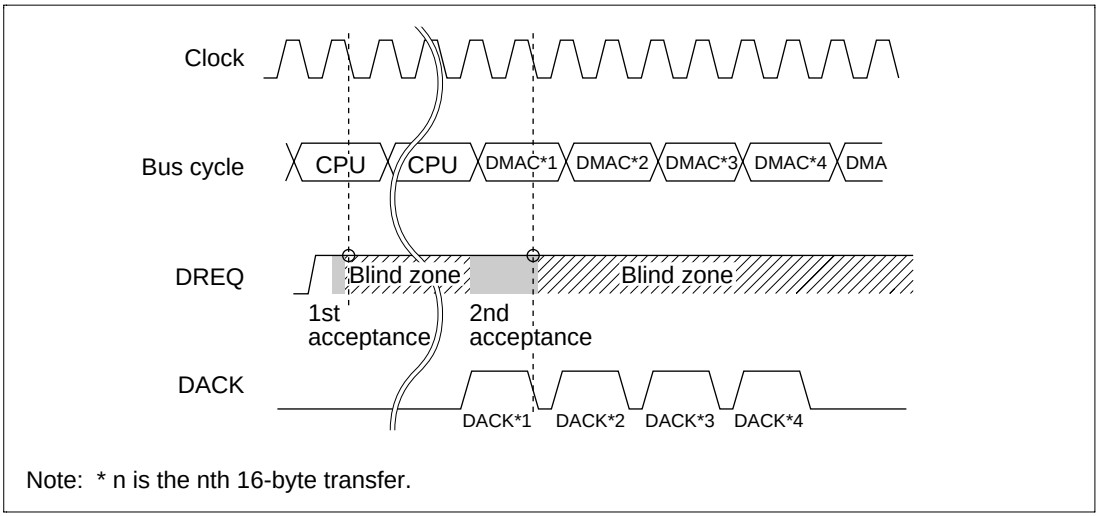



**Figure 9.40 DREQ Pin Input Detection Timing in Cycle-Steal Mode with Level Detection (16-Byte Transfer Setting)**

**DREQ Pin Input Detection Timing in Burst Mode:** In burst mode, the request detection timing is different for edge detection and level detection of DREQ input.

With edge detection of DREQ input, once a request is detected, DMA transfer continues until the transfer end condition is satisfied, regardless of the state of the DREQ pin. Request detection is not performed during this time. When the transfer start conditions are fulfilled after the end of transfer, request detection is performed again every cycle.

With level detection of DREQ input, if a request for the same channel is detected in the following request detection cycle, that channel will continue to operate, but if a request is not input, a bus cycle for another channel or another bus master will be executed.

- Burst Mode Level Detection—Byte/Word/Longword Size

| Transfer Width | Byte/Word/Longword | DREQ Detection Method | Level Detection |
|---|---|---|---|
| Transfer bus mode | Burst mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |

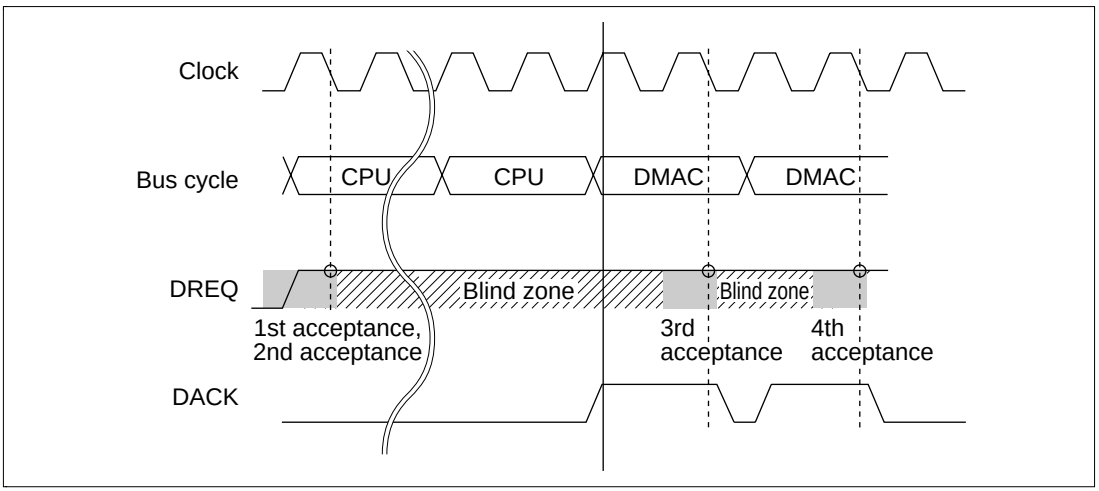


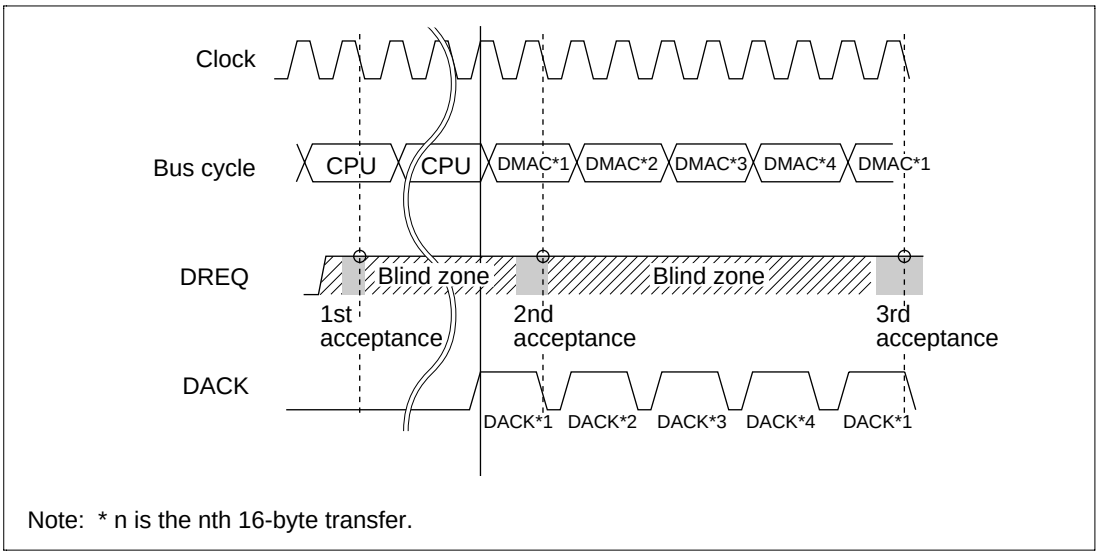

**Figure 9.41 DREQ Pin Input Detection Timing in Burst Mode with Level Detection (Byte/Word/Longword Setting)**

With level detection in burst mode, two transfers are performed on the first acceptance of the request signal, and the third transfer request signal is accepted by the first DACK signal. Subsequently, the fourth and fifth transfer requests are accepted in turn. However, if TCR = 1, the operation ends after one transfer on acceptance of the first request.

- Burst Mode Level Detection—16-Byte Transfer

  With 16-byte transfer, the first request signal is the first transfer request, and the second transfer request is accepted when the next request signal is accepted. The third and fourth requests are accepted in the same way.

| Transfer Width | 16-Byte Transfer | DREQ Detection Method | Level Detection |
|---|---|---|---|
| Transfer bus mode | Burst mode | DACK output timing | Read DACK/write DACK |
| Transfer address mode | Dual/single mode | Bus cycle | Basic bus cycle |




Note: * n is the nth 16-byte transfer.

**Figure 9.42 DREQ Pin Input Detection Timing in Burst Mode with Level Detection (16-Byte Transfer Setting)**

### 9.3.8 DMA Transfer End

The DMA transfer ending conditions vary when channels end individually and when both channels end together.

**Conditions for Channels Ending Individually:** When either of the following conditions is met, the transfer will end in the relevant channel only:

The DMA transfer count register (TCR) value becomes 0.
The DMA enable bit (DE) of the DMA channel control register (CHCR) is cleared to 0.

- Transfer end when TCR = 0

  When the TCR value becomes 0, the DMA transfer for that channel ends and the transfer-end flag bit (TE) is set in CHCR. If the IE (interrupt enable) bit has already been set, a DMAC interrupt (DEI) request is sent to the CPU. For 16-byte transfer, set the number of transfers × 4.

- Transfer end when DE = 0 in CHCR

  When the DMA enable bit (DE) in CHCR is cleared, DMA transfers in the affected channel are halted. The TE bit is not set when this happens.

**Conditions for Both Channels Ending Simultaneously:** Transfers on both channels end when either of the following conditions is met:

The NMIF (NMI flag) bit or AE (address error flag) bit in DMAOR is set to 1.
The DMA master enable (DME) bit is cleared to 0 in DMAOR.

- Transfer end when NMIF = 1 or AE = 1 in DMAOR

  When an NMI interrupt or DMAC address error occurs and the NMIF or AE bit is set to 1 in DMAOR, all channels stop their transfers. The DMA source address register (SAR), destination address register (DAR), and transfer count register (TCR) are all updated by the transfer immediately preceding the halt. When this transfer is the final transfer, TE = 1 and the transfer ends. To resume transfer after NMI interrupt exception handling or address error exception handling, clear the appropriate flag bit. When the DE bit is then set to 1, the transfer on that channel will restart. To avoid this, keep its DE bit at 0. In dual address mode, DMA transfer will be halted after the completion of the following write cycle even when the address error occurs in the initial read cycle. SAR, DAR and TCR are updated by the final transfer.

- Transfer end when DME = 0 in DMAOR

  Clearing the DME bit in DMAOR forcibly aborts the transfers on both channels at the end of the current bus cycle. When the transfer is the final transfer, TE = 1 and the transfer ends.

## 9.4 Usage Notes

### 9.4.1 DMAC-Related Notes

1. DMA request/response selection control registers 0 and 1 (DRCR0 and DRCR1) should be accessed in bytes. All other registers should be accessed in longword units.
2. Before rewriting a register in the DMAC, first clear the DE bit to 0 in the CHCR register for the specified channel, or clear the DME bit in DMAOR to 0.
3. When the DMAC is not operating, the NMIF bit in DMAOR is set even when an NMI interrupt is input.
4. The DMAC cannot access the cache memory.
5. Set to standby mode after the DME bit in DMAOR is set to 0.
6. When performing a frequency modification, the DME bit in DMAOR should first be cleared to 0.
7 Do not perform on-chip peripheral module transfers by means of the DMAC. Specifically, do not access DMAC addresses H'FFFF F000 to H'FFFF FCFF or H'FFFF FE00 to H'FFFF FEFF.
8. Do not access the cache (address array, data array, associative purge area).
9. Note that when level detection of the request signal is used in single address mode, the request signal may be detected before DACK is output.
10. When Eϕ exceeds 30 MHz, do not use transfer involving DACK output on ordinary space for word or longword access with an 8-bit bus width, or longword access with a 16-bit bus width.
11. In 16-byte transfer with cycle-steal mode and external request DREQ edge detection set, do not input a new edge in the blind zone.
12. When DMA transfer is performed in response to a DMA transfer request signal from a peripheral module, if clearing of the DMA transfer request signal from the peripheral module by the DMA transfer is not completed before the next transfer request signal from that module, subsequent DMA transfers may not be possible.
13. The following restrictions apply when using dual address mode for 16-byte transfer in cycle-steal mode:
    a. When auto-request is set, the operation is the same as for burst transfer.
    b. When external request level detection is set, the operation is the same as for burst transfer.
    c. When external request DREQ edge detection is set, if DREQ is input continuously the DMAC continues to operate without insertion of a CPU cycle. (However, a CPU cycle will begin if there is no request from DREQ.)
14. Notes (a) and (b) below apply in the case of DMAC burst transfer with external requests and level detection. To prevent these problems, CHCR should be cleared before enabling DMA. The clearing sequence is as follows:

    Step 1: Read CHCR (read TE = 1)
    Step 2: Clear CHCR to 0 (write TE = 0, DE = 0, disabling DMA)

Step 3: Set CHCR again (leaving DE = 0)
Step 4: Set DE = 1 in CHCR (enabling DMA)

a. The DREQ level is sampled even when DMA is disabled (DE = 0 or DME = 0). Therefore, depending on the DMA settings and DREQ input, DMA transfer may be initiated when DMA is enabled even though there is no DREQ input (figures 9.43 and 9.44).

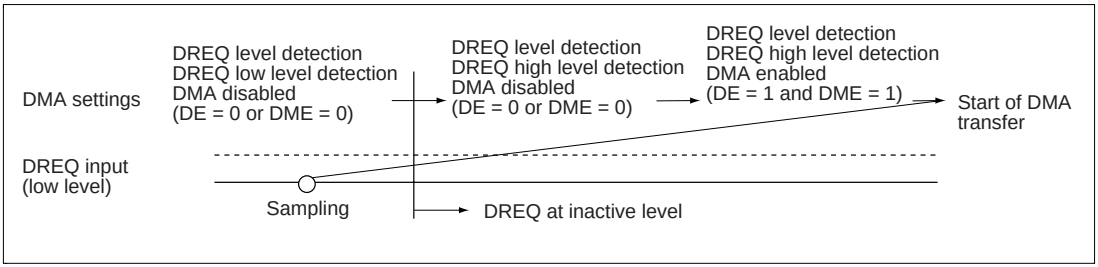



**Figure 9.43 Initiation of DMA Transfer (1)**

b. If DREQ is active when the last transfer is performed in burst transfer, DMA transfer will start when DMA is next enabled, even if DREQ is inactive (figure 9.44).

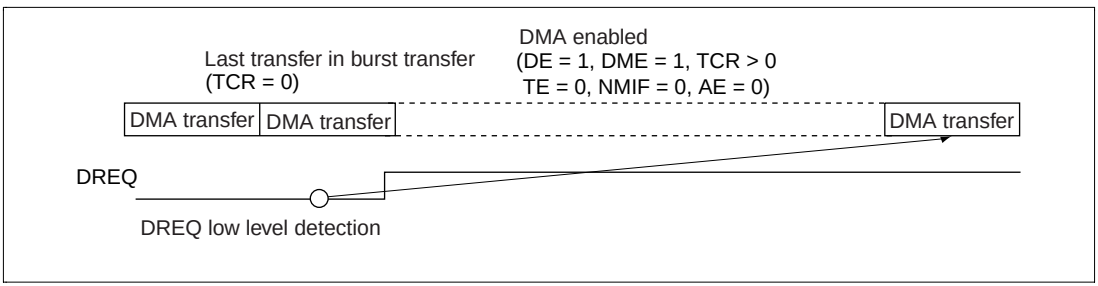



**Figure 9.44 Initiation of DMA Transfer (2)**

15. In DMAC cycle-steal mode with 16-byte length, dual address transfer, external request, and level detection, overrun 2 occurs if DREQ is negated in a cycle in which DACK is not output (figure 9.45).

    When halting the DMAC by negation of the external request signal with the above settings, negate the external request in a cycle in which DACK is asserted.

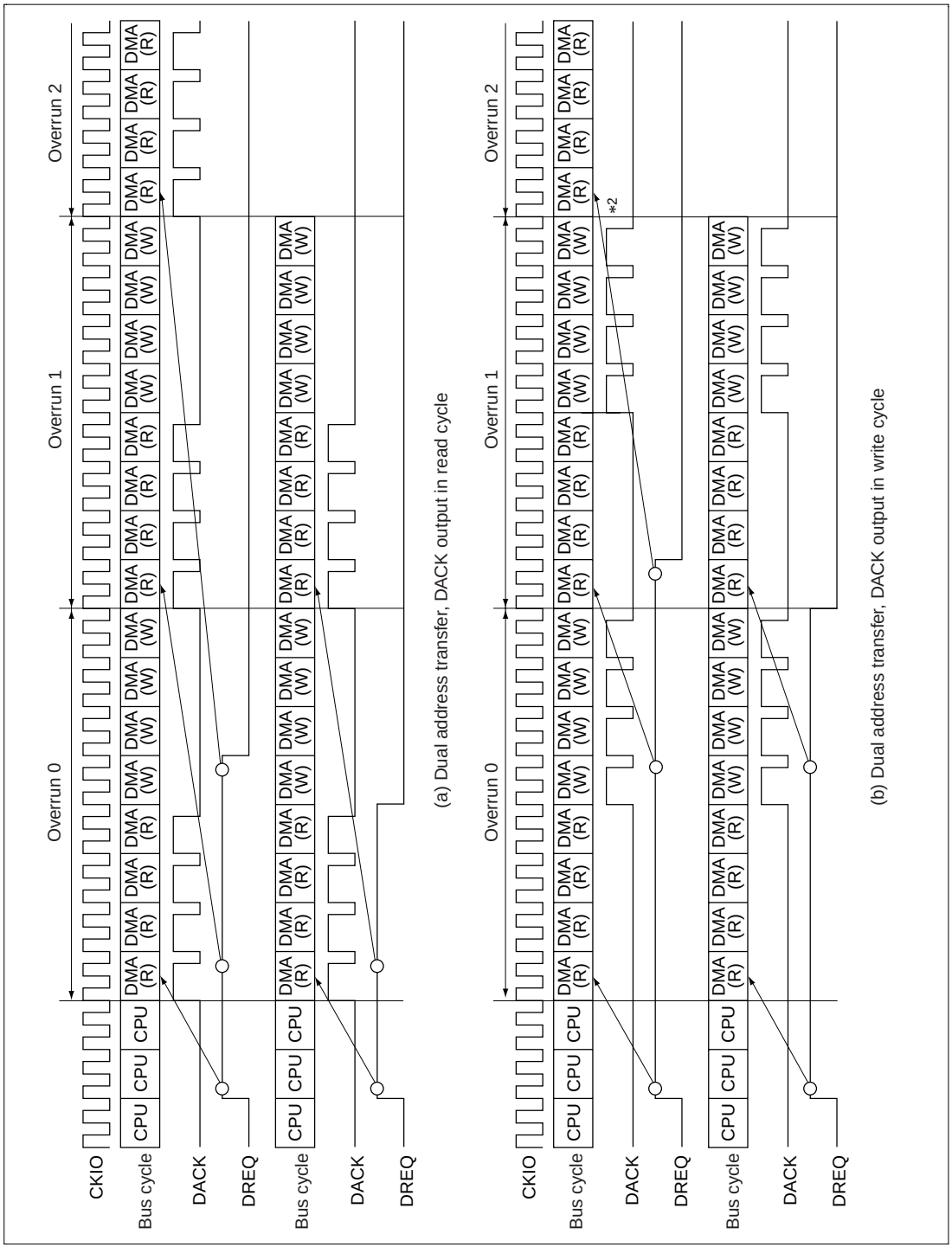



**Figure 9.45 When DMAC is Halted by Negation of External Request Signal**

### 9.4.2 Notes on Memory Access when Using the DMAC

**Bug occurring in synchronous DRAM single write mode when clock ratio Iø:Eø = 1:1**

Conditions: Clock ratio Iø:Eø = 1:1 (Iø: CPU clock, Eø: external memory clock (CKIO))
SDRAM single write mode
DMAC dual address transfer
Read from SDRAM immediately after write to SDRAM (see table 9.9)

**Table 9.9 Access Sequence**

| Write to SDRAM | Read from SDRAM |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur when a read is performed by the CPU immediately after a write, because the accesses are not consecutive internally.

Problem: When an SDRAM read access is performed, the SH7612 drives the data bus.

Remedy: When accessing SDRAM using the DMAC in dual address mode with clock ratio Iø:Eø = 1:1, the SDRAM should be used in burst write mode.

**Note on writing to normal space immediately after a synchronous DRAM write when clock ratio Iø:Eø = 1:1**

Conditions: Clock ratio Iø:Eø = 1:1 (Iø: CPU clock, Eø: external memory clock (CKIO))
Write to normal space immediately after write to SDRAM (see table 9.10)

**Table 9.10 Access Sequence**

| Write to SDRAM | Write to Normal Space |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur when a write is performed by the CPU immediately after a write, because the accesses are not consecutive internally.

Problem: When a write to normal space is performed, the data bus is not driven in the first cycle (figures 9.46 to 9.48).

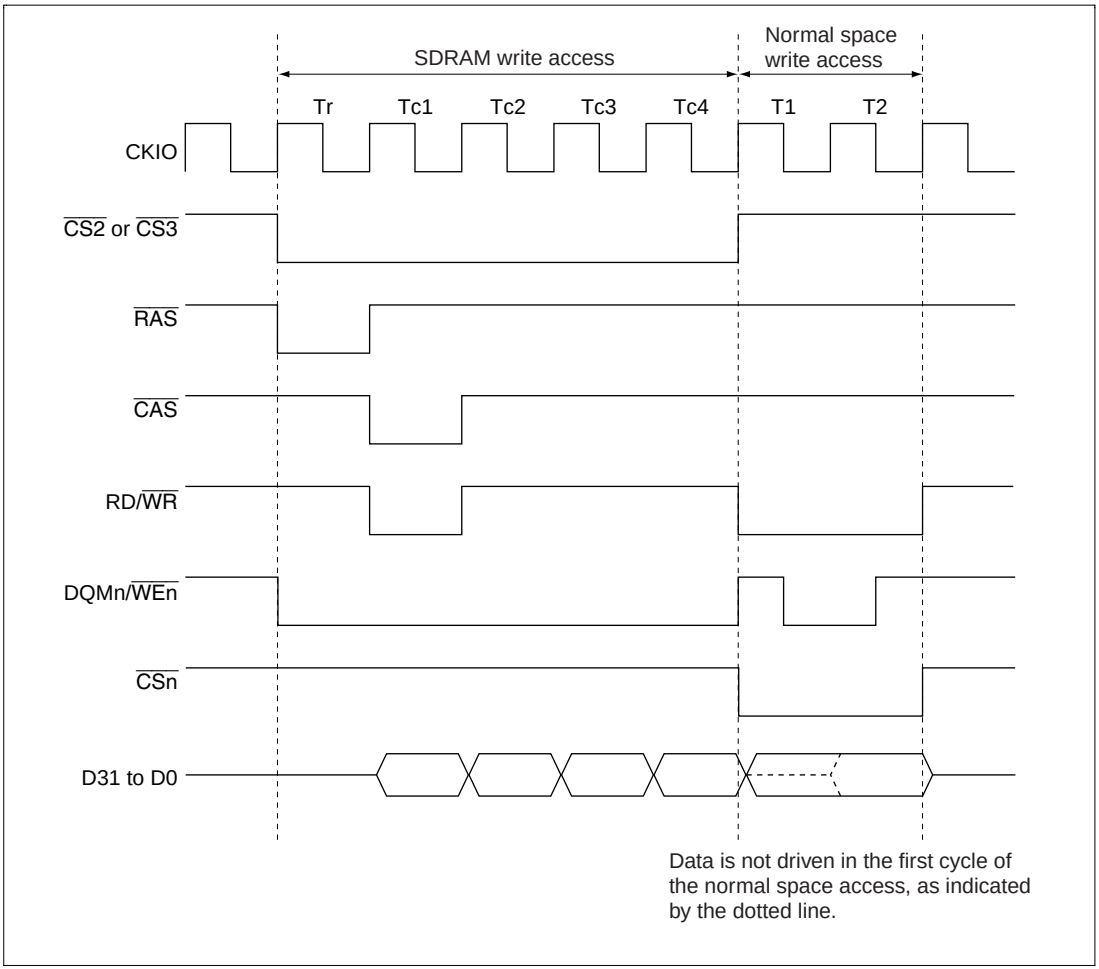



**Figure 9.46 Normal Space Access (No Wait)**

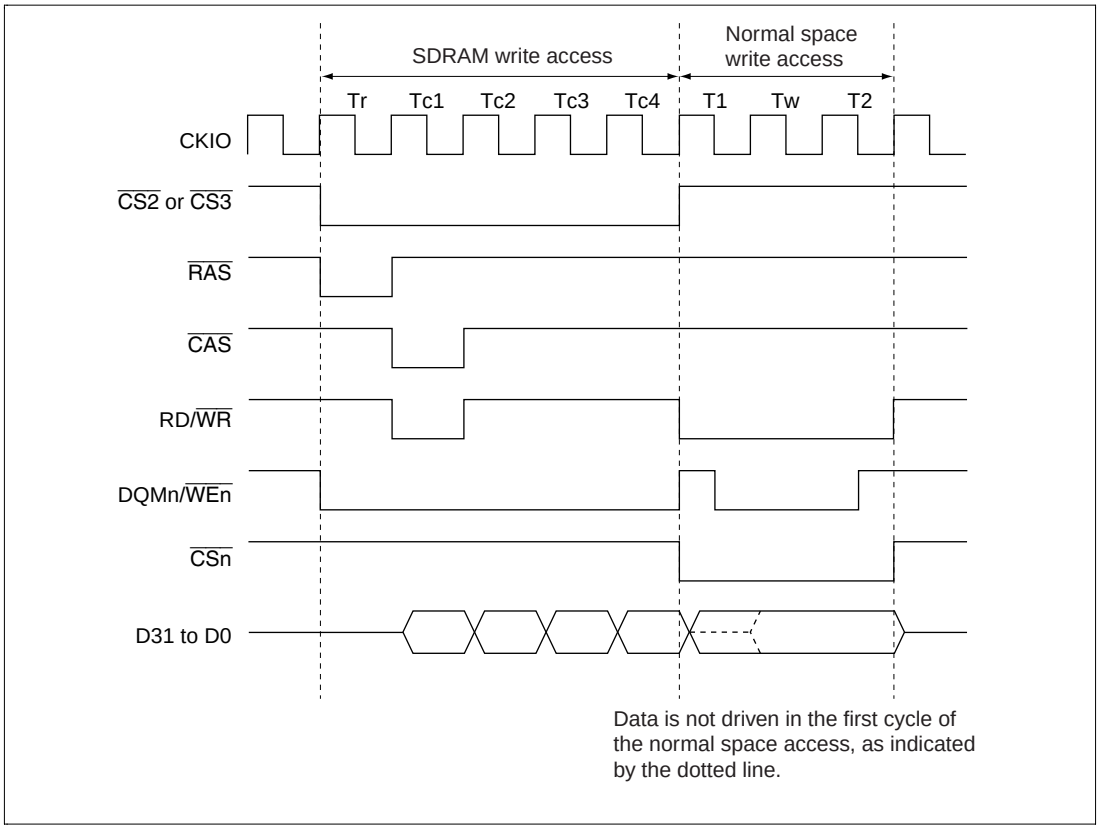



**Figure 9.47 Normal Space Access (One Wait)**

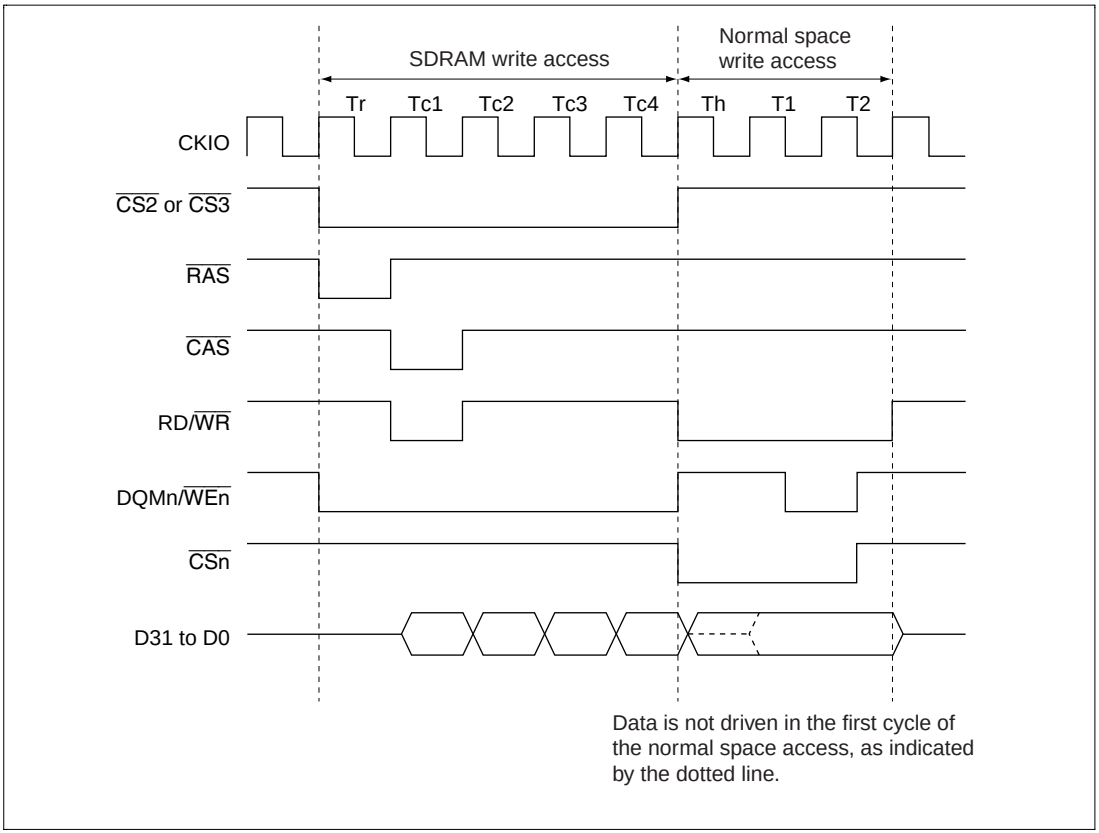



**Figure 9.48 Normal Space Access (One Setup Wait Cycle)**

Remedy: When the above conditions apply, in order to secure the normal space write data setup time, either increase the normal space wait by one cycle, or increase the number of cycles from address and $\overline{\text{CSn}}$ output to $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ assertion by one.

**Bug occurring in synchronous DRAM bank-active mode**

Conditions: SDRAM bank-active mode
SDRAM access by DMAC
Consecutive accesses to SDRAM (see table 9.11)

**Table 9.11 Access Sequence**

| Accsss to SDRAM | Access to SDRAM |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: The bug does not occur with consecutive accesses by the CPU, because the accesses are not consecutive internally.

Problem: The second SDRAM access is not performed correctly.

Remedy: Do not use bank-active mode when accessing SDRAM using the DMAC.

**EDO access bug occurring in DMAC single address mode**

Conditions:

- DMAC operating mode
  - — Single address transfer mode
  - — Transfer from memory to device with DACK (AM: CHRC0, CHRC1 bit 8 = 1)
- BSC setting
  - — EDO RAS down mode
  - — W31—W30 = 00

Problem: Although RAS down mode is set, $\overline{\text{RAS}}$ is negated for one cycle. The next access to EDO may not be performed correctly.

Remedy: When performing transfer from memory to a device with DACK (AM: CHRC0, CHRC1 bit 8 = 1) for EDO in DMAC single address transfer mode, either do not use RAS down mode, or else set bits W31—W30 to a value other than 00.

**Note on normal space access immediately after a synchronous DRAM write:** The DQMn/$\overline{\text{WEn}}$ signal negation for an SDRAM write and the $\overline{\text{CSn}}$ assertion for an immediately following normal space access occur at the same rising edge of CKIO (figures 9.49 and 9.50). The access sequences when an SDRAM write and normal space access are consecutive are shown in table 9.12.

**Table 9.12 Access Sequence**

| Write to SDRAM | Normal Space Access |
|---|---|
| CPU | DMA |
| DMA | CPU |
| DMA | DMA |

Note: When a write is performed by the CPU immediately after a write, the accesses are not consecutive internally.

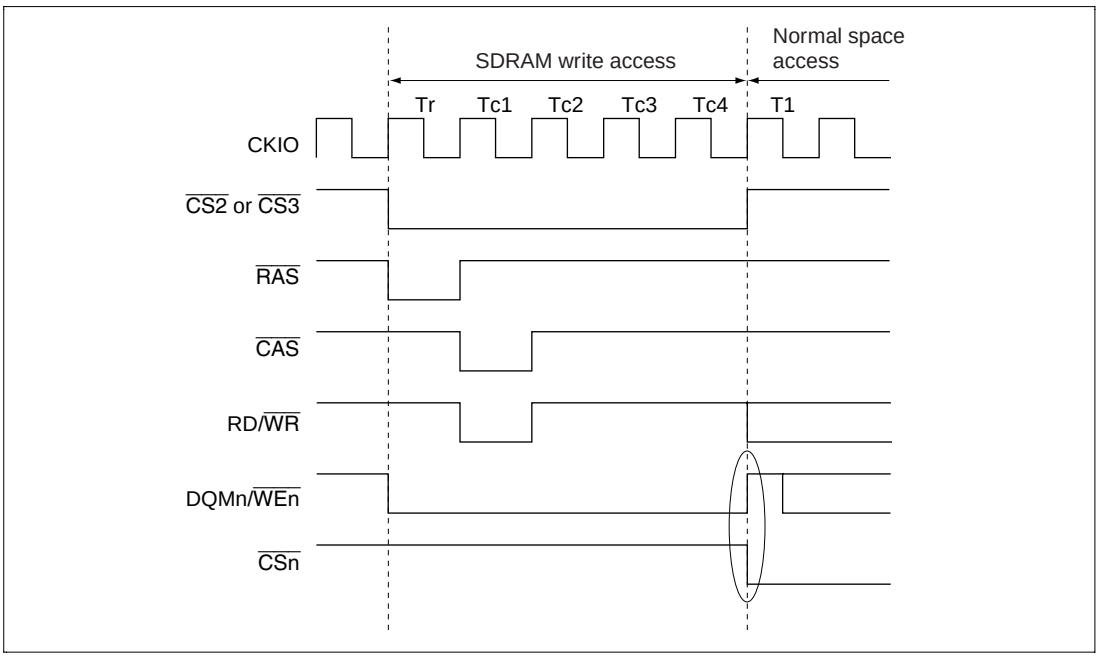



**Figure 9.49 Normal Space Access (Burst Write Mode)**

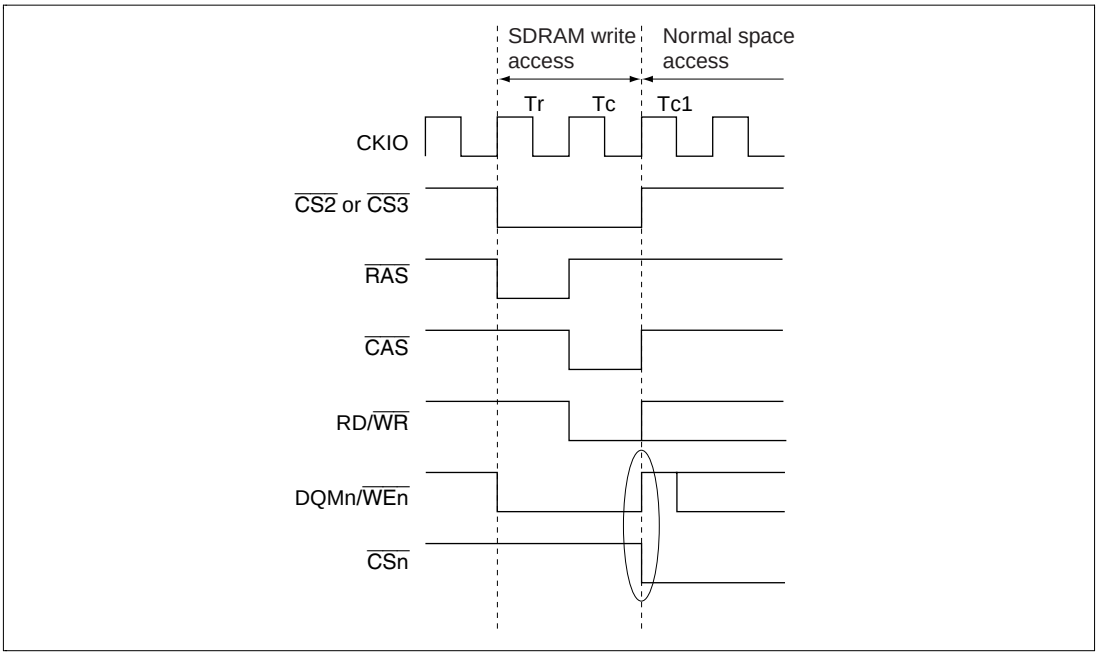



**Figure 9.50 Normal Space Access (Single Write Mode)**

**Note on DACK output when synchronous DRAM is accessed in DMAC single address mode:** In the case of consecutive accesses to synchronous DRAM by the DMAC, DACK is not negated in the Trwl and Tap cycles.

# Section 10 Division Unit (DIVU)

## 10.1 Overview

The division unit (DIVU) divides 64 bits by 32 bits and 32 bits by 32 bits. The results are expressed as a 32-bit quotient and a 32-bit remainder. When the operation produces an overflow, an interrupt can be generated as specified.

### 10.1.1 Features

The DIVU has the following features:

- Signed division of 64 bits by 32 bits and 32 bits by 32 bits operations
- 32-bit quotient, 32-bit remainder
- Operation execution in 39 cycles
- Overflow interrupt enable/disable control
- Instructions that do not access the division unit can be processed in parallel even during execution of division processing

### 10.1.2 Block Diagram

Figure 10.1 shows a block diagram of the DIVU.

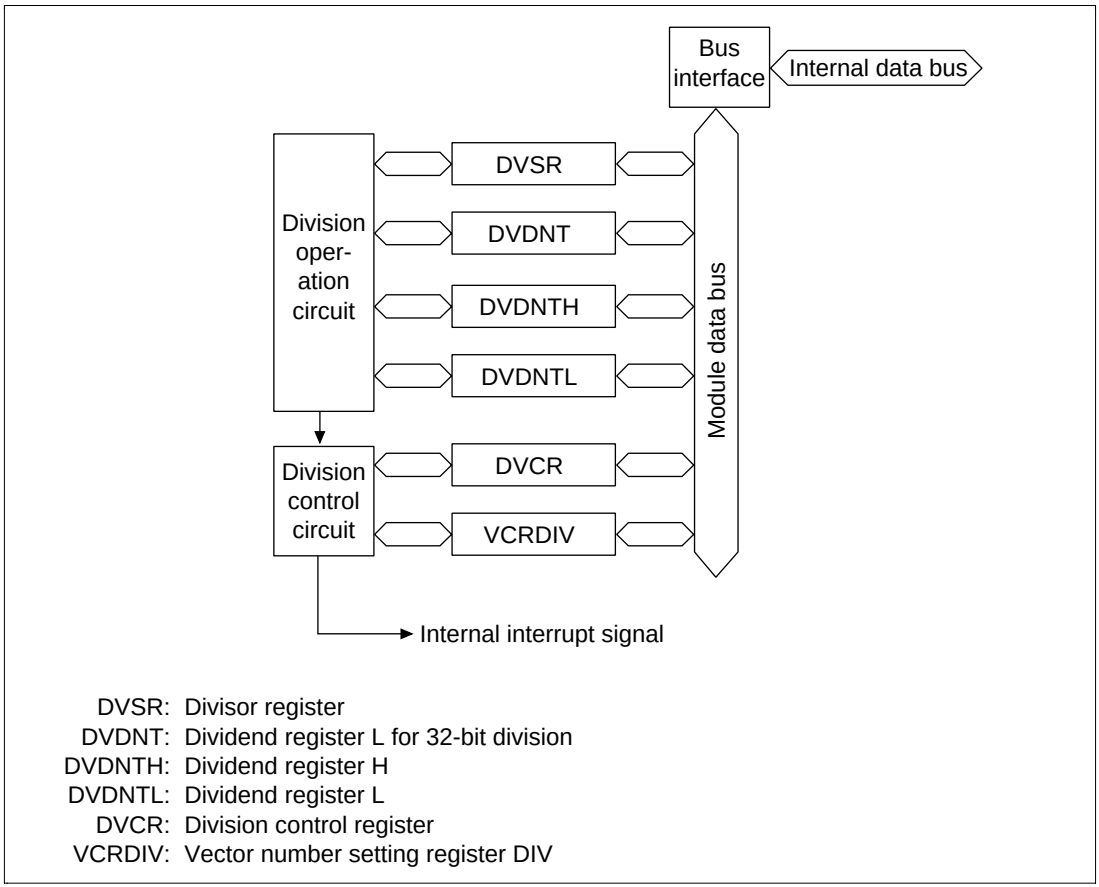



**Figure 10.1 DIVU Block Diagram**

### 10.1.3 Register Configuration

Table 10.1 shows the register configuration of the DIVU.

**Table 10.1 Register Configuration**

| Register | Abbr. | R/W | Initial Value | Address | Access Size[*1] |
|---|---|---|---|---|---|
| Divisor register | DVSR | R/W | Undefined | H'FFFFFF00 | 32 |
| Dividend register L for 32-bit division | DVDNT | R/W | Undefined | H'FFFFFF04 | 32 |
| Division control register | DVCR | R/W | H'00000000 | H'FFFFFF08 | 16, 32 |
| Vector number setting register DIV | VCRDIV | R/W | Undefined[*2] | H'FFFFFF0C | 16, 32 |
| Dividend register H | DVDNTH | R/W | Undefined | H'FFFFFF10 | 32 |
| Dividend register L | DVDNTL | R/W | Undefined | H'FFFFFF14 | 32 |

Notes: 1. Accesses to the division unit are read and written in 32-bit units. DVCR and VCRDIV permit 16 and 32-bit accesses. When registers other than DVCR and VCRDIV are accessed with word accesses, undefined values are read or written.
2. The initial value of VCRDIV is H'0000**** (asterisks represent undefined values).

## 10.2 Register Descriptions

### 10.2.1 Divisor Register (DVSR)

| Bit: | 31 | 30 | 29 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

The divisor register (DVSR) is a 32-bit read/write register in which the divisor for the operation is written. Its value is undefined after a power-on reset or manual reset, in standby mode, and when the module standby function is used.

### 10.2.2 Dividend Register L for 32-Bit Division (DVDNT)

| Bit: | 31 | 30 | 29 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

The dividend register L for 32-bit division (DVDNT) is a 32-bit read/write register in which the 32-bit dividend used for 32-bit ÷ 32-bit division operations is written. When 32-bit ÷ 32-bit division is run, the value set as the dividend is lost and the quotient written at the end of division. When this register is written to, the same value is written in the DVDNTL register. The MSB written is sign-extended in the DVDNTH register. Writing to this register starts the 32-bit ÷ 32-bit division operation. Its value is undefined after a power-on reset or manual reset, in standby mode, and when the module standby function is used.

### 10.2.3 Division Control Register (DVCR)

| Bit: | 31 | 30 | 29 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | … | — | — | OVFIE | OVF |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R/W | R/W |

The division control register (DVCR) is a 32-bit read/write register, but is also 16-bit accessible. It controls enabling/disabling of the overflow or underflow interrupt. This register is initialized to H'00000000 by a power-on reset or manual reset. Its value is undefined in standby mode and when the module standby function is used.

Bits 31 to 2: Reserved. These bits always read 0. The write value should always be 0.

Bit 1: OVF Interrupt Enable (OVFIE): Selects enabling or disabling of the OVF interrupt request (OVFI) upon overflow.

| Bit 1: OVFIE | Description | |
|---|---|---|
| 0 | Interrupt request (OVFI) caused by OVF disabled | (Initial value) |
| 1 | Interrupt request (OVFI) caused by OVF enabled | |

Note: Always set the OVFIE bit before starting the operation whenever executing interrupt handling for overflows.

Bit 0: Overflow Flag (OVF). Flag indicating an overflow has occurred.

| Bit 0: OVF | Description | |
|---|---|---|
| 0 | No overflow has occurred | (Initial value) |
| 1 | Overflow has occurred | |

### 10.2.4 Vector Number Setting Register DIV (VCRDIV)

| Bit: | 31 | 30 | 29 | … | 10 | 9 | 8 | 7 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | … | — | — | — | — |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R | R |

| Bit: | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | |
| Initial value: | — | — | — | — | — | — | — |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Vector number setting register DIV (VCRDIV) is a 32-bit read/write register, but is also 16-bit accessible. The destination vector number is set in VCRDIV when an interrupt occurs in the division unit due to an overflow or underflow. Values can be set in the 16 bits from bit 15 to bit 0, but only the last 7 bits (bits 6–0) are valid. Always set 0 for the 9 bits from bit 15 to bit 7. Its value is undefined after a power-on reset or manual reset, in standby mode, and when the module standby function is used.

Bits 31 to 7: Reserved. These bits always read 0. The write value should always be 0.

Bits 6 to 0: Interrupt Vector Number. Sets the interrupt destination vector number. Only the 7 bits 6–0 are valid (as the vector number).

### 10.2.5 Dividend Register H (DVDNTH)

| Bit: | 31 | 30 | 39 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

Dividend register H (DVDNTH) is a 32-bit read/write register in which the upper 32 bits of the dividend used for 64 bit ÷ 32 bit division operations are written. When a division operation is executed, the value set as the dividend is lost and the remainder written here at the end of the operation. The initial value of DVDNTH is undefined, and its value is also undefined after a power-on reset or manual reset, in standby mode, and during in module standby. When the DVDNT register is set with a dividend value, the previous DVDNTH value is lost and the MSB of the DVDNT register is extended to all bits in the DVDNTH register.

### 10.2.6 Dividend Register L (DVDNTL)

| Bit: | 31 | 30 | 39 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | — | — | — | … | — | — | — | — |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

Dividend register L (DVDNTL) is a 32-bit read/write register in which the lower 32 bits of the dividend used for 64-bit ÷ 32-bit division operations are written. When a value is set in this register, the 64-bit ÷ 32-bit division operation begins. The value written in the DVDNT register for 32-bit ÷ 32-bit division is also set in this register. When a 64-bit ÷ 32-bit division operation is executed, the value set as the dividend is lost and the quotient written here at the end of the operation. The contents of this register are undefined after a power-on reset or manual reset, in standby mode, and during module standby.

## 10.3 Operation

### 10.3.1 64-Bit ÷ 32-Bit Operations

64-bit ÷ 32-bit operations work as follows:

1. The 32-bit divisor is set in the divisor register (DVSR).
2. The 64-bit dividend is set in dividend registers H and L (DVDNTH and DVDNTL). First set the value in DVDNTH. When a value is written to DVDNTL, the 64-bit ÷ 32-bit operation begins.
3. This unit finishes a single operation in 39 cycles (starting from the setting of the value in DVDNTL). When an overflow occurs, however, the operation ends in 6 cycles. See section 10.3.3, Handling of Overflows, for more information. Note that operation is signed.
4. After the operation, the 32-bit remainder is written to DVDNTH and the 32-bit quotient is written to DVDNTL.

### 10.3.2 32-Bit ÷ 32-Bit Operations

32-bit ÷ 32-bit operations work as follows:

1. The 32-bit divisor is set in the divisor register (DVSR).
2. The 32-bit dividend is set in dividend register L (DVDNT) for 32-bit division. When a value is written to DVDNT, the 32-bit ÷ 32-bit operation begins.
3. This unit finishes a single operation in 39 cycles (starting from the setting of the value in DVDNT). When an overflow occurs, however, the operation ends in 6 cycles. See section 10.3.3, Handling of Overflows, for more information. Note that the operation is signed.

4. After the operation, the 32-bit remainder is written to DVDNTH and the 32-bit quotient is written to DVDNT.

### 10.3.3 Handling of Overflows

When the results of operations exceed the ranges expressed as signed 32 bits (when, in division between two negative numbers, the quotient is the maximum value and a remainder (negative number) is generated, or, in division of a positive number by a negative number, the quotient is the maximum value) or when the divisor is 0, an overflow will result.

When an overflow occurs, the OVF bit is set and an overflow interrupt is generated if interrupt generation is enabled (the OVFIE bit in DVCR is 1). The operation will then end with the result after 6 cycles of operation stored in the DVDNTH and DVDNTL registers. If interrupt generation is disabled (the OVFIE bit is 0), the operation will end with the operation result at 6 cycles set in DVDNTH and the maximum value H'7FFFFFFF or minimum value H'80000000 set in DVDNTL. In the SH7612, the maximum value results when a positive quotient overflows; the minimum value results when a negative quotient overflows. The first three cycles of the 6 cycles executed when an overflow occurs are used for flag setting within the division unit and the next three for division.

## 10.4 Usage Notes

### 10.4.1 Access

All accesses to the division unit except DVCR and VCRDIV must be 32-bit reads or writes. Word accesses to registers other than DVCR and VCRDIV result in reading or writing of undefined values. In the division unit, a read instruction is extended for one cycle immediately after an instruction that writes to a register, even if the register is the same, to ensure that the value written is accurately set in the destination register in the division unit.

When a read or write instruction is issued while the division unit is operating, the read or write instruction is continuously extended until the operation ends. This means that instructions that do not access the division unit can be parallel-processed. When an instruction is executed that writes to any register of the division unit immediately following an instruction that writes to the division start-up registers (DVDNTL or DVDNT), the correct value may not be set in the start-up register. Specify an instruction other than one that writes to a division unit register for the instruction immediately following instruction that writes to a start-up register.

Because of the above restrictions, efficient processing can be achieved by executing instructions that do not access the division unit for 39 cycles after starting the operation, then issuing a read instruction after the 39th cycle.

### 10.4.2 Overflow Flag

When an overflow occurs, the overflow flag (OVF) is set and is not automatically reset. When OVF is set, the operation is not affected. When necessary, clear it before the operation. The states of registers when overflow occurs are shown in table 10.2.

**Table 10.2 Overflow Processing**

| Register | Overflow Interrupt Enabled | Overflow Interrupt Disabled |
|---|---|---|
| DVSR | Holds the value written | Holds the value written |
| DVDNT | Holds the results of operations until overflow generation is detected* | The maximum value is set for overflow to the plus side, or the minimum value for overflow to the minus side |
| DVCR | The OVF bit is set | The OVF bit is set |
| VCRDIV | Holds the value written | Holds the value written |
| DVDNTH | Holds the results of operations until overflow generation is detected* | Holds the results of operations until overflow generation is detected * |
| DVDNTL | Holds the results of operations until overflow generation is detected* | The maximum value is set for overflow to the plus side, or the minimum value for overflow to the minus side |

Note: * In division processing, the intermediate operation result is written for cycles up to detection of overflow generation.

# Section 11 16-Bit Free-Running Timer (FRT)

## 11.1 Overview

A single-channel, 16-bit free-running timer (FRT) is included on-chip. The FRT is based on a 16-bit free-running counter (FRC) and can output two types of independent waveforms. The FRT can also measure the width of input pulses and the cycle of external clocks.

### 11.1.1 Features

The FRT has the following features:

- Choice of four counter input clocks
  The counter input clock can be selected from three internal clocks (Pϕ/8, Pϕ/32, Pϕ/128) and an external clock (enabling external event counting).
- Two independent comparators
  Two waveform outputs can be generated.
- Input capture
  Choice of rising edge or falling edge
- Counter clear specification
  The counter value can be cleared by compare match A.
- Four interrupt sources
  Two compare match sources, one input capture source, and one overflow source can issue requests independently.

### 11.1.2 Block Diagram

Figure 11.1 shows a block diagram of the FRT.

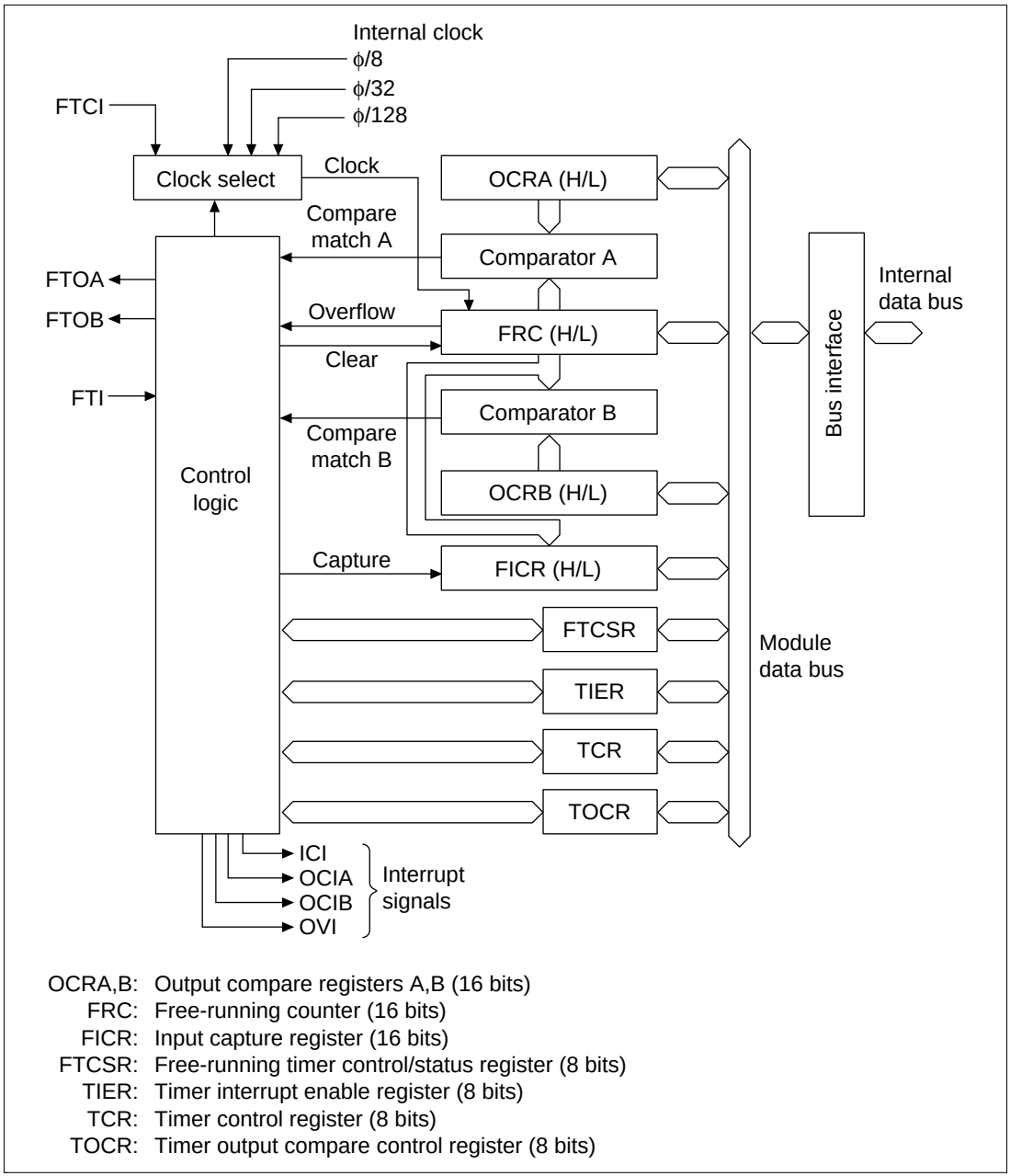



**Figure 11.1 FRT Block Diagram**

### 11.1.3 Pin Configuration

Table 11.1 lists FRT I/O pins and their functions.

**Table 11.1 Pin Configuration**

| Channel | Pin | I/O | Function |
|---|---|---|---|
| Counter clock input pin | FTCI | I | FRC counter clock input pin |
| Output compare A output pin | FTOA | O | Output pin for output compare A |
| Output compare B output pin | FTOB | O | Output pin for output compare B |
| Input capture input pin | FTI | I | Input pin for input capture |

### 11.1.4 Register Configuration

Table 11.2 shows the FRT register configuration.

**Table 11.2 Register Configuration**

| Register | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer interrupt enable register | TIER | R/W | H'01 | HFFFFFE10 |
| Free-running timer control/status register | FTCSR | R/(W)*1 | H'00 | HFFFFFE11 |
| Free-running counter H | FRC H | R/W | H'00 | HFFFFFE12 |
| Free-running counter L | FRC L | R/W | H'00 | HFFFFFE13 |
| Output compare register A H | OCRA H | R/W | H'FF | HFFFFFE14*2 |
| Output compare register A L | OCRA L | R/W | H'FF | HFFFFFE15*2 |
| Output compare register B H | OCRB H | R/W | H'FF | HFFFFFE14*2 |
| Output compare register B L | OCRB L | R/W | H'FF | HFFFFFE15*2 |
| Timer control register | TCR | R/W | H'00 | HFFFFFE16 |
| Timer output compare control register | TOCR | R/W | H'E0 | HFFFFFE17 |
| Input capture register H | FICR H | R | H'00 | HFFFFFE18 |
| Input capture register L | FICR L | R | H'00 | HFFFFFE19 |

Notes: 1. Bits 7 to 1 are read-only. The only value that can be written is a 0, which is used to clear flags. Bit 0 can be read or written.
2. OCRA and OCRB have the same address. The OCRS bit in TOCR is used to switch between them.
3. Use byte-size access for all registers.

## 11.2 Register Descriptions

### 11.2.1 Free-Running Counter (FRC)

| Bit: | 15 | 14 | 13 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

FRC is a 16-bit read/write register. It increments upon input of a clock. The input clock can be selected using clock select bits 1 and 0 (CKS1, CKS0) in TCR. FRC can be cleared upon compare match A.

When FRC overflows (H'FFFF → H'0000), the overflow flag (OVF) in FTCSR is set to 1. FRC can be read or written to by the CPU, but because it is 16 bits long, data transfers involving the CPU are performed via a temporary register (TEMP). See section 11.3, CPU Interface, for more detailed information.

FRC is initialized to H'0000 by a reset, in standby mode, and when the module standby function is used.

### 11.2.2 Output Compare Registers A and B (OCRA and OCRB)

| Bit: | 15 | 14 | 13 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | 1 | 1 | 1 | … | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | … | R/W | R/W | R/W | R/W |

OCR is composed of two 16-bit read/write registers (OCRA and OCRB). The contents of OCR are always compared to the FRC value. When the two values are the same, the output compare flags in FTCSR (OCFA and OCFB) are set to 1.

When the OCR and FRC values are the same (compare match), the output level values set in the output level bits (OLVLA and OLVLB) are output to the output compare pins (FTOA and FTOB). After a reset, FTOA and FTOB output 0 until the first compare match occurs.

Because OCR is a 16-bit register, data transfers involving the CPU are performed via a temporary register (TEMP). See section 11.3, CPU Interface, for more detailed information.

OCR is initialized to H'FFFF by a reset, in standby mode, and when the module standby function is used.

### 11.2.3 Input Capture Register (FICR)

| Bit: | 15 | 14 | 13 | … | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | … | | | | |
| Initial value: | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | … | R | R | R | R |

FICR is a 16-bit read-only register. When a rising edge or falling edge of the input capture signal (FTI pin) is detected, the current FRC value is transferred to ICR. At the same time, the input capture flag (ICF) in FTCSR is set to 1. The edge of the input signal can be selected using the input edge select bit (IEDG) in TCR.

Because ICR is a 16-bit register, data transfers involving the CPU are performed via a temporary register (TEMP). See Section 11.3, CPU Interface, for more detailed information.

FICR is initialized to H'0000 by a reset, in standby mode, and when the module standby function is used.

### 11.2.4 Timer Interrupt Enable Register (TIER)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | ICIE | — | — | — | OCIAE | OCIBE | OVIE | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R/W | — | — | — | R/W | R/W | R/W | — |

TIER is an 8-bit read/write register that controls enabling of all interrupt requests. TIER is initialized to H'01 by a reset, in standby mode, and when the module standby function is used.

Bit 7—Input Capture Interrupt Enable (ICIE): Selects enabling/disabling of the ICI interrupt request when the input capture flag (ICF) in FTCSR is set to 1.

| Bit 7: ICIE | Description | |
|---|---|---|
| 0 | Interrupt request (ICI) caused by ICF disabled | (Initial value) |
| 1 | Interrupt request (ICI) caused by ICF enabled | |

Bits 6 to 4—Reserved: These bits always read 0. The write value should always be 0.

Bit 3—Output Compare Interrupt A Enable (OCIAE): Selects enabling/disabling of the OCIA interrupt request when the output compare flag A (OCFA) in FTCSR is set to 1.

| Bit 3: OCIAE | Description | |
|---|---|---|
| 0 | Interrupt request (OCIA) caused by OCFA disabled | (Initial value) |
| 1 | Interrupt request (OCIA) caused by OCFA enabled | |

Bit 2—Output Compare Interrupt B Enable (OCIBE): Selects enabling/disabling of the OCIB interrupt request when the output compare flag B (OCFB) in FTCSR is set to 1.

| Bit 2: OCIBE | Description | |
|---|---|---|
| 0 | Interrupt request (OCIB) caused by OCFB disabled | (Initial value) |
| 1 | Interrupt request (OCIB) caused by OCFB enabled | |

Bit 1—Timer Overflow Interrupt Enable (OVIE): Selects enabling/disabling of the OVI interrupt request when the overflow flag (OVF) in FTCSR is set to 1.

| Bit 1: OVIE | Description | |
|---|---|---|
| 0 | Interrupt request (FOVI) caused by OVF disabled | (initial value) |
| 1 | Interrupt request (FOVI) caused by OVF enabled | |

Bit 0—Reserved: This bit always reads 1. The write value should always be 1.

### 11.2.5 Free-Running Timer Control/Status Register (FTCSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | ICF | — | — | — | OCFA | OCFB | OVF | CCLRA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/(W)* | — | — | — | R/(W)* | R/(W)* | R/(W)* | R/W |

Note: For bits 7, and 3 to 1, the only value that can be written is 0 (to clear the flags).

FTCSR is an 8-bit register that selects counter clearing and controls interrupt request signals. FTCSR is initialized to H'00 by a reset, in standby mode, and when the module standby function is used. See section 11.4, Operation, for the timing.

Bit 7—Input Capture Flag (ICF): Status flag that indicates that the FRC value has been sent to FICR by the input capture signal. This flag is cleared by software and set by hardware. It cannot be set by software.

| Bit 7: ICF | Description |
|---|---|
| 0 | Clear conditions: When ICF is read while set to 1, and then 0 is written to it (Initial value) |
| 1 | Set conditions: When the FRC value is sent to FICR by the input capture signal |

Bits 6 to 4—Reserved: These bits always read 0. The write value should always be 0.

Bit 3—Output Compare Flag A (OCFA): Status flag that indicates when the values of the FRC and OCRA match. This flag is cleared by software and set by hardware. It cannot be set by software.

| Bit 3: OCFA | Description |
|---|---|
| 0 | Clear conditions: When OCFA is read while set to 1, and then 0 is written to it (Initial value) |
| 1 | Set conditions: When the FRC value becomes equal to OCRA |

Bit 2—Output Compare Flag B (OCFB): Status flag that indicates when the values of FRC and OCRB match. This flag is cleared by software and set by hardware. It cannot be set by software.

| Bit 2: OCFB | Description |
|---|---|
| 0 | Clear conditions: When OCFB is read while set to 1, and then 0 is written to it (Initial value) |
| 1 | Set conditions: When the FRC value becomes equal to OCRB |

Bit 1—Timer Overflow Flag (OVF): Status flag that indicates when FRC overflows (from H'FFFF to H'0000). This flag is cleared by software and set by hardware. It cannot be set by software.

| Bit 1: OVF | Description |
|---|---|
| 0 | Clear conditions: When OVF is read while set to 1, and then 0 is written to it (Initial value) |
| 1 | Set conditions: When the FRC value changes from H'FFFF to H'0000 |

Bit 0—Counter Clear A (CCLRA): Selects whether or not to clear FRC on compare match A (signal indicating match of FRC and OCRA).

| Bit 0: CCLRA | Description |
|---|---|
| 0 | FRC clear disabled (Initial value) |
| 1 | FRC cleared on compare match A |

### 11.2.6 Timer Control Register (TCR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IEDG | — | — | — | — | — | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

TCR is an 8-bit read/write register that selects the input edge for input capture and selects the input clock for FRC. TCR is initialized to H'00 by a reset, in standby mode, and when the module standby function is used.

Bit 7—Input Edge Select (IEDG): Selects whether to capture the input capture input (FTI) on the falling edge or rising edge.

| Bit 7: IEDG | Description | |
|---|---|---|
| 0 | Input captured on falling edge | (Initial value) |
| 1 | Input captured on rising edge | |

Bits 6 to 2—Reserved: These bits always read 0. The write value should always be 0.

Bits 1 and 0—Clock Select (CKS1, CKS0): These bits select whether to use an external clock or one of three internal clocks for input to FRC. The external clock is counted at the rising edge.

| Bit 1: CKS1 | Bit 0: CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | Internal clock: count at $\phi$/8 | (Initial value) |
| | 1 | Internal clock: count at $\phi$/32 | |
| 1 | 0 | Internal clock: count at $\phi$/128 | |
| | 1 | External clock: count at rising edge | |

### 11.2.7 Timer Output Compare Control Register (TOCR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | OCRS | — | — | OLVLA | OLVLB |
| Initial value: | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | — | — | — | R/W | R/W | R/W | R/W | R/W |

TOCR is an 8-bit read/write register that selects the output level for output compare and controls switching between access of output compare registers A and B. TOCR is initialized to H'E0 by a reset, in standby mode, and when the module standby function is used.

Bits 7 to 5—Reserved: These bits always read 1. The write value should always be 1.

Bit 4—Output Compare Register Select (OCRS): OCRA and OCRB share the same address. The OCRS bit controls which register is selected when reading/writing to this address. It does not affect the operation of OCRA and OCRB.

| Bit 4: OCRS | Description | |
|---|---|---|
| 0 | OCRA register selected | (Initial value) |
| 1 | OCRB register selected | |

Bits 3 and 2—Reserved: These bits always read 0. The write value should always be 0.

Bit 1—Output Level A (OLVLA): Selects the level output to the output compare A output pin upon compare match A (signal indicating match of FRC and OCRA).

| Bit 1: OLVLA | Description | |
|---|---|---|
| 0 | 0 output on compare match A | (Initial value) |
| 1 | 1 output on compare match A | |

Bit 0—Output Level B (OLVLB): Selects the level output to the output compare B output pin upon compare match B (signal indicating match of FRC and OCRB).

| Bit 0: OLVLB | Description | |
|---|---|---|
| 0 | 0 output on compare match B | (Initial value) |
| 1 | 1 output on compare match B | |

## 11.3 CPU Interface

FRC, OCRA, OCRB, and FICR are 16-bit registers. The data bus width between the CPU and FRT, however, is only 8 bits. Access of these three types of registers from the CPU therefore needs to be performed via an 8-bit temporary register called TEMP.

The following describes how these registers are read from and written to:

- Writing to 16-bit Registers

  The upper byte is written, which results in the upper byte of data being stored in TEMP. The lower byte is then written, which results in 16 bits of data being written to the register when combined with the upper byte value in TEMP.

- Reading from 16-bit Registers

  The upper byte of data is read, which results in the upper byte value being transferred to the CPU. The lower byte value is transferred to TEMP. The lower byte is then read, which results in the lower byte value in TEMP being sent to the CPU.

When registers of these three types are accessed, two byte accesses should always be performed, first to the upper byte, then the lower byte. The same applies to accesses with the on-chip direct memory access controller. If only the upper byte or lower byte is accessed, the data will not be transferred properly.

Figure 11.2 and 11.3 show the flow of data when FRC is accessed. Other registers function in the same way. When reading OCRA and OCRB, however, both upper and lower-byte data is transferred directly to the CPU without passing through TEMP.

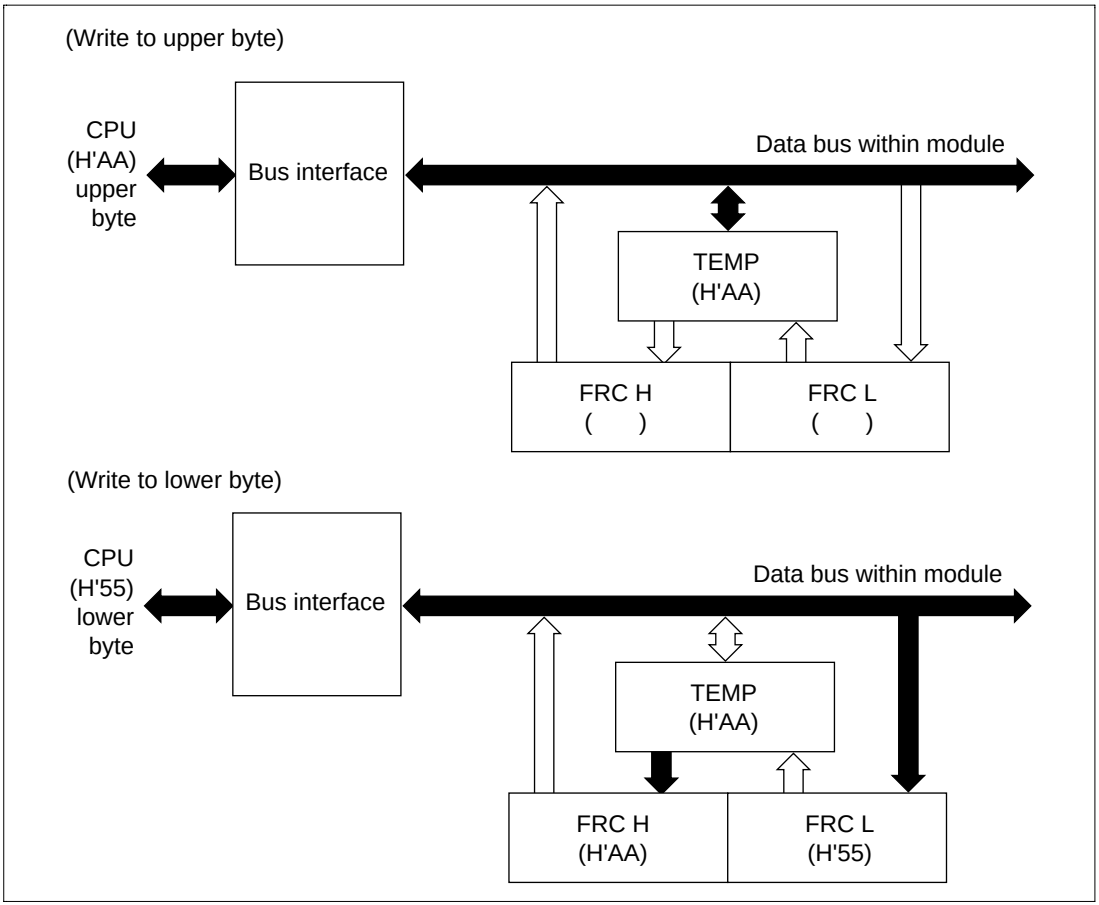



**Figure 11.2 FRC Access Operation (CPU Writes H'AA55 to FRC)**

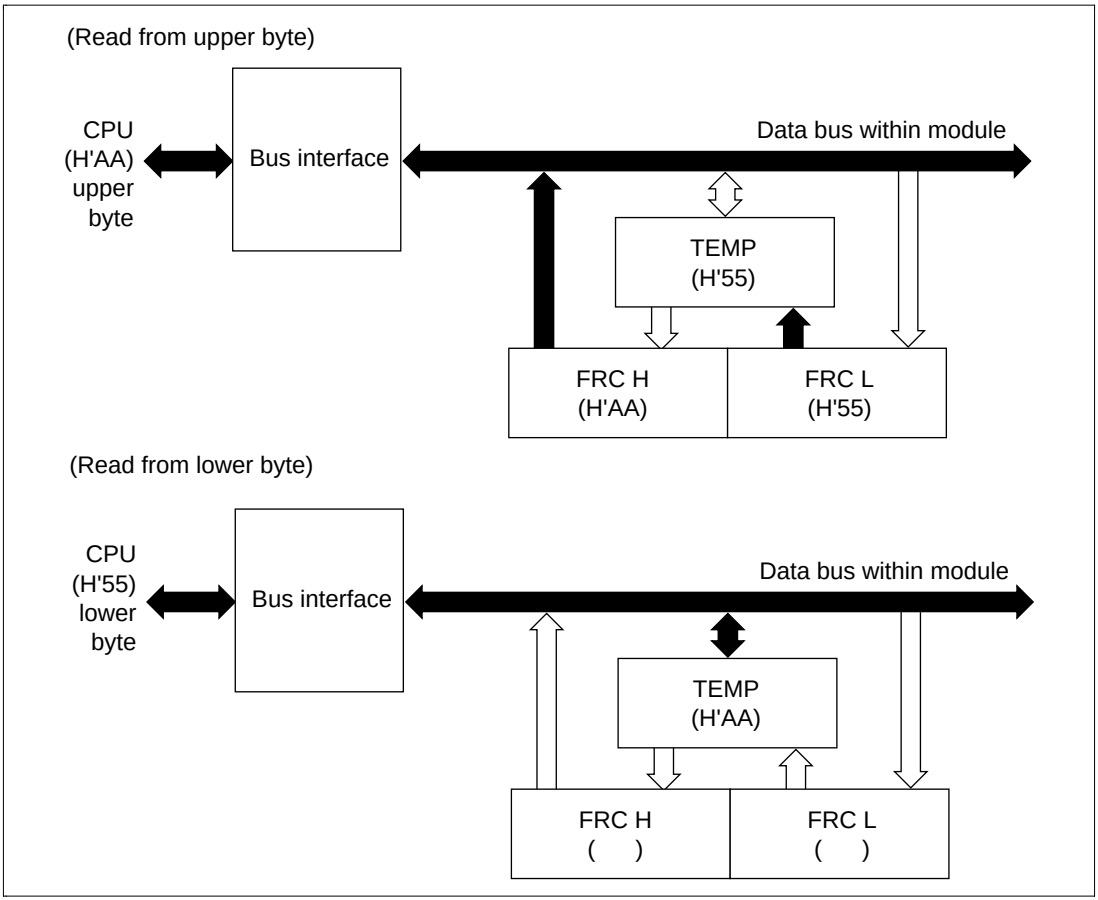



**Figure 11.3 FRC Access Operation (CPU Reads H'AA55 from FRC)**

## 11.4 Operation

### 11.4.1 FRC Count Timing

The FRC increments on clock input (internal or external).

**Internal Clock Operation:** Set the CKS1 and CKS0 bits in TCR to select which of the three internal clocks created by dividing system clock $\phi$ ($\phi$/8, $\phi$/32, $\phi$/128) is used. Figure 11.4 shows the timing.

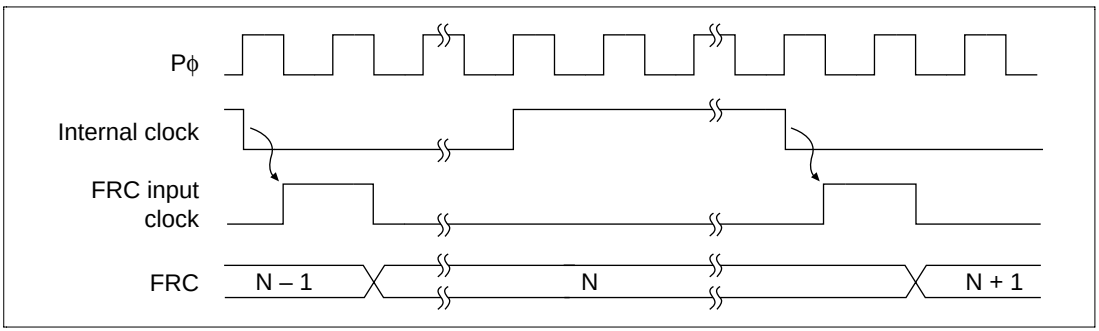



**Figure 11.4 Count Timing (Internal Clock Operation)**

**External Clock Operation:** Set the CKS1 and CKS0 bits in TCR to select the external clock. External clock pulses are counted on the rising edge. Figures 11.5 shows the timing.

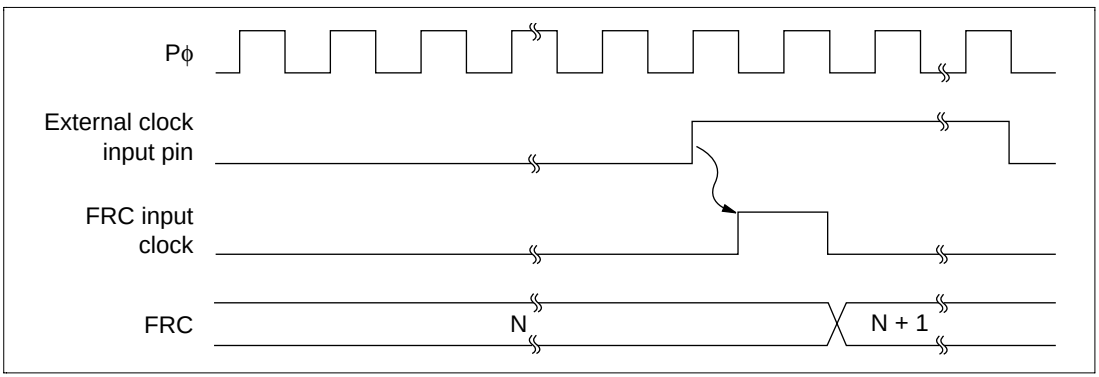



**Figure 11.5 Count Timing (External Clock Operation)**

### 11.4.2 Output Timing for Output Compare

When a compare match occurs, the output level set in the OLVL bit in TOCR is output from the output compare output pins (FTOA, FTOB). Figure 11.6 shows the timing for output of output compare A.

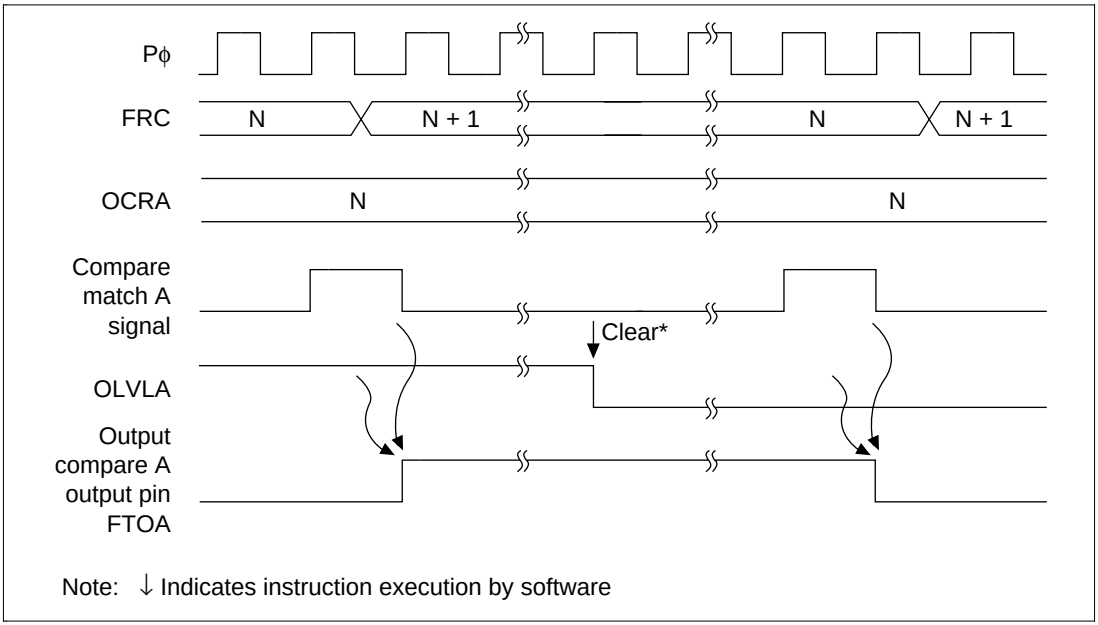



**Figure 11.6 Output Timing for Output Compare A**

### 11.4.3 FRC Clear Timing

FRC can be cleared on compare match A. Figure 11.7 shows the timing.

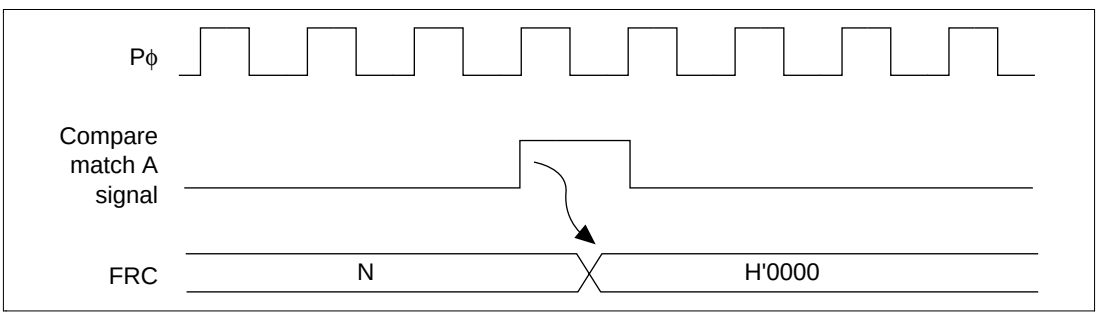



**Figure 11.7 Compare Match A Clear Timing**

### 11.4.4 Input Capture Input Timing

Either the rising edge or falling edge can be selected for input capture input using the IEDG bit in TCR. Figure 11.8 shows the timing when the rising edge is selected (IEDG = 1).

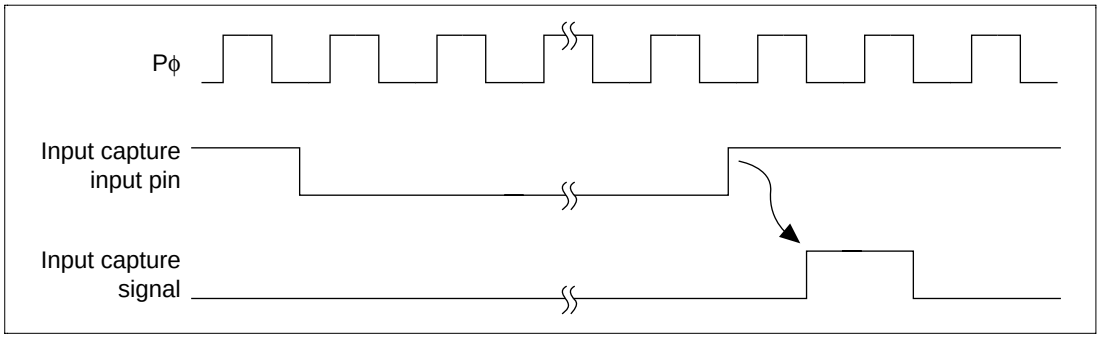



**Figure 11.8 Input Capture Signal Timing (Normal)**

When the input capture signal is input when FICR is read (upper-byte read), the input capture signal is delayed by one cycle of Pϕ. Figure 11.9 shows the timing.

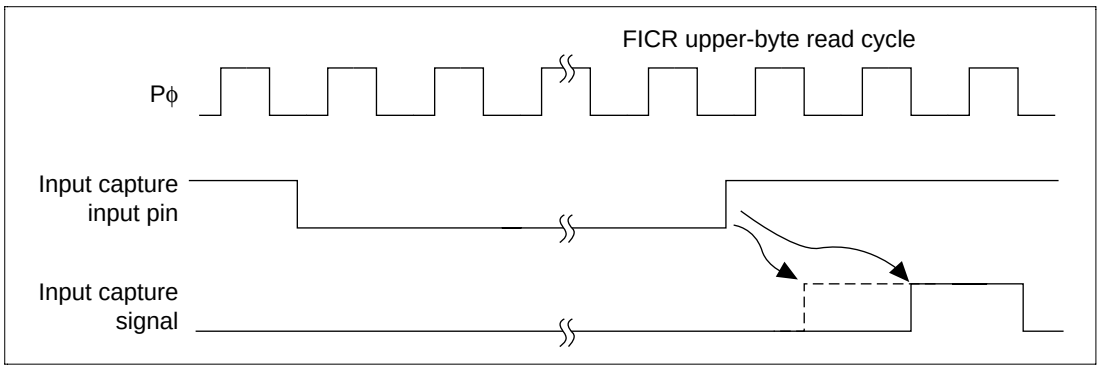



**Figure 11.9 Input Capture Signal Timing (Input Capture Input when FICR is Read)**

### 11.4.5 Input Capture Flag (ICF) Setting Timing

Input capture input sets the input capture flag (ICF) to 1 and simultaneously transfers the FRC value to FICR. Figure 11.10 shows the timing.

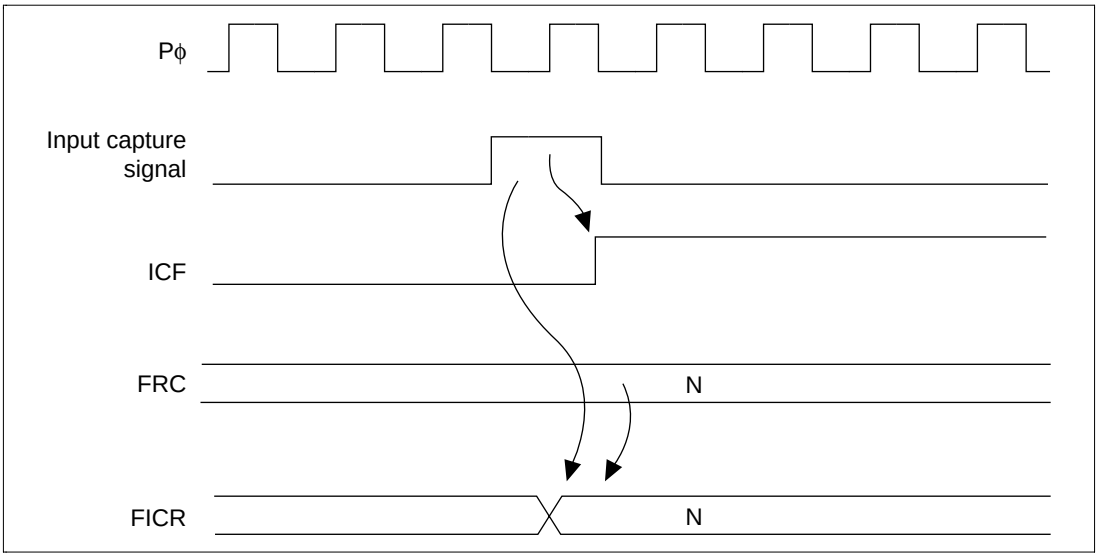



**Figure 11.10 ICF Setting Timing**

### 11.4.6 Output Compare Flag (OCFA, OCFB) Setting Timing

The compare match signal output (when OCRA or OCRB matches the FRC value) sets output compare flag OCFA or OCFB to 1. The compare match signal is generated in the last state in which the values matched (at the timing for updating the count value that matched the FRC). After OCRA or OCRB matches the FRC, no compare match is generated until the next increment occurs. Figure 11.11 shows the timing for setting OCFA and OCFB.

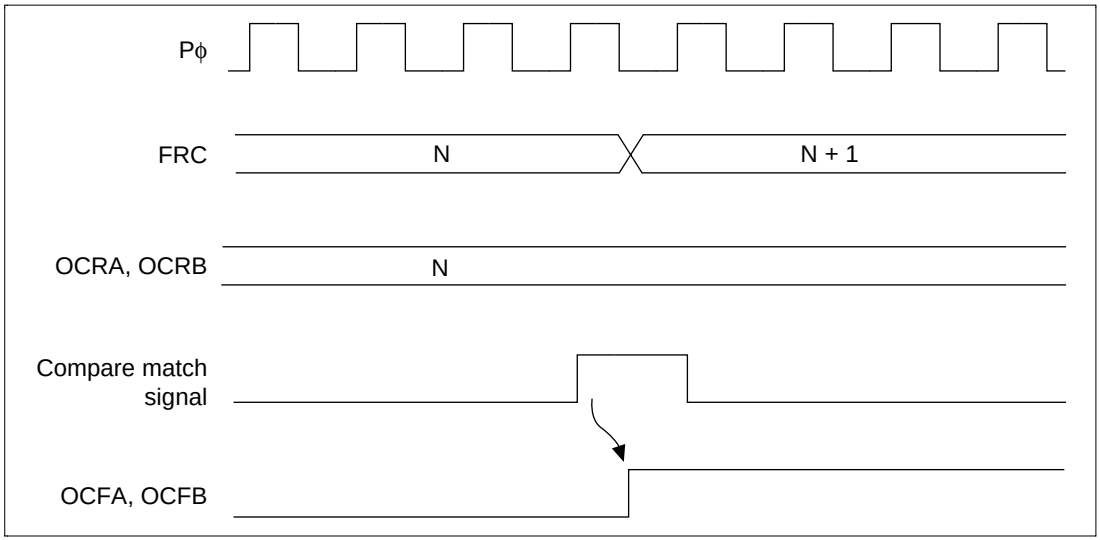



**Figure 11.11 OCF Setting Timing**

### 11.4.7 Timer Overflow Flag (OVF) Setting Timing

FRC overflow (from H'FFFF to H'0000) sets the timer overflow flag (OVF) to 1. Figure 11.12 shows the timing.

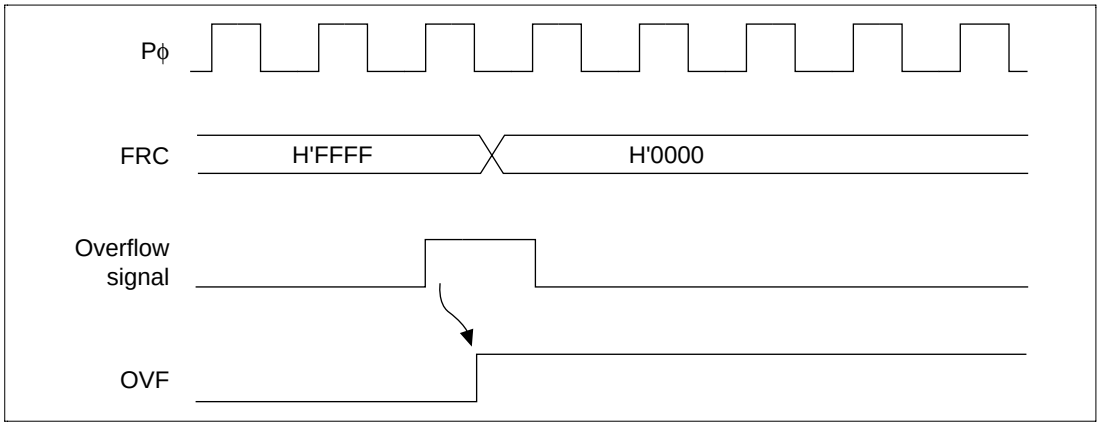



**Figure 11.12 OVF Setting Timing**

## 11.5 Interrupt Sources

There are four FRT interrupt sources of three types (ICI, OCIA/OCIB, and OVI). Table 11.3 lists the interrupt sources and their priorities after a reset is cleared. The interrupt enable bits in TIER are used to enable or disable the interrupt bits. Each interrupt request is sent to the interrupt controller independently. See section 4, Exception Handling, for more information about priorities and the relationship to interrupts other than those of the FRT.

**Table 11.3 FRT Interrupt Sources and Priorities**

| Interrupt Source | Description | Priority |
|---|---|---|
| ICI | Interrupt by ICF | High |
| OCIA, OCIB | Interrupt by OCFA or OCFB | ↕ |
| OVI | Interrupt by OVF | Low |

## 11.6 Example of FRT Use

Figure 11.13 shows an example in which pulses with a 50% duty factor and arbitrary phase relationship are output. The procedure is as follows:

1. Set the CCLRA bit in FTCSR to 1.
2. The OLVLA and OLVLB bits are inverted by software whenever a compare match occurs.

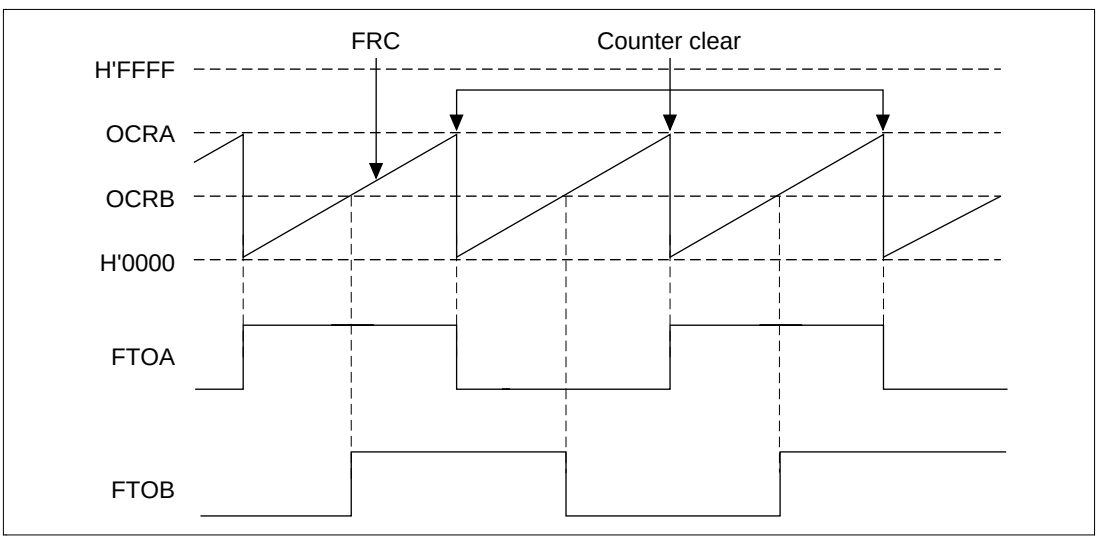



**Figure 11.13 Example of Pulse Output**

## 11.7 Usage Notes

Note that the following contention and operations occur when the FRT is operating:

### 11.7.1 Contention between FRC Write and Clear

When a counter clear signal is generated with the timing shown in figure 11.14 during the write cycle for the lower byte of FRC, writing does not occur to the FRC, and the FRC clear takes priority.

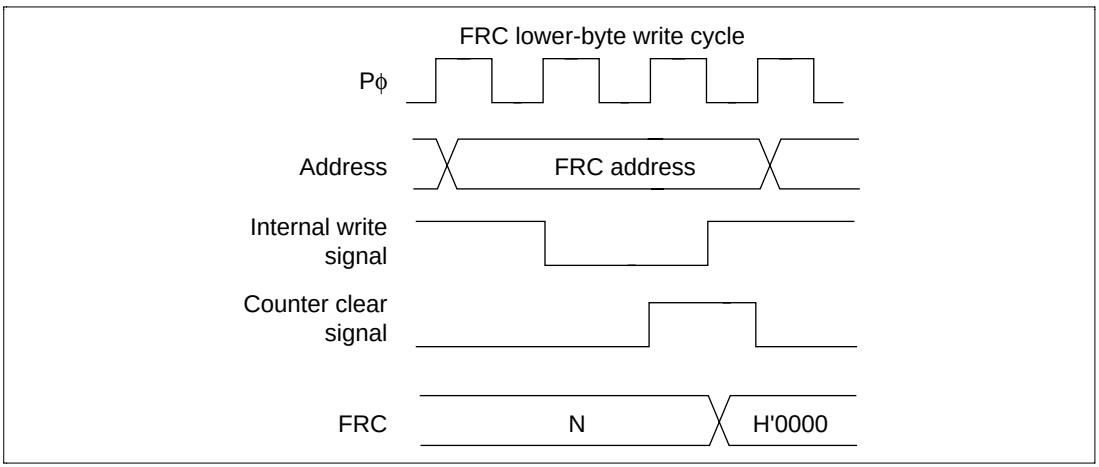



**Figure 11.14 Contention between FRC Write and Clear**

### 11.7.2 Contention between FRC Write and Increment

When an increment occurs with the timing shown in figure 11.15 during the write cycle for the lower byte of FRC, no increment is performed and the counter write takes priority.

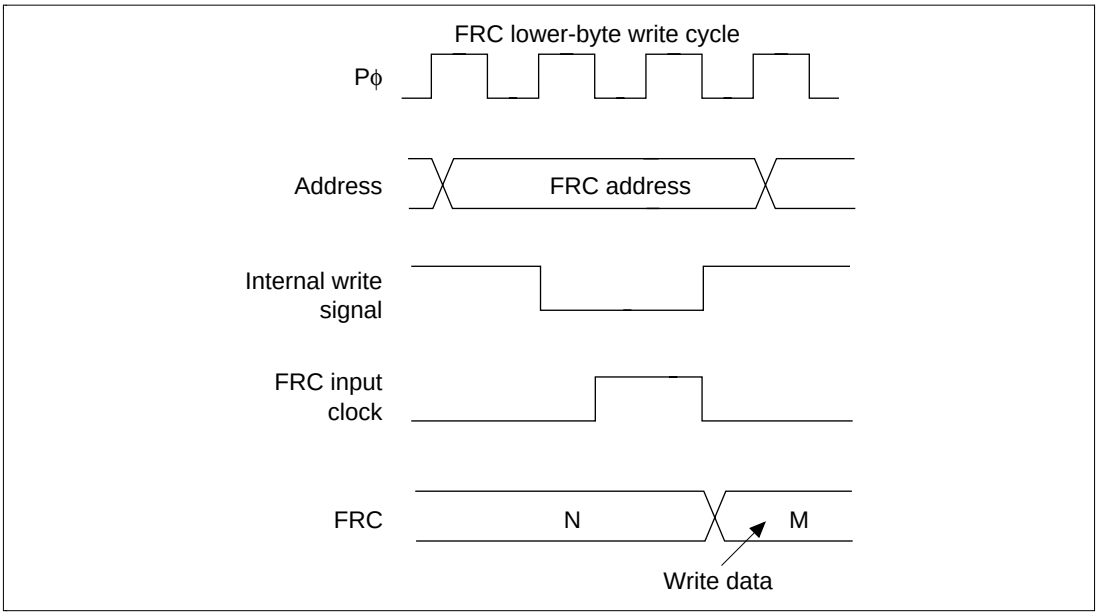



**Figure 11.15 Contention between FRC Write and Increment**

### 11.7.3 Contention between OCR Write and Compare Match

When a compare match occurs with the timing shown in figure 11.16, during the write cycle for the lower byte of OCRA or OCRB, the OCR write takes priority and the compare match signal is disabled.

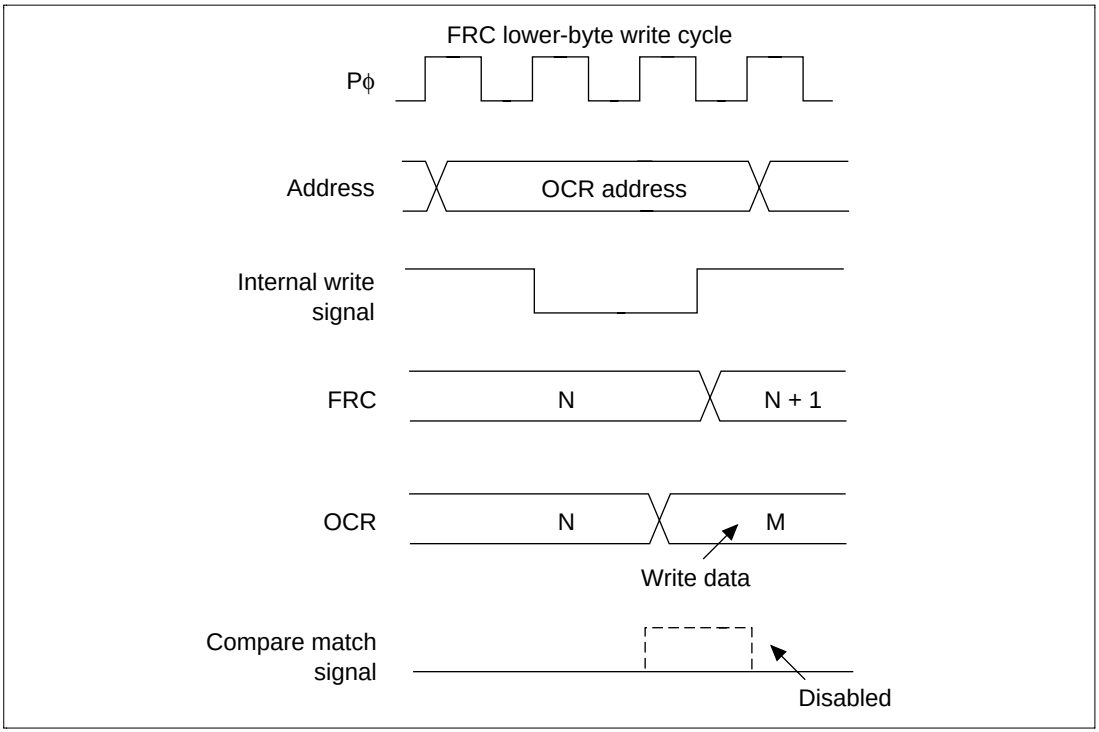



**Figure 11.16 Contention between OCR and Compare Match**

### 11.7.4 Internal Clock Switching and Counter Operation

FRC will sometimes begin incrementing because of the timing of switching between internal clocks. Table 11.4 shows the relationship between internal clock switching timing (CKS1 and CKS0 bit rewrites) and FRC operation.

When an internal clock is used, the FRC clock is generated when the falling edge of an internal clock (created by dividing the system clock (ϕ)) is detected. When a clock is switched to high before the switching and to low after switching, as shown in case 3 in table 11.4, the switchover is considered a falling edge and an FRC clock pulse is generated, causing FRC to increment. FRC may also increment when switching between an internal clock and an external clock.

**Table 11.4 Internal Clock Switching and FRC Operation**

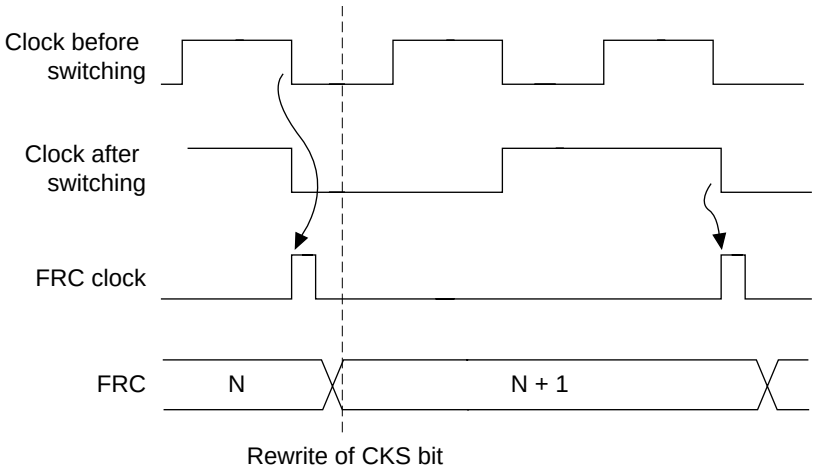
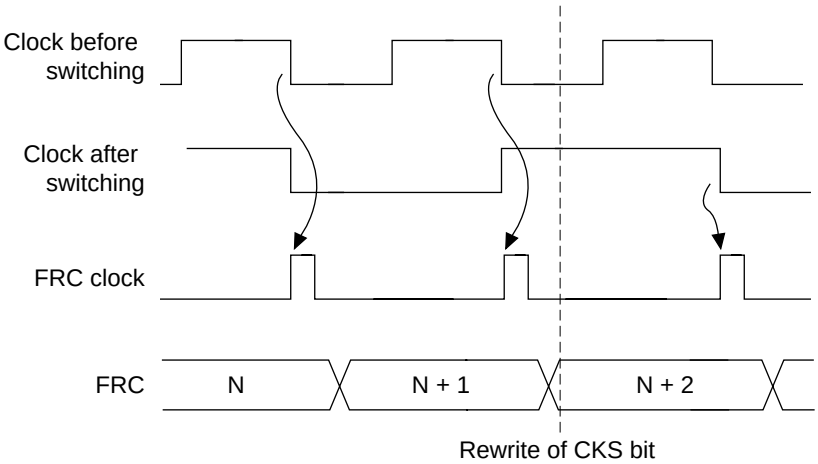
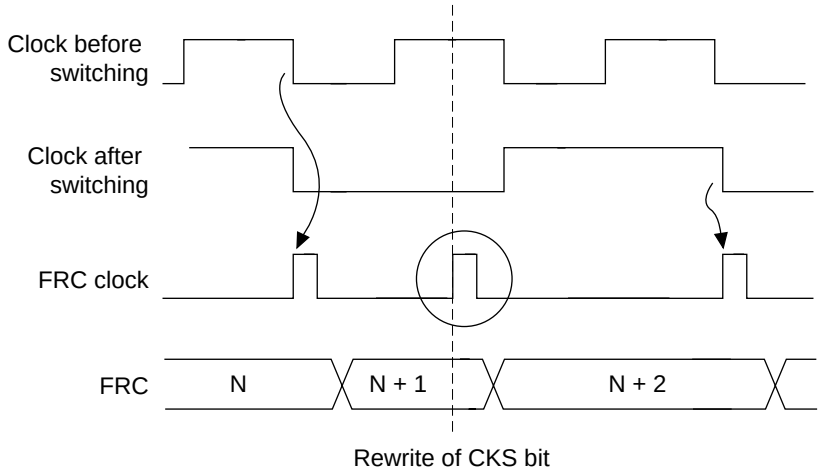
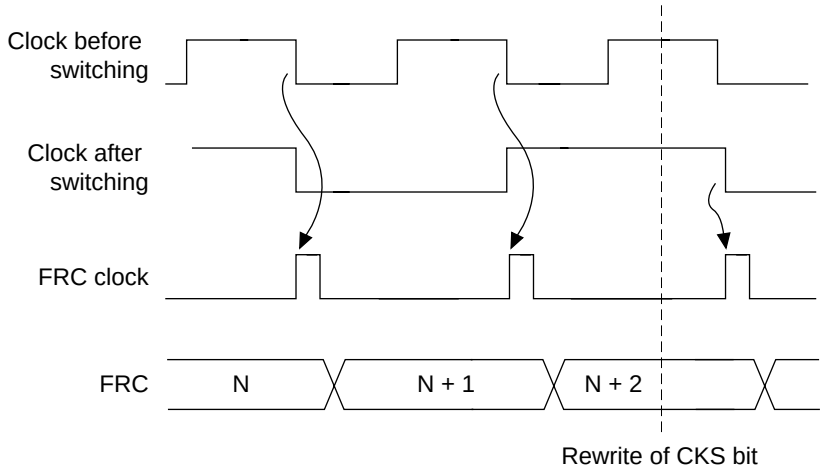
| No. | Timing of Rewrite of CKS1 and CKS0 Bits | FRC Operation |
|---|---|---|
| 1 | Low-to-low switch |  |
| 2 | Low-to-high switch |  |

**Table 11.4 Internal Clock Switching and FRC Operation (cont)**

| No. | Timing of Rewrite of CKS1 and CKS0 Bits | FRC Operation |
|---|---|---|
| 3 | High-to-low switch |  |
| 4 | High-to-high switch |  |

Note: Because the switchover is considered a falling edge, FRC starts counting up.

### 11.7.5 Timer Output (FTOA, FTOB)

During a power-on reset, the timer outputs (FTOA, FTOB) will be unreliable until the oscillation stabilizes. The initial value is output after the oscillation settling time has elapsed.

# Section 12 Watchdog Timer (WDT)

## 12.1 Overview

A single-channel watchdog timer (WDT) is provided on-chip for monitoring system operations. If a system becomes uncontrolled and the timer counter overflows without being rewritten correctly by the CPU, an overflow signal ($\overline{\text{WDTOVF}}$) is output externally. The WDT can simultaneously generate an internal reset signal for the entire chip.

When this watchdog function is not needed, the WDT can be used as an interval timer. In the interval timer operation, an interval timer interrupt is generated at each counter overflow. The WDT is also used when recovering from standby mode, in modifying a clock frequency, and in clock pause mode.

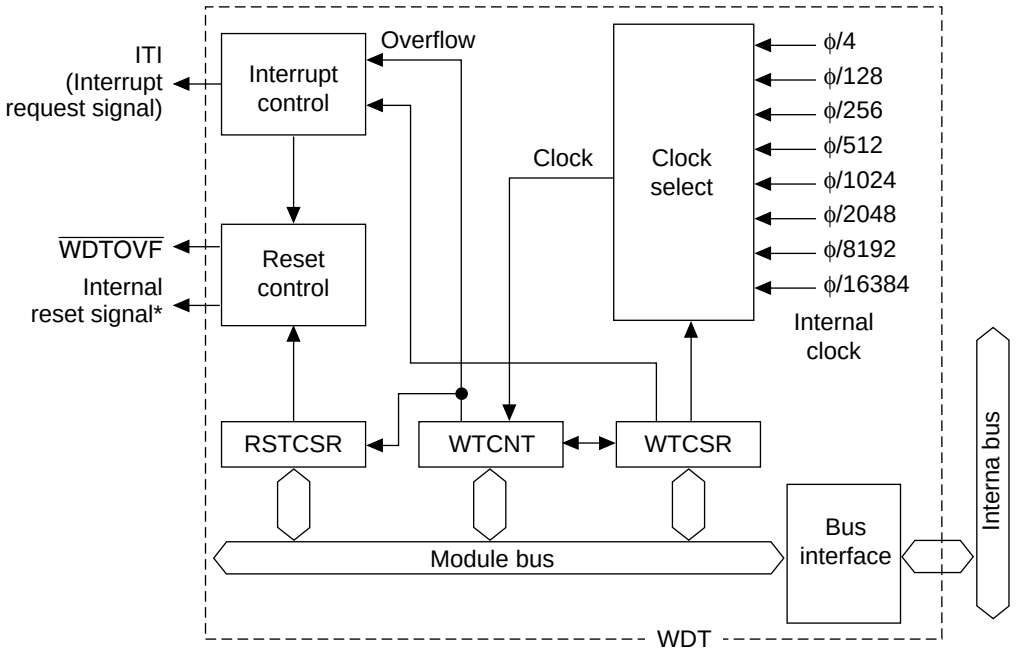
### 12.1.1 Features

The WDT includes the following features.

- Can be switched between watchdog timer mode and interval timer mode.
- $\overline{\text{WDTOVF}}$ output in watchdog timer mode
  The $\overline{\text{WDTOVF}}$ signal is output externally when the counter overflows, and a simultaneous internal reset of the chip can also be selected (either a power-on reset or manual reset can be specified).
- Interrupt generation in interval timer mode
  An interval timer interrupt is generated when the counter overflows.
- Used when standby mode is cleared or the clock frequency is changed, and in clock pause mode.
- Choice of eight counter input clocks

### 12.1.2 Block Diagram

Figure 12.1 shows a block diagram of the WDT.




φ: See figure 3.1, Block Diagram of Clock Pulse Generator Circuit.

WTCSR: Watchdog timer control/status register
WTCNT: Watchdog timer counter
RSTCSR: Reset control/status register

Note: The internal reset signal can be generated by a register setting. The type of reset can be selected (power-on or manual reset).

**Figure 12.1 WDT Block Diagram**

### 12.1.3 Pin Configuration

Table 12.1 shows the pin configuration.

**Table 12.1 Pin Configuration**

| Pin | Abbreviation | I/O | Function |
|---|---|---|---|
| Watchdog timer overflow | $\overline{\text{WDTOVF}}$ | O | Outputs the counter overflow signal in watchdog timer mode |

### 12.1.4 Register Configuration

Table 12.2 summarizes the three WDT registers. They are used to select the clock, switch the WDT mode, and control the reset signal.

**Table 12.2 Register Configuration**

| | | | | Address | |
|---|---|---|---|---|---|
| **Name** | **Abbreviation** | **R/W** | **Initial Value** | **Write**[*1] | **Read**[*2] |
| Watchdog timer control/status register | WTCSR | R/(W)[*3] | H'18 | H'FFFFFE80 | H'FFFFFE80 |
| Watchdog timer counter | WTCNT | R/W | H'00 | H'FFFFFE80 | H'FFFFFE81 |
| Reset control/status register | RSTCSR | R/(W)[*3] | H'1F | H'FFFFFE82 | H'FFFFFE83 |

Notes: 1. Write by word access. It cannot be written by byte or longword access.
2. Read by byte access. The correct value cannot be read by word or longword access.
3. Only 0 can be written in bit 7 to clear the flag.

## 12.2 Register Descriptions

### 12.2.1 Watchdog Timer Counter (WTCNT)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

WTCNT is an 8-bit read/write register. The method of writing to WTCNT differs from that of most other registers to prevent inadvertent rewriting. See section 12.2.4, Notes on Register Access, for details. When the timer enable bit (TME) in the watchdog timer control/status register (WTCSR) is set to 1, the watchdog timer counter starts counting pulses of an internal clock source selected by clock select bits 2 to 0 (CKS2 to CKS0) in WTCSR. When the value of WTCNT overflows (changes from H'FF to H'00), a watchdog timer overflow signal ($\overline{\text{WDTOVF}}$) or interval timer interrupt (ITI) is generated, depending on the mode selected in the WT/$\overline{\text{IT}}$ bit in WTCSR. WTCNT is initialized to H'00 by a reset and when the TME bit is cleared to 0. It is not initialized in standby mode, when the clock frequency is changed, or in clock pause mode.

### 12.2.2 Watchdog Timer Control/Status Register (WTCSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | OVF | WT/$\overline{IT}$ | TME | — | — | CKS2 | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| R/W: | R/(W)* | R/W | R/W | — | — | R/W | R/W | R/W |

The watchdog timer control/status register (WTCSR) is an 8-bit read/write register. The method of writing to WTCSR differs from that of most other registers to prevent inadvertent rewriting. See section 12.2.4, Notes on Register Access, for details. Its functions include selecting the timer mode and clock source. Bits 7 to 5 are initialized to 000 by a reset, in standby mode, when the clock frequency is changed, and in clock pause mode. Bits 2 to 0 are initialized to 000 by a reset, but are not initialized in standby mode, when the clock frequency is changed, or in clock pause mode.

Bit 7—Overflow Flag (OVF): Indicates that WTCNT has overflowed from H'FF to H'00 in interval timer mode. It is not set in watchdog timer mode.

| Bit 7: OVF | Description | |
|---|---|---|
| 0 | No overflow of WTCNT in interval timer mode | (Initial value) |
| | Cleared by reading OVF, then writing 0 in OVF | |
| 1 | WTCNT overflow in interval timer mode | |

Bit 6—Timer Mode Select (WT/$\overline{IT}$): Selects whether to use the WDT as a watchdog timer or interval timer. When WTCNT overflows, the WDT either generates an interval timer interrupt (ITI) or generates a $\overline{WDTOVF}$ signal, depending on the mode selected.

| Bit 6: WT/$\overline{IT}$ | Description | |
|---|---|---|
| 0 | Interval timer mode: interval timer interrupt (ITI) request to the CPU when WTCNT overflows | (Initial value) |
| 1 | Watchdog timer mode: $\overline{WDTOVF}$ signal output externally when WTCNT overflows. Section 12.2.3, Reset Control/Status Register (RSTCSR), describes in detail what happens when WTCNT overflows in watchdog timer mode | |

Bit 5—Timer Enable (TME): Enables or disables the timer.

| Bit 5: TME | Description |
|---|---|
| 0 | Timer disabled: WTCNT is initialized to H'00 and count-up stops (Initial value) |
| 1 | Timer enabled: WTCNT starts counting. A $\overline{\text{WDTOVF}}$ signal or interrupt is generated when WTCNT overflows |

Bits 4 and 3—Reserved: These bits always read 1. The write value should always be 1.

Bits 2 to 0—Clock Select 2 to 0 (CKS2 to CKS0): These bits select one of eight internal clock sources for input to WTCNT. The clock signals are obtained by dividing the frequency of the system clock (φ).

| | | | Description | |
|---|---|---|---|---|
| **Bit 2: CKS2** | **Bit 1: CKS1** | **Bit 0: CKS0** | **Clock Source** | **Overflow Interval* (φ = 60 MHz)** |
| 0 | 0 | 0 | φ/4 (Initial value) | 17.0 μs |
| | | 1 | φ/128 | 544 μs |
| | 1 | 0 | φ/256 | 1.1 ms |
| | | 1 | φ/512 | 2.2 ms |
| 1 | 0 | 0 | φ/1024 | 4.4 ms |
| | | 1 | φ/2048 | 8.7 ms |
| | 1 | 0 | φ/8192 | 34.8 ms |
| | | 1 | φ/16384 | 69.6 ms |

Note: * The overflow interval listed is the time from when the WTCNT begins counting at H'00 until an overflow occurs.

### 12.2.3 Reset Control/Status Register (RSTCSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | WOVF | RSTE | RSTS | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/(W)* | R/W | R/W | — | — | — | — | — |

Note: * Only 0 can be written in bit 7, to clear the flag.

RSTCSR is an 8-bit read/write register that controls output of the reset signal generated by watchdog timer counter (WTCNT) overflow and selects the internal reset signal type. The method of writing to RSTCSR differs from that of most other registers to prevent inadvertent rewriting. See section 12.2.4, Notes on Register Access, for details. RSTCR is initialized to H'1F by input of

a reset signal from the $\overline{\text{RES}}$ pin, but is not initialized by the internal reset signal generated by overflow of the WDT. It is initialized to H'1F in standby mode, when the clock frequency is changed, and in clock pause mode.

Bit 7—Watchdog Timer Overflow Flag (WOVF): Indicates that WTCNT has overflowed (from H'FF to H'00) in watchdog timer mode. It is not set in interval timer mode.

| Bit 7: WOVF | Description | |
|---|---|---|
| 0 | No WTCNT overflow in watchdog timer mode | (Initial value) |
| | Cleared by reading WOVF, then writing 0 in WOVF | |
| 1 | Set by WTCNT overflow in watchdog timer mode | |

Bit 6—Reset Enable (RSTE): Selects whether to reset the chip internally if WTCNT overflows in watchdog timer mode.

| Bit 6: RSTE | Description | |
|---|---|---|
| 0 | Not reset when WTCNT overflows | (Initial value) |
| | LSI not reset internally, but WTCNT and WTCSR reset within WDT | |
| 1 | Reset when WTCNT overflows | |

Bit 5—Reset Select (RSTS): Selects the type of internal reset generated if WTCNT overflows in watchdog timer mode.

| Bit 5: RSTS | Description | |
|---|---|---|
| 0 | Power-on reset | (Initial value) |
| 1 | Manual reset | |

Bits 4 to 0—Reserved: These bits always read as 1. The write value should always be 1.

### 12.2.4 Notes on Register Access

The watchdog timer's WTCNT, WTCSR, and RSTCSR registers differ from other registers in that they are more difficult to write. The procedures for writing and reading these registers are given below.

**Writing to WTCNT and WTCSR:** These registers must be written by a word transfer instruction. They cannot be written by byte or longword transfer instructions. WTCNT and WTCSR both have the same write address. The write data must be contained in the lower byte of the written word. The upper byte must be H'5A (for WTCNT) or H'A5 (for WTCSR) (figure 12.2). This transfers the write data from the lower byte to WTCNT or WTCSR.

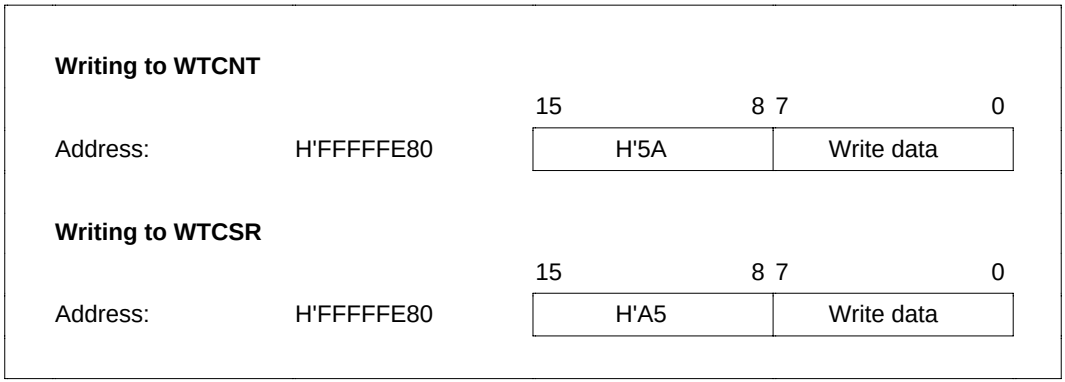



**Figure 12.2 Writing to WTCNT and WTCSR**

**Writing to RSTCSR:** RSTCSR must be written by a word access to address H'FFFFFE82. It cannot be written by byte or longword transfer instructions. Procedures for writing 0 in WOVF (bit 7) and for writing to RSTE (bit 6) and RSTS (bit 5) are different, as shown in figure 12.3. To write 0 in the WOVF bit, the write data must be H'A5 in the upper byte and H'00 in the lower byte. This clears the WOVF bit to 0. The RSTE and RSTS bits are not affected. To write to the RSTE and RSTS bits, the upper byte must be H'5A and the lower byte must be the write data. The values of bits 6 and 5 of the lower byte are transferred to the RSTE and RSTS bits, respectively. The WOVF bit is not affected.

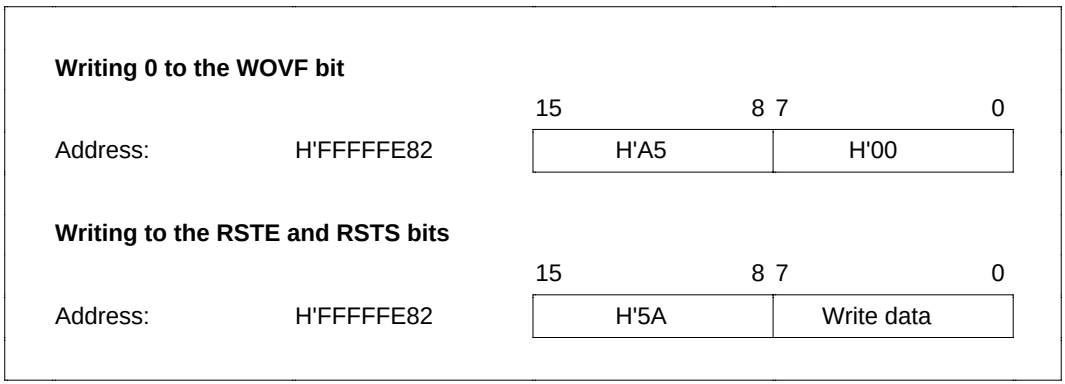



**Figure 12.3 Writing to RSTCSR**

**Reading from WTCNT, WTCSR, and RSTCSR:** WTCNT, WTCSR, and RSTCSR are read like other registers. Use byte transfer instructions. The read addresses are H'FFFFFE80 for WTCSR, H'FFFFFE81 for WTCNT, and H'FFFFFE83 for RSTCSR.

## 12.3 Operation

### 12.3.1 Operation in Watchdog Timer Mode

To use the WDT as a watchdog timer, set the WT/$\overline{IT}$ and TME bits in WTCSR to 1. Software must prevent WTCNT overflow by rewriting the WTCNT value (normally by writing H'00) before overflow occurs. Thus, WTCNT will not overflow while the system is operating normally, but if WTCNT fails to be rewritten and overflows occur due to a system crash or the like, a $\overline{WDTOVF}$ signal is output (figure 12.4). The $\overline{WDTOVF}$ signal can be used to reset the system. The $\overline{WDTOVF}$ signal is output for 512 $\phi$ clock cycles.

If the RSTE bit in RSTCSR is set to 1, a signal to reset the chip will be generated internally simultaneously with the $\overline{WDTOVF}$ signal when WTCNT overflows. Either a power-on reset or a manual reset can be selected by the RSTS bit. The internal reset signal is output for 2048 $\phi$ clock cycles.

If a reset due to the input signal from the $\overline{RES}$ pin and a reset due to WDT overflow occur simultaneously, the $\overline{RES}$ reset takes priority and the WOVF bit in RSTCSR is cleared to 0.

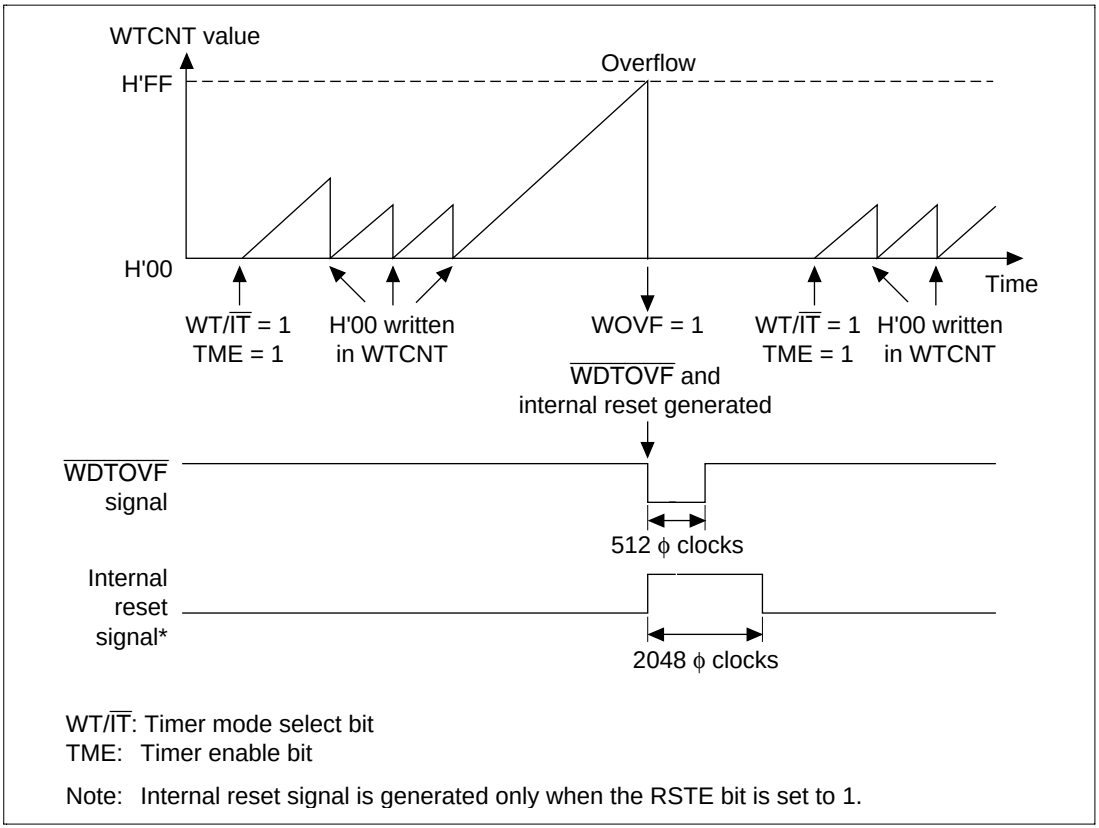



**Figure 12.4 Operation in Watchdog Timer Mode**

### 12.3.2 Operation in Interval Timer Mode

To use the WDT as an interval timer, clear WT/$\overline{IT}$ to 0 and set TME to 1 in WTCSR. An interval timer interrupt (ITI) is generated each time the watchdog timer counter (WTCNT) overflows. This function can be used to generate interval timer interrupts at regular intervals (figure 12.5).

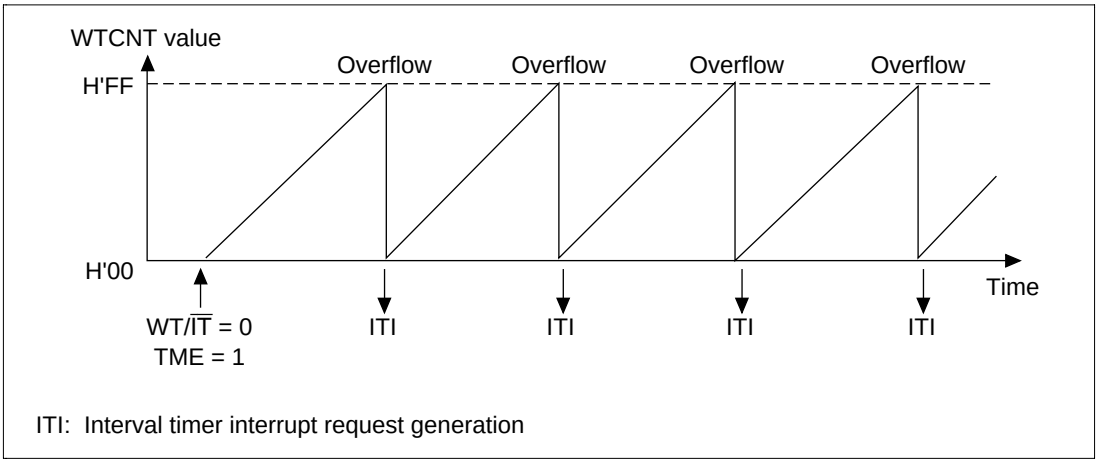



**Figure 12.5 Operation in Interval Timer Mode**

### 12.3.3 Operation when Standby Mode is Cleared

The watchdog timer has a special function to clear standby mode with an NMI interrupt. When using standby mode, set the WDT as described below.

**Transition to Standby Mode:** The TME bit in WTCSR must be cleared to 0 to stop the watchdog timer counter before it enters standby mode. The chip cannot enter standby mode while the TME bit is set to 1. Set bits CKS2 to CKS0 in WTCSR so that the counter overflow interval is equal to or longer than the oscillation settling time. See section 23, Electrical Characteristics, for the oscillation settling time.

**Recovery from Standby Mode:** When an NMI request signal is received in standby mode the clock oscillator starts running and the watchdog timer starts counting at the rate selected by bits CKS2 to CKS0 before standby mode was entered. When WTCNT overflows (changes from H'FF to H'00) the system clock ($\phi$) is presumed to be stable and usable; clock signals are supplied to the entire chip and standby mode ends.

For details on standby mode, see section 22, Power Down Modes.

### 12.3.4 Timing of Overflow Flag (OVF) Setting

In interval timer mode, when WTCNT overflows, the OVF flag in WTCSR is set to 1 and an interval timer interrupt (ITI) is requested (figure 12.6).

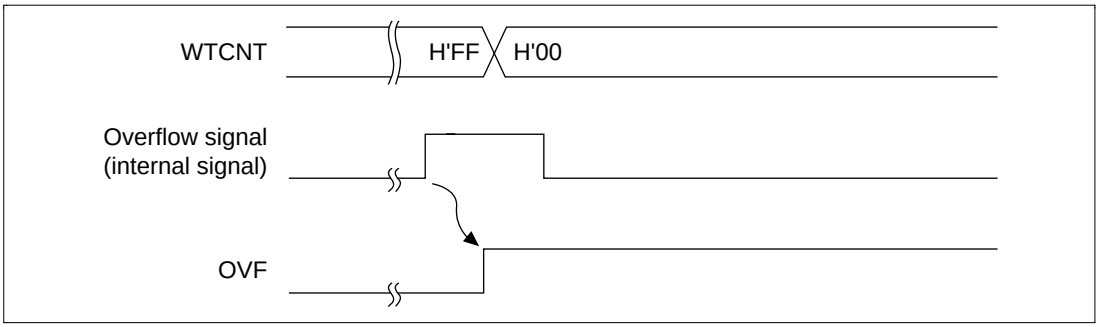



**Figure 12.6 Timing of OVF Setting**

### 12.3.5 Timing of Watchdog Timer Overflow Flag (WOVF) Setting

When WTCNT overflows the WOVF flag in RSTCSR is set to 1 and a $\overline{\text{WDTOVF}}$ signal is output. When the RSTE bit is set to 1, WTCNT overflow enables an internal reset signal to be generated for the entire chip (figure 12.7).

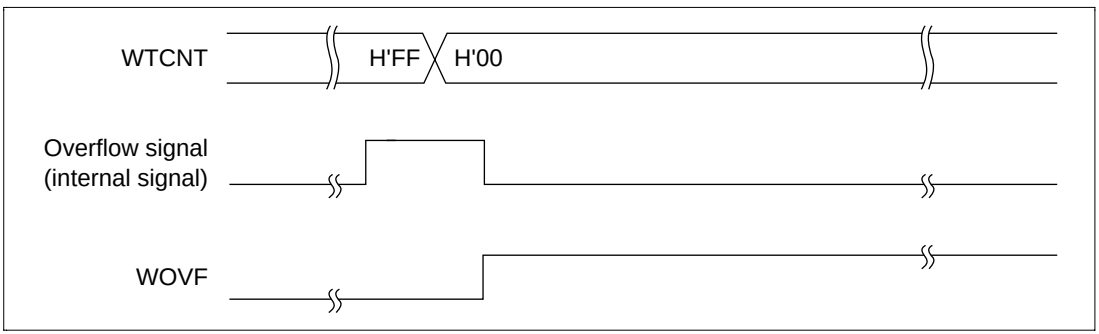



**Figure 12.7 Timing of WOVF Setting**

## 12.4 Usage Notes

### 12.4.1 Contention between WTCNT Write and Increment

If a count-up pulse is generated at the timing shown in figure 12.8 during a watchdog timer counter (WTCNT) write cycle, the write takes priority and the timer counter is not incremented (figure 12.8).

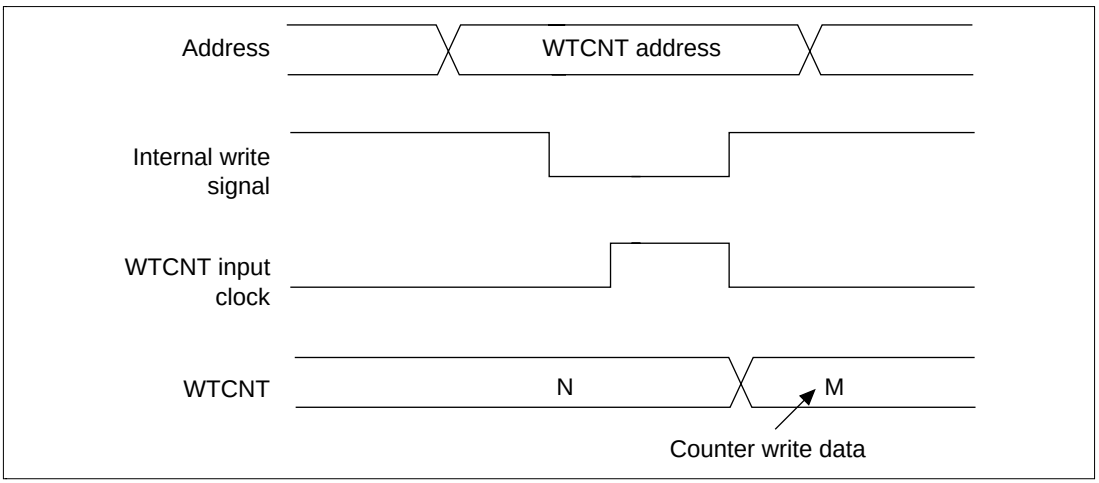



**Figure 12.8 Contention between WTCNT Write and Increment**

### 12.4.2 Changing CKS2 to CKS0 Bit Values

If the values of bits CKS2 to CKS0 are altered while the WDT is running, the count may increment incorrectly. Always stop the watchdog timer (by clearing the TME bit to 0) before changing the values of bits CKS2 to CKS0.

### 12.4.3 Switching between Watchdog Timer Mode and Interval Timer Mode

The WDT may not operate correctly if it is switched between watchdog timer mode and interval timer mode while it is running.

To ensure correct operation, always stop the watchdog timer (by clearing the TME bit to 0) before switching between watchdog timer mode and interval timer mode.

### 12.4.4 System Reset with $\overline{\text{WDTOVF}}$

If a $\overline{\text{WDTOVF}}$ signal is input to the $\overline{\text{RES}}$ pin, the device cannot initialize correctly. Avoid logical input of the $\overline{\text{WDTOVF}}$ output signal to the $\overline{\text{RES}}$ input pin. To reset the entire system with the $\overline{\text{WDTOVF}}$ signal, use the circuit shown in figure 12.9.

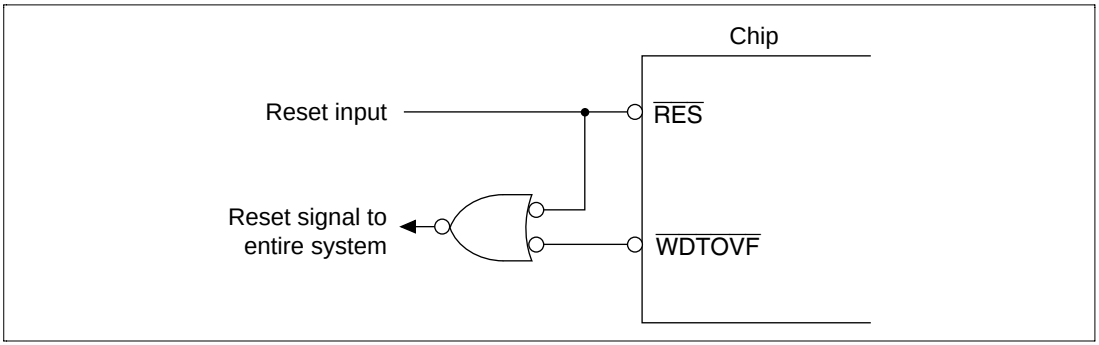



**Figure 12.9 Example of Circuit for System Reset with $\overline{\text{WDTOVF}}$ Signal**

### 12.4.5 Internal Reset in Watchdog Timer Mode

If the RSTE bit is cleared to 0 in watchdog timer mode, the chip will not reset internally when a WTCNT overflow occurs, but WTCNT and WTCSR in the WDT will reset.

# Section 13 Serial Communication Interface (SCI)

## 13.1 Overview

An on-chip serial communication interface (SCI) is provided that supports both asynchronous and clocked synchronous serial communication. It also has a multiprocessor communication function for serial communication among two or more processors.

A smart card interface is supported. This comprises smart card interface serial communication functions conforming to ISO/IEC 7816-3 (Identification Card). For details, see section 14, Smart Card Interface.

### 13.1.1 Features

The SCI has the following features.

- Choice of asynchronous or clock synchronous as the serial communication mode

  Asynchronous mode:
  - Serial data communication is synchronized by the start-stop method in character units. The SCI can communicate with a Universal Asynchronous Receiver/Transmitter (UART), an Asynchronous Communication Interface Adapter (ACIA), or any other chip that employs a standard asynchronous serial communication. (It can also communicate with two or more other processors using the multiprocessor communication function.) There are twelve selectable serial data transfer formats.
  - Data length: seven or eight bits
  - Stop bit length: one or two bits
  - Parity: even, odd, or none
  - Multiprocessor bit: one or none
  - Receive error detection: parity, overrun, and framing errors

  Clocked synchronous mode:
  - Serial data communication is synchronized with a clock signal. The SCI can communicate with other chips having a clocked synchronous communication function. There is one serial data transfer format.
  - Data length: eight bits
  - Receive error detection: overrun errors
- Full-duplex communication: Transmission and reception are independent, so the SCI can transmit and receive simultaneously. Double buffering permits continuous data transfer in both the transmit and receive directions.

- On-chip baud-rate generator: Software selectable bit rates
- Internal or external transmit/receive clock source. Baud rate generator (internal) or SCK pin (external)
- Four types of interrupts. Transmit-data-empty, transmit-end, receive-data-full, and receive-error interrupts are requested independently.

### 13.1.2 Block Diagram

Figure 13.1 shows a block diagram of the SCI.

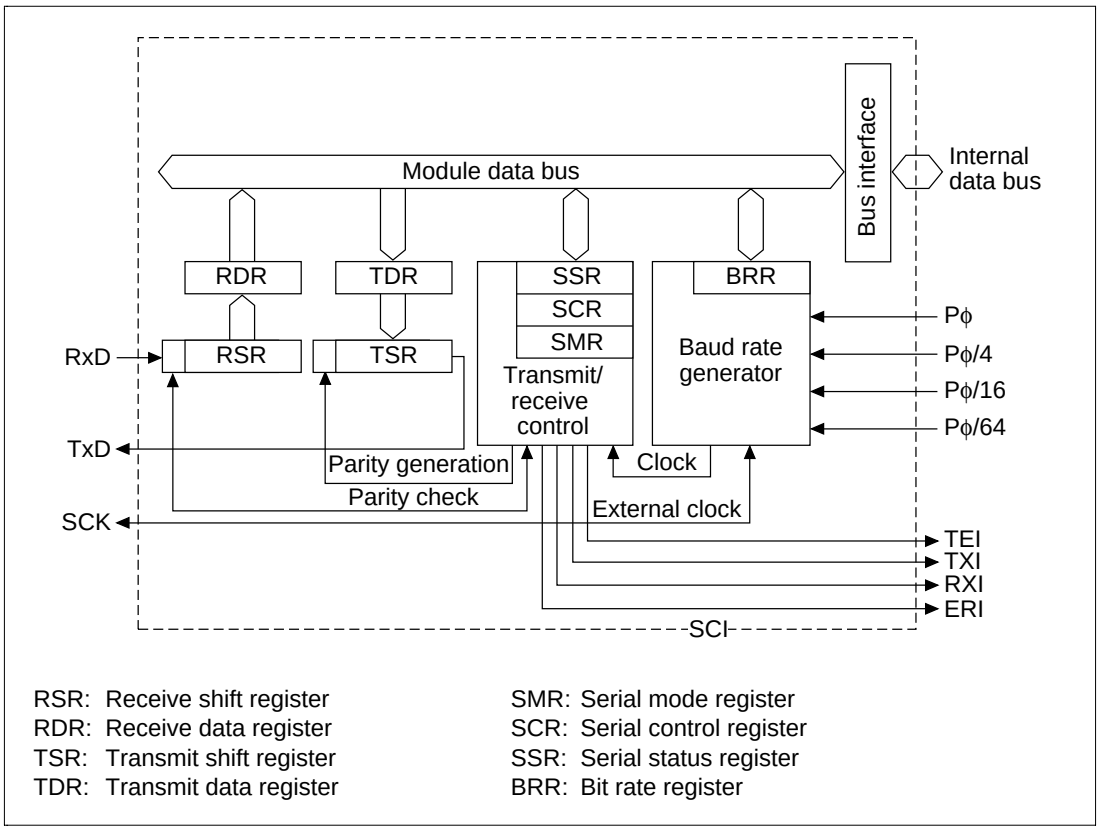



**Figure 13.1 SCI Block Diagram**

### 13.1.3 Pin Configuration

Table 13.1 shows the SCI pins configuration.

**Table 13.1 Pin Configuration**

| Pin Name | Abbreviation | Input/Output | Function |
|---|---|---|---|
| Serial clock pin | SCK | Input/output | Clock input/output |
| Receive data pin | RxD | Input | Receive data input |
| Transmit data pin | TxD | Output | Transmit data output |

### 13.1.4 Register Configuration

Table 13.2 summarizes the SCI internal registers. These registers select the communication mode (asynchronous or clock synchronous), specify the data format and bit rate, and control the transmitter and receiver sections.

**Table 13.2 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address | Access size |
|---|---|---|---|---|---|
| Serial mode register | SMR | R/W | H'00 | H'FFFFFE00 | 8 |
| Bit rate register | BRR | R/W | H'FF | H'FFFFFE01 | 8 |
| Serial control register | SCR | R/W | H'00 | H'FFFFFE02 | 8 |
| Transmit data register | TDR | R/W | H'FF | H'FFFFFE03 | 8 |
| Serial status register | SSR | R/(W)* | H'84 | H'FFFFFE04 | 8 |
| Receive data register | RDR | R | H'00 | H'FFFFFE05 | 8 |

Note: * The only value that can be written is a 0 to clear the flags.

## 13.2 Register Descriptions

### 13.2.1 Receive Shift Register (RSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | — | — | — | — | — | — | — | — |

The receive shift register (RSR) is an 8-bit register that receives serial data. Data input at the RxD pin is loaded into RSR in the order received, LSB (bit 0) first, converting the data to parallel form. When one byte has been received, it is automatically transferred to RDR. The CPU cannot read or write to RSR directly.

### 13.2.2 Receive Data Register (RDR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

The receive data register (RDR) is an 8-bit register that stores serial receive data. The SCI completes the reception of one byte of serial data by moving the received data from the receive shift register (RSR) into RDR for storage. RSR is then ready to receive the next data. This double buffering allows the SCI to receive data continuously.

The CPU can read but not write to RDR. RDR is initialized to H'00 by a reset and in standby and module standby mode.

### 13.2.3 Transmit Shift Register (TSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | — | — | — | — | — | — | — | — |

The transmit shift register (TSR) is an 8-bit register that transmits serial data. The SCI loads transmit data from the transmit data register (TDR) into TSR, then transmits the data serially from the TxD pin, LSB (bit 0) first. After transmitting one data byte, the SCI automatically loads the next transmit data from TDR into TSR and starts transmitting again. If the TDRE bit in SSR is 1, however, the SCI does not load the TDR contents into TSR. The CPU cannot read or write to TSR directly.

### 13.2.4 Transmit Data Register (TDR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The transmit data register (TDR) is an 8-bit register that stores data for serial transmission. When the SCI detects that the transmit shift register (TSR) is empty, it moves transmit data written in TDR into TSR and starts serial transmission. Continuous serial transmission is possible by writing the next transmit data in TDR during serial transmission from TSR.

The CPU can always read and write to TDR. TDR is initialized to H'FF by a reset and in standby and module standby mode.

### 13.2.5 Serial Mode Register (SMR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | MP | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The serial mode register (SMR) is an 8-bit register that specifies the SCI serial transfer format and selects the clock source for the baud rate generator.

The CPU can always read and write to SMR. SMR is initialized to H'00 by a reset and in standby and module standby mode.

Bit 7—Communication Mode (C/$\overline{A}$): Selects whether the SCI operates in asynchronous or clocked synchronous mode.

| Bit 7: C/$\overline{A}$ | Description | |
|---|---|---|
| 0 | Asynchronous mode | (Initial value) |
| 1 | Clocked synchronous mode | |

Bit 6—Character Length (CHR): Selects 7-bit or 8-bit data in asynchronous mode. In clocked synchronous mode, the data length is always eight bits, regardless of the CHR setting.

| Bit 6: CHR | Description | |
|---|---|---|
| 0 | 8-bit data | (Initial value) |
| 1 | 7-bit data<br>When 7-bit data is selected, the MSB (bit 7) of the transmit data register is not transmitted | |

Bit 5—Parity Enable (PE): Selects whether to add a parity bit to transmit data and to check the parity of receive data, in asynchronous mode. In clocked synchronous mode, a parity bit is neither added nor checked, regardless of the PE setting.

| Bit 5: PE | Description | |
|---|---|---|
| 0 | Parity bit not added or checked | (Initial value) |
| 1 | Parity bit added and checked<br>When PE is set to 1, an even or odd parity bit is added to transmit data, depending on the parity mode (O/$\overline{E}$) setting. Receive data parity is checked according to the even/odd (O/$\overline{E}$) mode setting | |

Bit 4—Parity Mode (O/$\overline{E}$): Selects even or odd parity when parity bits are added and checked. The O/$\overline{E}$ setting is used only in asynchronous mode and only when the parity enable bit (PE) is set to 1 to enable parity addition and checking. The O/$\overline{E}$ setting is ignored in clocked synchronous mode, and in asynchronous mode when parity addition and checking is disabled.

| Bit 4: O/$\overline{E}$ | Description | |
|---|---|---|
| 0 | Even parity<br>If even parity is selected, the parity bit is added to transmit data to make an even number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an even number of 1s in the received character and parity bit combined | (Initial value) |
| 1 | Odd parity<br>If odd parity is selected, the parity bit is added to transmit data to make an odd number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an odd number of 1s in the received character and parity bit combined | |

Bit 3—Stop Bit Length (STOP): Selects one or two bits as the stop bit length in asynchronous mode. This setting is used only in asynchronous mode. It is ignored in clocked synchronous mode because no stop bits are added.

| Bit 3: STOP | Description | |
|---|---|---|
| 0 | One stop bit | (Initial value) |
| | In transmitting, a single 1-bit is added at the end of each transmitted character | |
| 1 | Two stop bits | |
| | In transmitting, two 1-bits are added at the end of each transmitted character | |

In receiving, only the first stop bit is checked, regardless of the STOP bit setting. If the second stop bit is 1, it is treated as a stop bit, but if the second stop bit is 0, it is treated as the start bit of the next incoming character.

Bit 2—Multiprocessor Mode (MP): Selects multiprocessor format. When multiprocessor format is selected, settings of the parity enable (PE) and parity mode (O/$\overline{E}$) bits are ignored. The MP bit setting is used only in asynchronous mode; it is ignored in clocked synchronous mode. For the multiprocessor communication function, see section 13.3.3, Multiprocessor Communication.

| Bit 2: MP | Description | |
|---|---|---|
| 0 | Multiprocessor function disabled | (Initial value) |
| 1 | Multiprocessor format selected | |

Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source of the built-in baud rate generator. Four clock sources are available. P$\phi$, P$\phi$/4, P$\phi$/16 and P$\phi$/64. For further information on the clock source, bit rate register settings, and baud rate, see section 13.2.8, Bit Rate Register.

| Bit 1: CKS1 | Bit 0: CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | P$\phi$ | (Initial value) |
| | 1 | P$\phi$/4 | |
| 1 | 0 | P$\phi$/16 | |
| | 1 | P$\phi$/64 | |

### 13.2.6 Serial Control Register (SCR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The serial control register (SCR) operates the SCI transmitter/receiver, selects the serial clock output in asynchronous mode, enables/disables interrupts, and selects the transmit/receive clock source. The CPU can always read and write to SCR. SCR is initialized to H'00 by a reset and in standby and module standby modes.

Bit 7—Transmit Interrupt Enable (TIE): Enables or disables the transmit-data-empty interrupt (TXI) requested when the transmit data register empty bit (TDRE) in the serial status register (SSR) is set to 1 due to transfer of serial transmit data from TDR to TSR.

| Bit 7: TIE | Description |
|---|---|
| 0 | Transmit-data-empty interrupt request (TXI) is disabled (Initial value)<br>The TXI interrupt request can be cleared by reading TDRE after it has been set to 1, then clearing TDRE to 0, or by clearing TIE to 0 |
| 1 | Transmit-data-empty interrupt request (TXI) is enabled |

Bit 6—Receive Interrupt Enable (RIE): Enables or disables the receive-data-full interrupt (RXI) requested when the receive data register full bit (RDRF) in the serial status register (SSR) is set to 1 due to transfer of serial receive data from RSR to RDR. It also enables or disables receive-error interrupt (ERI) requests.

| Bit 6: RIE | Description |
|---|---|
| 0 | Receive-data-full interrupt (RXI) and receive-error interrupt (ERI) requests are disabled (Initial value)<br>RXI and ERI interrupt requests can be cleared by reading the RDRF flag or error flag (FER, PER, or ORER) after it has been set to 1, then clearing the flag to 0, or by clearing RIE to 0 |
| 1 | Receive-data-full interrupt (RXI) and receive-error interrupt (ERI) requests are enabled |

Bit 5—Transmit Enable (TE): Enables or disables the start of SCI serial transmitting operations.

| Bit 5: TE | Description |
|---|---|
| 0 | Transmitter disabled (Initial value)<br>The transmit data register empty bit (TDRE) in the serial status register (SSR) is locked at 1 |
| 1 | Transmitter enabled<br>Serial transmission starts when the transmit data register empty (TDRE) bit in the serial status register (SSR) is cleared to 0 after writing of transmit data into TDR. Select the transmit format in SMR before setting TE to 1 |

Bit 4—Receive Enable (RE): Enables or disables the SCI serial receiver.

| Bit 4: RE | Description |
|---|---|
| 0 | Receiver disabled (Initial value)<br>Clearing RE to 0 does not affect the receive flags (RDRF, FER, PER, ORER). These flags retain their previous values |
| 1 | Receiver enabled<br>Serial reception starts when a start bit is detected in asynchronous mode, or synchronous clock input is detected in clocked synchronous mode. Select the receive format in SMR before setting RE to 1 |

Bit 3—Multiprocessor Interrupt Enable (MPIE): Enables or disables multiprocessor interrupts. The MPIE setting is used only in asynchronous mode, and only if the multiprocessor mode bit (MP) in the serial mode register (SMR) is set to 1 during reception. The MPIE setting is ignored in clocked synchronous mode or when the MP bit is cleared to 0.

| Bit 3: MPIE | Description |
|---|---|
| 0 | Multiprocessor interrupts are disabled (normal receive operation) (Initial value)<br>MPE is cleared to 0 when MPIE is cleared to 0, or the multiprocessor bit (MPB) is set to 1 in receive data |
| 1 | Multiprocessor interrupts are enabled<br>Receive-data-full interrupt requests (RXI), receive-error interrupt requests (ERI), and setting of the RDRF, FER, and ORER status flags in the serial status register (SSR) are disabled until the multiprocessor bit is set to 1<br>The SCI does not transfer receive data from RSR to RDR, does not detect receive errors, and does not set the RDRF, FER, and ORER flags in the serial status register (SSR). When it receives data that includes MPB = 1, MPB is set to 1 in SSR, and the SCI automatically clears MPIE to 0, generates RXI and ERI interrupts (if the TIE and RIE bits in SCR are set to 1), and enables the FER and ORER bits to be set |

Bit 2—Transmit-End Interrupt Enable (TEIE): Enables or disables the transmit-end interrupt (TEI) requested if TDR does not contain new valid transmit data when the MSB is transmitted.

| Bit 2: TEIE | Description |
|---|---|
| 0 | Transmit-end interrupt (TEI) requests are disabled* (Initial value) |
| 1 | Transmit-end interrupt (TEI) requests are enabled* |

Note: * The TEI request can be cleared by reading the TDRE bit in the serial status register (SSR) after it has been set to 1, then clearing TDRE to 0; by clearing the transmit end (TEND) bit to 0; or by clearing the TEIE bit to 0.

Bits 1 and 0—Clock Enable 1 and 0 (CKE1 and CKE0): These bits select the SCI clock source and enable or disable clock output from the SCK pin. Depending on the combination of CKE1 and CKE0, the SCK pin can be used for general-purpose input/output, serial clock output, or serial clock input.

The CKE0 setting is valid only in asynchronous mode, and only when the SCI is internally clocked (CKE1 = 0). The CKE0 setting is ignored in clocked synchronous mode, or when an external clock source is selected (CKE1 = 1). Select the SCI operating mode in the serial mode register (SMR) before setting CKE1 and CKE0. For further details on selection of the SCI clock source, see table 13.10 in section 13.3, Operation.

| Bit 1: CKE1 | Bit 0: CKE0 | Description | |
|---|---|---|---|
| 0 | 0 | Asynchronous mode | Internal clock, SCK pin used for input pin (input signal is ignored)*1 |
| | | Clocked synchronous mode | Internal clock, SCK pin used for synchronous clock output*1 |
| 0 | 1 | Asynchronous mode | Internal clock, SCK pin used for clock output*2 |
| | | Clocked synchronous mode | Internal clock, SCK pin used for synchronous clock output |
| 1 | 0 | Asynchronous mode | External clock, SCK pin used for clock input*3 |
| | | Clocked synchronous mode | External clock, SCK pin used for synchronous clock input |
| 1 | 1 | Asynchronous mode | External clock, SCK pin used for clock input*3 |
| | | Clocked synchronous mode | External clock, SCK pin used for synchronous clock input |

Notes: 1. Initial value
2. The output clock frequency is the same as the bit rate.
3. The input clock frequency is 16 times the bit rate.

### 13.2.7 Serial Status Register (SSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TDRE | RDRF | ORER | FER | PER | TEND | MPB | MPBT |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R | R | R/W |

Note: * The only value that can be written is a 0 to clear the flag.

The serial status register (SSR) is an 8-bit register containing multiprocessor bit values, and status flags that indicate the SCI operating status.

The CPU can always read and write to SSR, but cannot write 1 in the status flags (TDRE, RDRF, ORER, PER, and FER). These flags can be cleared to 0 only if they have first been read (after being set to 1). Bits 2 (TEND) and 1 (MPB) are read-only bits that cannot be written. SSR is initialized to H'84 by a reset and in standby and module standby mode.

Bit 7—Transmit Data Register Empty (TDRE): Indicates that the SCI has loaded transmit data from TDR into TSR and new serial transmit data can be written in TDR.

| Bit 7: TDRE | Description |
|---|---|
| 0 | TDR contains valid transmit data<br>TDRE is cleared to 0 when software reads TDRE after it has been set to 1, then writes 0 in TDRE |
| 1 | TDR does not contain valid transmit data (Initial value)<br>TDRE is set to 1 when the chip is reset or enters standby mode, the TE bit in the serial control register (SCR) is cleared to 0, or TDR contents are loaded into TSR, so new data can be written in TDR |

Bit 6—Receive Data Register Full (RDRF): Indicates that RDR contains received data.

| Bit 6: RDRF | Description |
|---|---|
| 0 | RDR does not contain valid receive data (Initial value)<br>RDRF is cleared to 0 when the chip is reset or enters standby mode, or software reads RDRF after it has been set to 1, then writes 0 in RDRF |
| 1 | RDR contains valid received data<br>RDRF is set to 1 when serial data is received normally and transferred from RSR to RDR |

Note: RDR and RDRF are not affected by detection of receive errors or by clearing of the RE bit to 0 in the serial control register. They retain their previous contents. If RDRF is still set to 1 when reception of the next data ends, an overrun error (ORER) occurs and the received data is lost.

Bit 5—Overrun Error (ORER): Indicates that data reception ended abnormally due to an overrun error.

| Bit 5: ORER | Description |
|---|---|
| 0 | Receiving is in progress or has ended normally (Initial value)<br>Clearing the RE bit to 0 in the serial control register does not affect the ORER bit, which retains its previous value<br>ORER is cleared to 0 when the chip is reset or enters standby mode, or software reads ORER after it has been set to 1, then writes 0 in ORER |
| 1 | A receive overrun error occurred<br>RDR continues to hold the data received before the overrun error, so subsequent receive data is lost. Serial receiving cannot continue while ORER is set to 1. In clocked synchronous mode, serial transmitting is disabled<br>ORER is set to 1 if reception of the next serial data ends when RDRF is set to 1 |

Bit 4—Framing Error (FER): Indicates that data reception ended abnormally due to a framing error in asynchronous mode.

| Bit 4: FER | Description |
|---|---|
| 0 | Receiving is in progress or has ended normally (Initial value)<br>Clearing the RE bit to 0 in the serial control register does not affect the FER bit, which retains its previous value<br>FER is cleared to 0 when the chip is reset or enters standby mode, or software reads FER after it has been set to 1, then writes 0 in FER |
| 1 | A receive framing error occurred<br>When the stop bit length is two bits, only the first bit is checked. The second stop bit is not checked. When a framing error occurs, the SCI transfers the receive data into RDR but does not set RDRF. Serial receiving cannot continue while FER is set to 1. In clocked synchronous mode, serial transmitting is also disabled<br>FER is set to 1 if the stop bit at the end of receive data is checked and found to be 0 |

Bit 3—Parity Error (PER): Indicates that data reception (with parity) ended abnormally due to a parity error in asynchronous mode.

| Bit 3: PER | Description |
|---|---|
| 0 | Receiving is in progress or has ended normally (Initial value)<br>Clearing the RE bit to 0 in the serial control register does not affect the PER bit, which retains its previous value<br>PER is cleared to 0 when the chip is reset or enters standby mode, or software reads PER after it has been set to 1, then writes 0 in PER |
| 1 | A receive parity error occurred<br>When a parity error occurs, the SCI transfers the receive data into RDR but does not set RDRF. Serial receiving cannot continue while PER is set to 1. In clocked synchronous mode, serial transmitting is also disabled<br>PER is set to 1 if the number of 1s in receive data, including the parity bit, does not match the even or odd parity setting of the parity mode bit (O/$\overline{E}$) in the serial mode register (SMR) |

Bit 2—Transmit End (TEND): Indicates that when the last bit of a serial character was transmitted, TDR did not contain valid data, so transmission has ended. TEND is a read-only bit and cannot be written.

| Bit 2: TEND | Description |
|---|---|
| 0 | Transmission is in progress<br>TEND is cleared to 0 when software reads TDRE after it has been set to 1, then writes 0 in TDRE |
| 1 | End of transmission (Initial value)<br>TEND is set to 1 when the chip is reset or enters standby mode, TE is cleared to 0 in the serial control register (SCR), or TDRE is 1 when the last bit of a one-byte serial character is transmitted |

Bit 1—Multiprocessor Bit (MPB): Stores the value of the multiprocessor bit in receive data when a multiprocessor format is selected for receiving in asynchronous mode. MPB is a read-only bit and cannot be written.

| Bit 1: MPB | Description |
|---|---|
| 0 | Multiprocessor bit value in receive data is 0 (Initial value)<br>If RE is cleared to 0 when a multiprocessor format is selected, MPB retains its previous value |
| 1 | Multiprocessor bit value in receive data is 1 |

Bit 0—Multiprocessor Bit Transfer (MPBT): Stores the value of the multiprocessor bit added to transmit data when a multiprocessor format is selected for transmitting in asynchronous mode. The MPBT setting is ignored in clocked synchronous mode, when a multiprocessor format is not selected, or when the SCI is not transmitting.

| Bit 0: MPBT | Description |
|---|---|
| 0 | Multiprocessor bit value in transmit data is 0 (Initial value) |
| 1 | Multiprocessor bit value in transmit data is 1 |

### 13.2.8 Bit Rate Register (BRR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The bit rate register (BRR) is an 8-bit register that, together with the baud rate generator clock source selected by the CKS1 and CKS0 bits in the serial mode register (SMR), determines the serial transmit/receive bit rate.

The CPU can always read and write to BRR. BRR is initialized to H'FF by a reset, by the module standby function, and in standby mode.

Table 13.3 shows examples of BRR settings in asynchronous mode; table 13.4 shows examples of BRR settings in clocked synchronous mode.

**Table 13.3 Bit Rates and BRR Settings in Asynchronous Mode**

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

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 3.6864 | | | 4 | | | 4.9152 | | | 5 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 64 | 0.70 | 2 | 70 | 0.03 | 2 | 86 | 0.31 | 2 | 88 | –0.25 |
| 150 | 1 | 191 | 0.00 | 1 | 207 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 |
| 300 | 1 | 95 | 0.00 | 1 | 103 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 |
| 600 | 0 | 191 | 0.00 | 0 | 207 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 |
| 1200 | 0 | 95 | 0.00 | 0 | 103 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 |
| 2400 | 0 | 47 | 0.00 | 0 | 51 | 0.16 | 0 | 63 | 0.00 | 0 | 64 | 0.16 |
| 4800 | 0 | 23 | 0.00 | 0 | 25 | 0.16 | 0 | 31 | 0.00 | 0 | 32 | –1.36 |
| 9600 | 0 | 11 | 0.00 | 0 | 12 | 0.16 | 0 | 15 | 0.00 | 0 | 15 | 1.73 |
| 19200 | 0 | 5 | 0.00 | 0 | 6 | –6.99 | 0 | 7 | 0.00 | 0 | 7 | 1.73 |
| 31250 | 0 | 3 | –7.84 | 0 | 3 | 0.00 | 0 | 4 | –1.70 | 0 | 4 | 0.00 |
| 38400 | 0 | 2 | 0.00 | 0 | 2 | 8.51 | 0 | 3 | 0.00 | 0 | 3 | 1.73 |

**Table 13.3 Bit Rates and BRR Settings in Asynchronous Mode (cont)**

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | | | 6.144 | | | 7.37288 | | | 8 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 106 | –0.44 | 2 | 108 | 0.08 | 2 | 130 | –0.07 | 2 | 141 | 0.03 |
| 150 | | 2777 | 0.16 | 2 | 79 | 0.00 | 2 | 95 | 0.00 | 2 | 103 | 0.16 |
| 300 | 1 | 155 | 0.16 | 1 | 159 | 0.00 | 1 | 191 | 0.00 | 1 | 207 | 0.16 |
| 600 | 1 | 77 | 0.16 | 1 | 79 | 0.00 | 1 | 95 | 0.00 | 1 | 103 | 0.16 |
| 1200 | 0 | 155 | 0.16 | 0 | 159 | 0.00 | 0 | 191 | 0.00 | 0 | 207 | 0.16 |
| 2400 | 0 | 77 | 0.16 | 0 | 79 | 0.00 | 0 | 95 | 0.00 | 0 | 103 | 0.16 |
| 4800 | 0 | 38 | 0.16 | 0 | 39 | 0.00 | 0 | 47 | 0.00 | 0 | 51 | 0.16 |
| 9600 | 0 | 19 | –2.34 | 0 | 19 | 0.00 | 0 | 23 | 0.00 | 0 | 25 | 0.16 |
| 19200 | 0 | 9 | –2.34 | 0 | 9 | 0.00 | 0 | 11 | 0.00 | 0 | 12 | 0.16 |
| 31250 | 0 | 5 | 0.00 | 0 | 5 | 2.40 | 0 | 6 | 5.33 | 0 | 7 | 0.00 |
| 38400 | 0 | 4 | –2.34 | 0 | 4 | 0.00 | 0 | 5 | 0.00 | 0 | 6 | –6.99 |

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 9.8304 | | | 10 | | | 12 | | | 12.288 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 174 | –0.26 | 2 | 177 | –0.25 | 2 | 212 | 0.03 | 2 | 217 | 0.08 |
| 150 | 2 | 127 | 0.00 | 2 | 129 | 0.16 | 2 | 155 | 0.16 | 2 | 159 | 0.00 |
| 300 | 1 | 255 | 0.00 | 2 | 64 | 0.16 | 2 | 77 | 0.16 | 2 | 79 | 0.00 |
| 600 | 1 | 127 | 0.00 | 1 | 129 | 0.16 | 1 | 155 | 0.16 | 1 | 159 | 0.00 |
| 1200 | 0 | 255 | 0.00 | 1 | 64 | 0.16 | 1 | 77 | 0.16 | 1 | 79 | 0.00 |
| 2400 | 0 | 127 | 0.00 | 0 | 129 | 0.16 | 0 | 155 | 0.16 | 0 | 159 | 0.00 |
| 4800 | 0 | 63 | 0.00 | 0 | 64 | 0.16 | 0 | 77 | 0.16 | 0 | 79 | 0.00 |
| 9600 | 0 | 31 | 0.00 | 0 | 32 | –1.36 | 0 | 38 | 0.16 | 0 | 39 | 0.00 |
| 19200 | 0 | 15 | 0.00 | 0 | 15 | 1.73 | 0 | 19 | –2.34 | 0 | 19 | 0.00 |
| 31250 | 0 | 9 | –1.70 | 0 | 9 | 0.00 | 0 | 11 | 0.00 | 0 | 11 | 2.40 |
| 38400 | 0 | 7 | 0.00 | 0 | 7 | 1.73 | 0 | 9 | –2.34 | 0 | 9 | 0.00 |

**Table 13.3 Bit Rates and BRR Settings in Asynchronous Mode (cont)**

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14.7456 | | | 16 | | | 19.6608 | | | 20 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 64 | 0.70 | 3 | 70 | 0.03 | 3 | 86 | 0.31 | 3 | 88 | –0.25 |
| 150 | 2 | 191 | 0.00 | 2 | 207 | 0.16 | 2 | 255 | 0.00 | 3 | 64 | 0.16 |
| 300 | 2 | 95 | 0.00 | 2 | 103 | 0.16 | 2 | 127 | 0.00 | 2 | 129 | 0.16 |
| 600 | 1 | 191 | 0.00 | 1 | 207 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 |
| 1200 | 1 | 95 | 0.00 | 1 | 103 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 |
| 2400 | 0 | 191 | 0.00 | 0 | 207 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 |
| 4800 | 0 | 95 | 0.00 | 0 | 103 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 |
| 9600 | 0 | 47 | 0.00 | 0 | 51 | 0.16 | 0 | 63 | 0.00 | 0 | 64 | 0.16 |
| 19200 | 0 | 23 | 0.00 | 0 | 25 | 0.16 | 0 | 31 | 0.00 | 0 | 32 | –1.36 |
| 31250 | 0 | 14 | –1.70 | 0 | 15 | 0.00 | 0 | 19 | –1.70 | 0 | 19 | 0.00 |
| 38400 | 0 | 11 | 0.00 | 0 | 12 | 0.16 | 0 | 15 | 0.00 | 0 | 15 | 1.73 |

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 24 | | | 24.576 | | | 28.7 | | | 30 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 106 | –0.44 | 3 | 108 | 0.08 | 3 | 126 | 0.31 | 3 | 132 | 0.13 |
| 150 | 3 | 77 | 0.16 | 3 | 79 | 0.00 | 3 | 92 | 0.46 | 3 | 97 | –0.35 |
| 300 | 2 | 155 | 0.16 | 2 | 159 | 0.00 | 2 | 186 | –0.08 | 2 | 194 | 0.16 |
| 600 | 2 | 77 | 0.16 | 2 | 79 | 0.00 | 2 | 92 | 0.46 | 2 | 97 | –0.35 |
| 1200 | 1 | 155 | 0.16 | 1 | 159 | 0.00 | 1 | 186 | –0.08 | 1 | 194 | 0.16 |
| 2400 | 1 | 77 | 0.16 | 1 | 79 | 0.00 | 1 | 92 | 0.46 | 1 | 97 | –0.35 |
| 4800 | 0 | 155 | 0.16 | 0 | 159 | 0.00 | 0 | 186 | –0.08 | 0 | 194 | 0.16 |
| 9600 | 0 | 77 | 0.16 | 0 | 79 | 0.00 | 0 | 92 | 0.46 | 0 | 97 | –0.35 |
| 19200 | 0 | 38 | 0.16 | 0 | 39 | 0.00 | 0 | 46 | –0.61 | 0 | 48 | –0.35 |
| 31250 | 0 | 23 | 0.00 | 0 | 24 | –1.70 | 0 | 28 | –1.03 | 0 | 29 | 0.00 |
| 38400 | 0 | 19 | –2.34 | 0 | 19 | 0.00 | 0 | 22 | 1.55 | 0 | 23 | 1.73 |

**Table 13.4 Bit Rates and BRR Settings in Clocked Synchronous Mode**

| Bit Rate (bits/s) | Pϕ (MHz) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 4 | | 8 | | 16 | | 28.7 | | 30 | |
| | 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 | 3 | 223 | 3 | 233 |
| 1k | 1 | 249 | 2 | 124 | 2 | 249 | 3 | 111 | 3 | 116 |
| 2.5k | 1 | 99 | 1 | 199 | 2 | 99 | 2 | 178 | 2 | 187 |
| 5k | 0 | 199 | 1 | 99 | 1 | 199 | 2 | 89 | 2 | 93 |
| 10k | 0 | 99 | 0 | 199 | 1 | 99 | 1 | 178 | 1 | 187 |
| 25k | 0 | 39 | 0 | 79 | 0 | 159 | 1 | 71 | 1 | 74 |
| 50k | 0 | 19 | 0 | 39 | 0 | 79 | 0 | 143 | 0 | 149 |
| 100k | 0 | 9 | 0 | 19 | 0 | 39 | 0 | 71 | 0 | 74 |
| 250k | 0 | 3 | 0 | 7 | 0 | 15 | — | — | 0 | 29 |
| 500k | 0 | 1 | 0 | 3 | 0 | 7 | — | — | 0 | 14 |
| 1M | 0 | 0* | 0 | 1 | 0 | 3 | — | — | — | — |
| 2M | | | 0 | 0* | 0 | 1 | — | — | — | — |

Note: Settings with an error of 1% or less are recommended.

Explanation of symbols:

Blank: No setting possible

—: Setting possible, but error occurs

*: Continuous transmission/reception not possible

The BRR setting is calculated as follows:

Asynchronous mode:

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

Clocked synchronous mode:

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

B: Bit rate (bit/s)

N: BRR setting for baud rate generator ($0 \leq N \leq 255$)

Pϕ: Peripheral module clock (MHz)

n: Baud rate generator input clock source (n = 0, 1, 2, 3) (For the clock sources and values of n, see table 13.5.)

**Table 13.5 SMR Settings**

| | | SMR Settings | |
|---|---|---|---|
| **n** | **Clock Source** | **CKS1** | **CKS0** |
| 0 | Pϕ | 0 | 0 |
| 1 | Pϕ/4 | 0 | 1 |
| 2 | Pϕ/16 | 1 | 0 |
| 3 | Pϕ/64 | 1 | 1 |

The bit rate error for asynchronous mode is given by the following equation:

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

Table 13.6 shows the maximum bit rates in asynchronous mode when the baud rate generator is being used. Tables 13.7 and 13.8 show the maximum rates for external clock input.

**Table 13.6 Maximum Bit Rates for Various Frequencies with Baud Rate Generator (Asynchronous Mode)**

| | | Settings | |
|---|---|---|---|
| **Pϕ (MHz)** | **Maximum Bit Rate (bits/s)** | **n** | **N** |
| 2 | 62500 | 0 | 0 |
| 2.097152 | 65536 | 0 | 0 |
| 2.4576 | 76800 | 0 | 0 |
| 3 | 93750 | 0 | 0 |
| 3.6864 | 115200 | 0 | 0 |
| 4 | 125000 | 0 | 0 |
| 4.9152 | 153600 | 0 | 0 |
| 8 | 250000 | 0 | 0 |
| 9.8304 | 307200 | 0 | 0 |
| 12 | 375000 | 0 | 0 |
| 14.7456 | 460800 | 0 | 0 |
| 16 | 500000 | 0 | 0 |
| 19.6608 | 614400 | 0 | 0 |
| 20 | 625000 | 0 | 0 |
| 24 | 750000 | 0 | 0 |
| 24.576 | 768000 | 0 | 0 |
| 28.7 | 896875 | 0 | 0 |
| 30 | 937500 | 0 | 0 |

**Table 13.7 Maximum Bit Rates with External Clock Input (Asynchronous Mode)**

| Pϕ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bits/s) |
|---|---|---|
| 2 | 0.5000 | 31250 |
| 2.097152 | 0.5243 | 32768 |
| 2.4576 | 0.6144 | 38400 |
| 3 | 0.7500 | 46875 |
| 3.6864 | 0.9216 | 57600 |
| 4 | 1.0000 | 62500 |
| 4.9152 | 1.2288 | 76800 |
| 8 | 2.0000 | 125000 |
| 9.8304 | 2.4576 | 153600 |
| 12 | 3.0000 | 187500 |
| 14.7456 | 3.6864 | 230400 |
| 16 | 4.0000 | 250000 |
| 19.6608 | 4.9152 | 307200 |
| 20 | 5.0000 | 312500 |
| 24 | 6.0000 | 375000 |
| 24.576 | 6.1440 | 384000 |
| 28.7 | 7.1750 | 448436 |
| 30 | 7.5000 | 468750 |

**Table 13.8 Maximum Bit Rates with External Clock Input (Clocked Synchronous Mode)**

| Pϕ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bits/s) |
|---|---|---|
| 8 | 1.3333 | 1333333.3 |
| 16 | 2.6667 | 2666666.7 |
| 24 | 4.0000 | 4000000.0 |
| 28.7 | 4.7833 | 4783333.3 |
| 30 | 5.0000 | 5000000.0 |

## 13.3 Operation

### 13.3.1 Overview

For serial communication, the SCI has an asynchronous mode in which characters are synchronized individually, and a clocked synchronous mode in which communication is synchronized with clock pulses. Asynchronous/clocked synchronous mode and the transfer format are selected in the serial mode register (SMR), as shown in table 13.9. The SCI clock source is selected by the C/$\overline{A}$ bit in the serial mode register (SMR) and the CKE1 and CKE0 bits in the serial control register (SCR), as shown in table 13.10.

**Asynchronous Mode:**

- Data length is selectable. seven or eight bits.
- Parity and multiprocessor bits are selectable, as is the stop bit length (one or two bits). The preceding selections constitute the transfer format and character length.
- In receiving, it is possible to detect framing errors, parity errors, overrun errors.
- An internal or external clock can be selected as the SCI clock source.
  - — When an internal clock is selected, the SCI operates using the built-in baud rate generator, and can output a serial clock signal with a frequency matching the bit rate.
  - — When an external clock is selected, the external clock input must have a frequency 16 times the bit rate. (The built-in baud rate generator is not used.)

**Clocked Synchronous Mode:**

- The transfer format has a fixed eight-bit data length.
- In receiving, it is possible to detect overrun errors.
- An internal or external clock can be selected as the SCI clock source.
  - — When an internal clock is selected, the SCI operates using the built-in baud rate generator, and outputs a synchronous clock signal to external devices.
  - — When an external clock is selected, the SCI operates on the input synchronous clock. The built-in baud rate generator is not used.

**Table 13.9 Serial Mode Register Settings and SCI Transfer Formats**

| | SMR Settings | | | | | SCI Communication Format | | | |
|---|---|---|---|---|---|---|---|---|---|
| Mode | Bit 7 C/$\overline{A}$ | Bit 6 CHR | Bit 5 PE | Bit 2 MP | Bit 3 STOP | Data Length | Parity Bit | Multipro-cessor Bit | Stop Bit Length |
| Asynchronous | 0 | 0 | 0 | 0 | 0 | 8-bit | Not set | Not set | 1 bit |
| | | | | | 1 | | | | 2 bits |
| | | | 1 | | 0 | | Set | | 1 bit |
| | | | | | 1 | | | | 2 bits |
| | | 1 | 0 | | 0 | 7-bit | Not set | | 1 bit |
| | | | | | 1 | | | | 2 bits |
| | | | 1 | | 0 | | Set | | 1 bit |
| | | | | | 1 | | | | 2 bits |
| Asynchronous (multiprocessor format) | 0 | 0 | * | 1 | 0 | 8-bit | Not set | Set | 1 bit |
| | | | * | | 1 | | | | 2 bits |
| | | 1 | * | | 0 | 7-bit | | | 1 bit |
| | | | * | | 1 | | | | 2 bits |
| Clocked synchronous | 1 | * | * | * | * | 8-bit | Not set | Not set | None |

Note: Asterisks (*) in the table indicate don't care bits.

**Table 13.10 SMR and SCR Settings and SCI Clock Source Selection**

| | SMR | SCR Settings | | SCI Transmit/Receive Clock | |
|---|---|---|---|---|---|
| Mode | Bit 7 C/$\overline{A}$ | Bit 1 CKE1 | Bit 0 CKE0 | Clock Source | SCK Pin Function |
| Asynchronous | 0 | 0 | 0 | Internal | SCI does not use the SCK pin |
| | | | 1 | | Outputs a clock with frequency matching the bit rate |
| | | 1 | 0 | External | Inputs a clock with frequency 16 times the bit rate |
| | | | 1 | | |
| Clocked synch-ronous | 1 | 0 | 0 | Internal | Outputs the synchronous clock |
| | | | 1 | | |
| | | 1 | 0 | External | Inputs the synchronous clock |
| | | | 1 | | |

### 13.3.2 Operation in Asynchronous Mode

In asynchronous mode, each transferred character begins with a start bit and ends with a stop bit. Serial communication is synchronized one character at a time.

The transmitting and receiving sections of the SCI are independent, so full duplex communication is possible. The transmitter and receiver are both double buffered, so data can be written and read while transmitting and receiving are in progress, enabling continuous transmitting and receiving.

Figure 13.2 shows the general format of asynchronous serial communication. In asynchronous serial communication, the communication line is normally held in the mark (high) state. The SCI monitors the line and starts serial communication when the line goes to the space (low) state, indicating a start bit. One serial character consists of a start bit (low), data (LSB first), parity bit (high or low), and stop bit (high), in that order.

When receiving in asynchronous mode, the SCI synchronizes on the falling edge of the start bit. The SCI samples each data bit on the eighth pulse of a clock with a frequency 16 times the bit rate. Receive data is latched at the center of each bit.

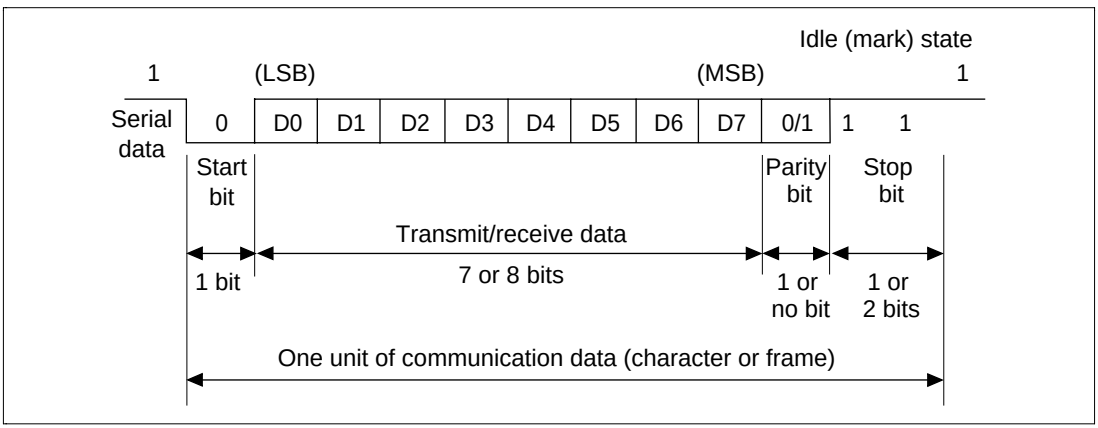



**Figure 13.2 Example of Data Format in Asynchronous Communication (8-Bit Data with Parity and Two Stop Bits)**

**Transfer Formats:** Table 13.11 shows the 12 transfer formats that can be selected in asynchronous mode. The format is selected by settings in the serial mode register (SMR).

**Table 13.11 Serial Transfer Formats (Asynchronous Mode)**

| SMR Bits | | | | Serial Transmit/Receive Format and Frame Length | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHR | PE | MP | STOP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | 0 | START | 8-bit data | | | | | | | | STOP | | |
| 0 | 0 | 0 | 1 | START | 8-bit data | | | | | | | | STOP | STOP | |
| 0 | 1 | 0 | 0 | START | 8-bit data | | | | | | | | P | STOP | |
| 0 | 1 | 0 | 1 | START | 8-bit data | | | | | | | | P | STOP | STOP |
| 1 | 0 | 0 | 0 | START | 7-bit data | | | | | | | STOP | | | |
| 1 | 0 | 0 | 1 | START | 7-bit data | | | | | | | STOP | STOP | | |
| 1 | 1 | 0 | 0 | START | 7-bit data | | | | | | | P | STOP | | |
| 1 | 1 | 0 | 1 | START | 7-bit data | | | | | | | P | STOP | STOP | |
| 0 | — | 1 | 0 | START | 8-bit data | | | | | | | | MPB | STOP | |
| 0 | — | 1 | 1 | START | 8-bit data | | | | | | | | MPB | STOP | STOP |
| 1 | — | 1 | 0 | START | 7-bit data | | | | | | | MPB | STOP | | |
| 1 | — | 1 | 1 | START | 7-bit data | | | | | | | MPB | STOP | STOP | |

Note: —: Don't care bits.
START: Start bit
STOP: Stop bit
P: Parity bit
MPB: Multiprocessor bit

**Clock:** An internal clock generated by the built-in baud rate generator or an external clock input from the SCK pin can be selected as the SCI transfer clock. The clock source is selected by the C/$\overline{A}$ bit in the serial mode register (SMR) and bits CKE1 and CKE0 in the serial control register (SCR) (table 13.10).

When an external clock is input at the SCK pin, it must have a frequency equal to 16 times the desired bit rate.

When the SCI operates on an internal clock, it can output a clock signal at the SCK pin. The frequency of this output clock is equal to the bit rate. The phase is aligned as in figure 13.3 so that the rising edge of the clock occurs at the center of each transmit data bit.

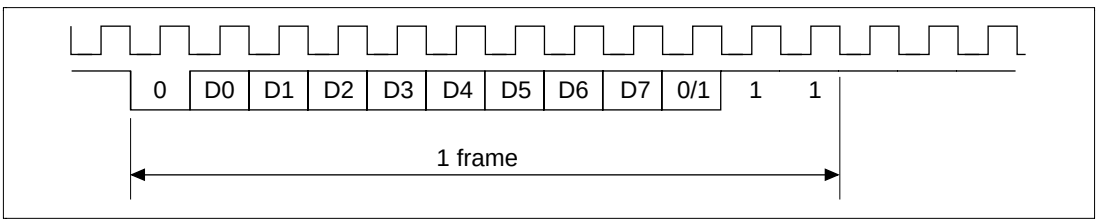



**Figure 13.3 Output Clock and Serial Data Timing (Asynchronous Mode)**

**Transferring Data**

**SCI Initialization (Asynchronous Mode):** Before transmitting or receiving, clear the TE and RE bits to 0 in the serial control register (SCR), then initialize the SCI as follows.

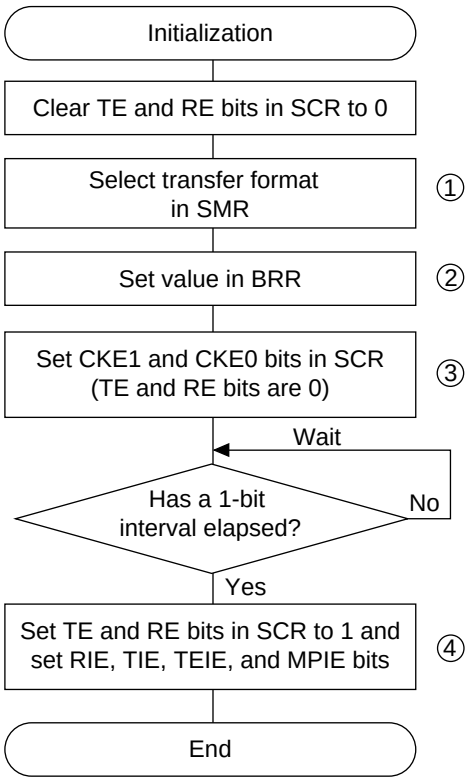
When changing the operation mode or transfer format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 sets TDRE to 1 and initializes the transmit shift register (TSR). Clearing RE to 0, however, does not initialize the RDRF, PER, FER, and ORER flags and receive data register (RDR), which retain their previous contents.

When an external clock is used, the clock should not be stopped during initialization or subsequent operation. SCI operation becomes unreliable if the clock is stopped.

Figure 13.4 shows a sample flowchart for initializing the SCI. The procedure for initializing the SCI is as follows:

1. Select the transfer format in the serial mode register (SMR).
2. Write the value corresponding to the bit rate in the bit rate register (BRR) unless an external clock is used.
3. Select the clock source in the serial control register (SCR). Leave RIE, TIE, TEIE, MPIE, TE and RE cleared to 0. If clock output is selected in asynchronous mode, clock output starts immediately after the setting is made in SCR.

4. Wait for at least the interval required to transmit or receive one bit, then set TE or RE in the serial control register (SCR) to 1. Also set RIE, TIE, TEIE and MPIE as necessary. Setting TE or RE enables the SCI to use the TxD or RxD pin. The initial states are the mark state when transmitting, and the idle state when receiving (waiting for a start bit).

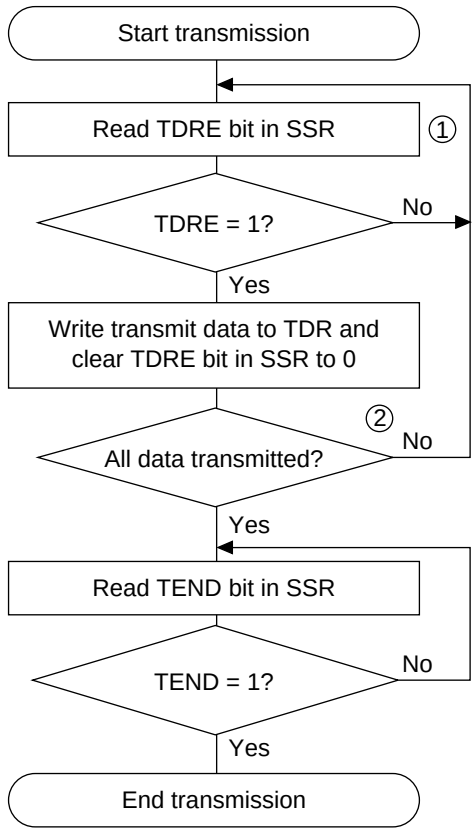



Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.4 Sample Flowchart for SCI Initialization**

**Transmitting Serial Data (Asynchronous Mode):** Figure 13.5 shows a sample flowchart for transmitting serial data. The procedure for transmitting serial data is as follows:

1. SCI status check and transmit data write: read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR) and clear TDRE to 0.
2. Serial transmission continuation procedure: to continue transmitting serial data, read 1 from the TDRE bit to confirm that writing is possible, then write data in TDR, then clear TDRE to 0.




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.5 Sample Flowchart for Transmitting Serial Data**

In transmitting serial data, the SCI operates as follows:

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0, the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from TDR into the transmit shift register (TSR).

2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) is set to 1 in SCR, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

   Serial transmit data is transmitted in the following order from the TxD pin:

   a. Start bit: one 0-bit is output.
   b. Transmit data: seven or eight bits of data are output, LSB first.
   c. Parity bit or multiprocessor bit: one parity bit (even or odd parity) or one multiprocessor bit is output. Formats in which neither a parity bit nor a multiprocessor bit is output can also be selected.
   d. Stop bit: one or two 1-bits (stop bits) are output.
   e. Mark state: output of 1-bits continues until the start bit of the next transmit data.

3. The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads new data from TDR into TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit to 1 in SSR, outputs the stop bit, then continues output of 1-bits (mark state). If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested.

Figure 13.6 shows an example of SCI transmit operation in asynchronous mode.

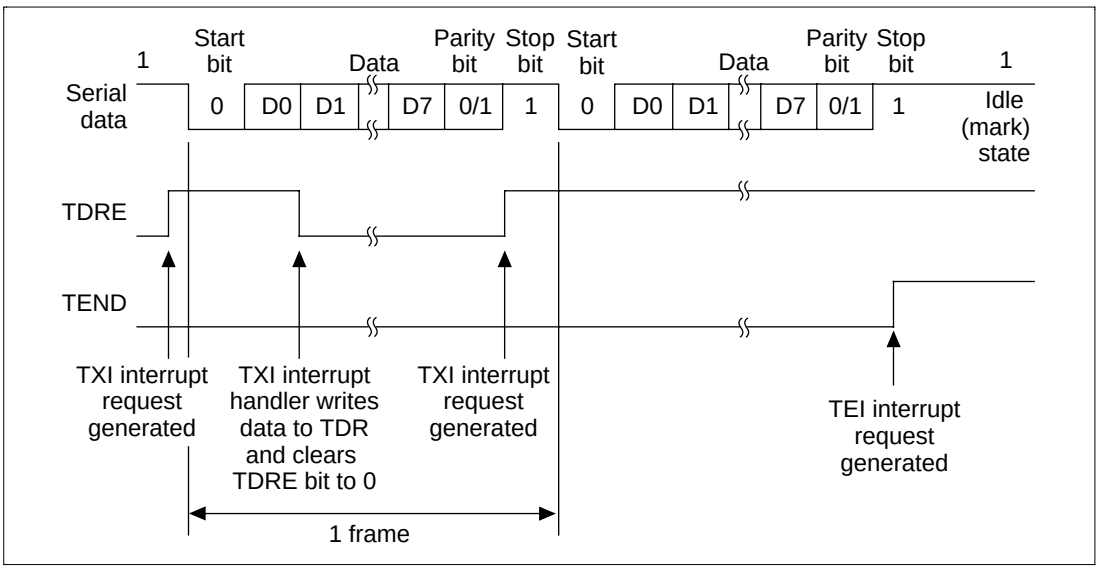
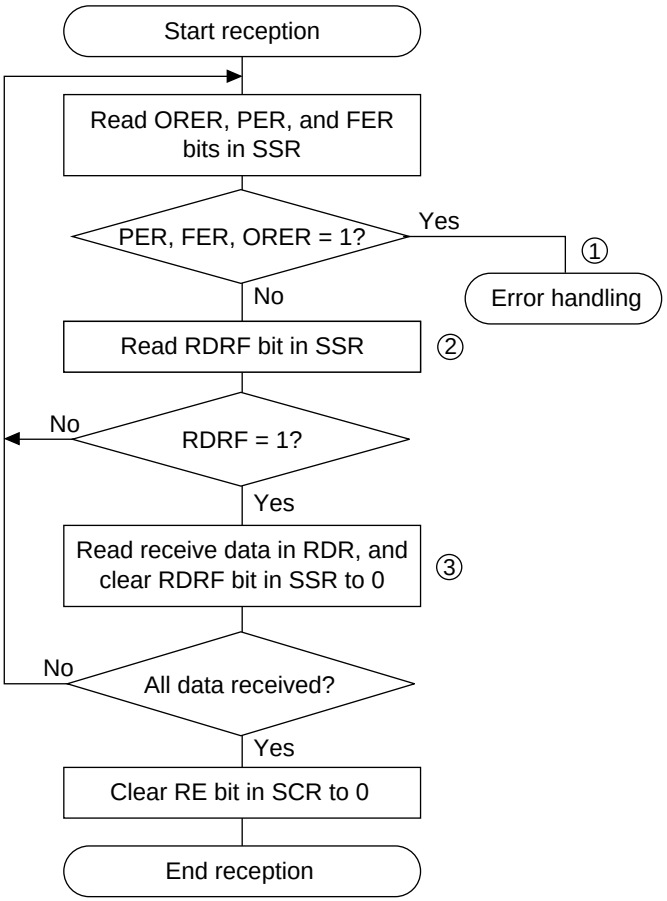



**Figure 13.6 Example of SCI Transmit Operation in Asynchronous Mode (8-Bit Data with Parity and One Stop Bit)**

**Receiving Serial Data (Asynchronous Mode):** Figure 13.7 shows a sample flowchart for receiving serial data. The procedure for receiving serial data is as follows:

1. Receive error handling: if a receive error occurs, read the ORER, PER and FER bits of the SSR to identify the error. After executing the necessary error handling, clear ORER, PER and FER all to 0. Receiving cannot resume if ORER, PER or FER remain set to 1.
2. SCI status check and receive-data read: read the serial status register (SSR), check that RDRF is set to 1, then read receive data from the receive data register (RDR) and clear RDRF to 0. The RXI interrupt can also be used to determine if the RDRF bit has changed from 0 to 1.
3. To continue receiving serial data: read the RDRF and RDR bits and clear RDRF to 0 before the stop bit of the current frame is received.




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.7 Sample Flowchart for Receiving Serial Data**

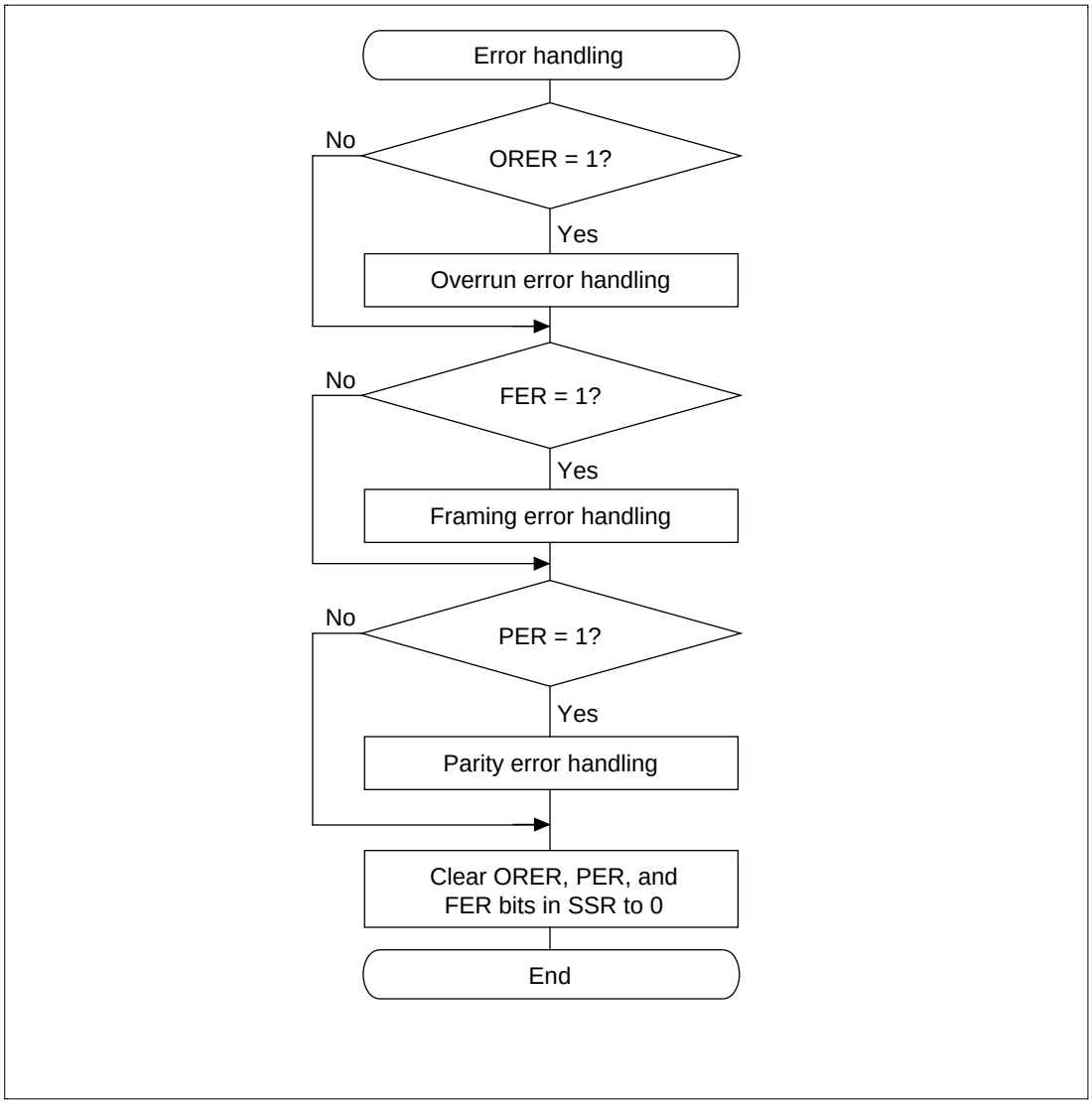



**Figure 13.7 Sample Flowchart for Receiving Serial Data (cont)**

In receiving, the SCI operates as follows:

1. The SCI monitors the receive data line. When it detects a start bit (0), the SCI synchronizes internally and starts receiving.
2. Receive data is shifted into RSR in order from LSB to MSB.
3. The parity bit and stop bit are received. After receiving these bits, the SCI makes the following checks:
   a. Parity check: the number of 1s in the receive data must match the even or odd parity setting of the O/$\overline{E}$ bit in SMR.
   b. Stop bit check: the stop bit value must be 1. If there are two stop bits, only the first stop bit is checked.
   c. Status check: RDRF must be 0 so that receive data can be loaded from RSR into RDR.

   If these checks all pass, the SCI sets RDRF to 1 and stores the received data in RDR. If one of the checks fails (receive error), the SCI operates as indicated in table 13.11.

   Note: When a receive error flag is set, further receiving is disabled. In reception, the RDRF bit is not set to 1. Be sure to clear the error flags.

4. After setting RDRF to 1, if the receive-data-full interrupt enable bit (RIE) is set to 1 in SCR, the SCI requests a receive-data-full interrupt (RXI). If one of the error flags (ORER, PER, or FER) is set to 1 and the receive-data-full interrupt enable bit (RIE) in SCR is also set to 1, the SCI requests a receive-error interrupt (ERI).

Figure 13.8 shows an example of SCI receive operation in asynchronous mode.

**Table 13.12 Receive Error Conditions and SCI Operation**

| Receive Error | Abbreviation | Condition | Data Transfer |
|---|---|---|---|
| Overrun error | ORER | Receiving of next data ends while RDRF is still set to 1 in SSR | Receive data not loaded from RSR into RDR |
| Framing error | FER | Stop bit is 0 | Receive data loaded from RSR into RDR |
| Parity error | PER | Parity of receive data differs from even/odd parity setting in SMR | Receive data loaded from RSR into RDR |

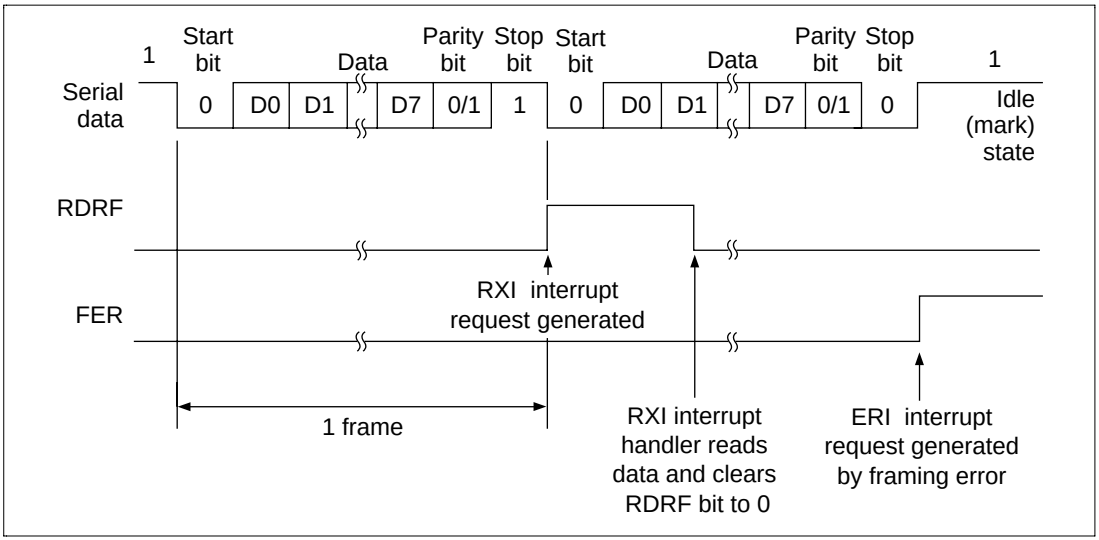



**Figure 13.8 Example of SCI Receive Operation (8-Bit Data with Parity and One Stop Bit)**

### 13.3.3 Multiprocessor Communication

The multiprocessor communication function enables several processors to share a single serial communication line. The processors communicate in asynchronous mode using a format with an additional multiprocessor bit (multiprocessor format).

In multiprocessor communication, each receiving processor is addressed by a unique ID. A serial communication cycle consists of an ID-sending cycle that identifies the receiving processor, and a data-sending cycle. The multiprocessor bit distinguishes ID-sending cycles from data-sending cycles. The transmitting processor starts by sending the ID of the receiving processor with which it wants to communicate as data with the multiprocessor bit set to 1. Next, the transmitting processor sends transmit data with the multiprocessor bit cleared to 0.

Receiving processors skip incoming data until they receive data with the multiprocessor bit set to 1. When they receive data with the multiprocessor bit set to 1, receiving processors compare the data with their IDs. The receiving processor with a matching ID continues to receive further incoming data. Processors with IDs not matching the received data skip further incoming data until they again receive data with the multiprocessor bit set to 1. Multiple processors can send and receive data in this way.

Figure 13.9 shows an example of communication among processors using the multiprocessor format.

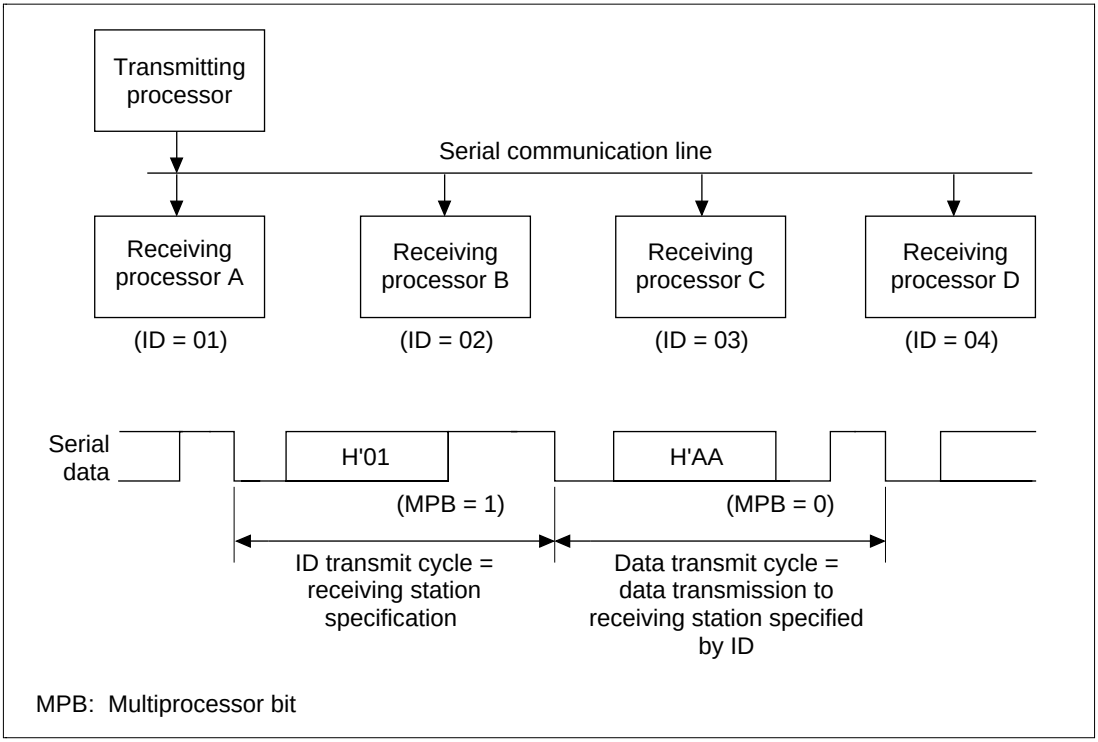


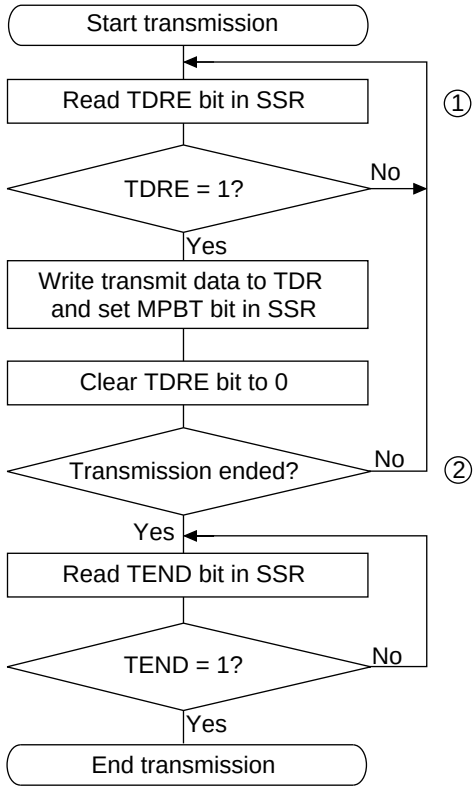

**Figure 13.9 Example of Communication among Processors Using Multiprocessor Format (Sending Data H'AA to Receiving Processor A)**

**Transfer Formats:** Four formats are available. Parity-bit settings are ignored when the multiprocessor format is selected. For details see table 13.8.

**Clock:** See the description in the asynchronous mode section.

**Transmitting Multiprocessor Serial Data:** Figure 13.10 shows a sample flowchart for transmitting multiprocessor serial data. The procedure for transmitting multiprocessor serial data is as follows:

1. SCI status check and transmit data write: read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR). Also set the MPBT (multiprocessor bit transfer) bit to 0 or 1 in SSR. Finally, clear TDRE to 0.
2. Serial transmission continuation procedure: to continue transmitting serial data, read the TDRE bit to check whether it is safe to write (if it reads 1); if so, write data in TDR, then clear TDRE to 0.




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.10 Sample Flowchart for Transmitting Multiprocessor Serial Data**

In transmitting serial data, the SCI operates as follows:

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from TDR into the transmit shift register (TSR).

2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) in SCR is set to 1, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

   Serial transmit data is transmitted in the following order from the TxD pin:

   a. Start bit: one 0 bit is output.
   b. Transmit data: seven or eight bits are output, LSB first.
   c. Multiprocessor bit: one multiprocessor bit (MPBT value) is output.
   d. Stop bit: one or two 1-bits (stop bits) are output.
   e. Mark state: output of 1-bits continues until the start bit of the next transmit data.

3. The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads data from TDR into TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit in SSR to 1, outputs the stop bit, then continues output of 1-bits in the mark state. If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested at this time.

Figure 13.11 shows an example of SCI transmit operation using a multiprocessor format.

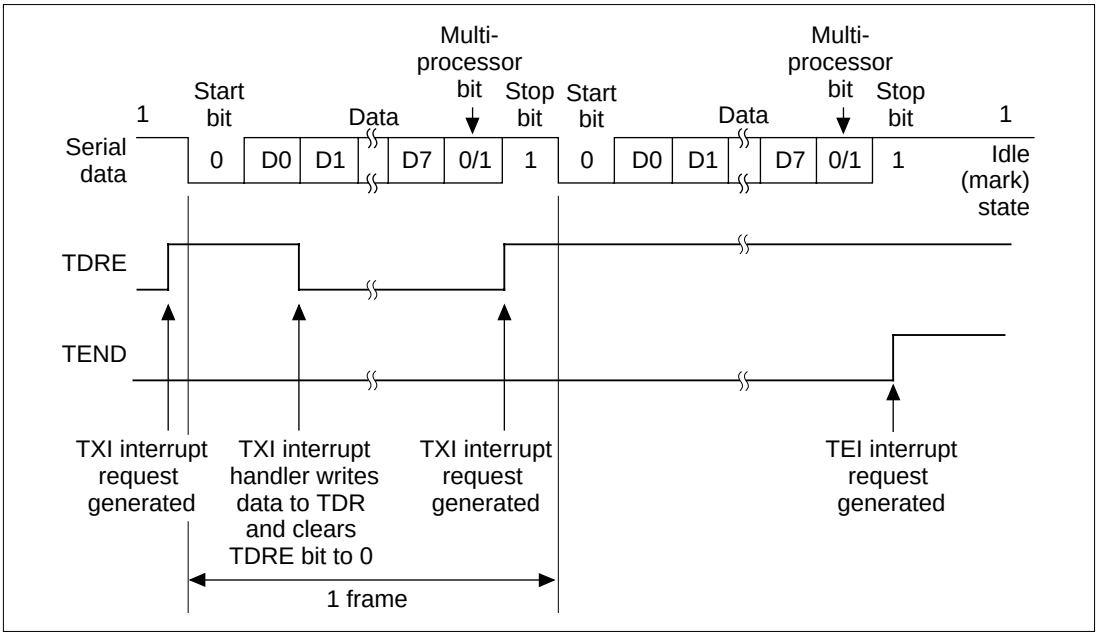
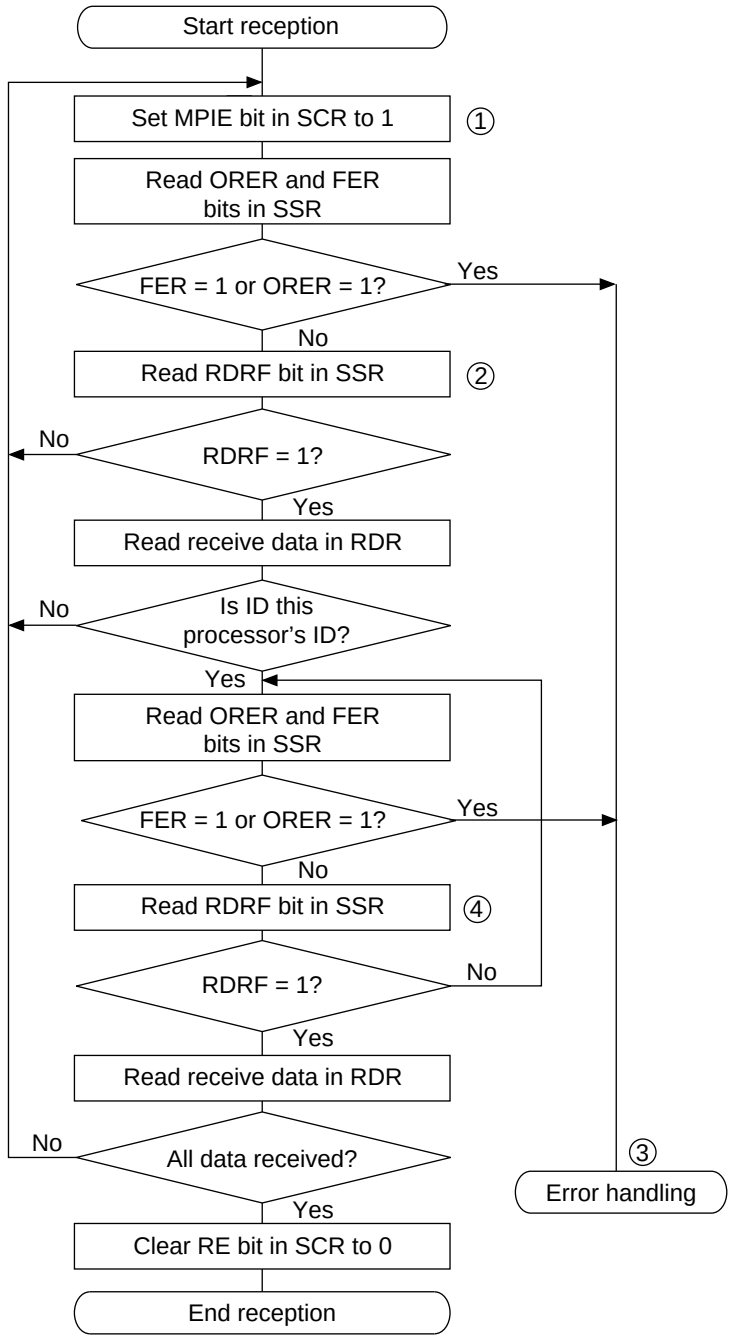



**Figure 13.11 Example of SCI Multiprocessor Transmit Operation (8-Bit Data with Multiprocessor Bit and One Stop Bit)**

**Receiving Multiprocessor Serial Data:** Figure 13.12 shows a sample flowchart for receiving multiprocessor serial data. The procedure for receiving multiprocessor serial data is as follows.

1. ID receive cycle: set the MPIE bit in the serial control register (SCR) to 1.
2. SCI status check, ID reception and comparison: read the serial status register (SSR), check that RDRF is set to 1, then read data from the receive data register (RDR) and compare with the processor's own ID. If the ID does not match the receive data, set MPIE to 1 again and clear RDRF to 0. If the ID matches the receive data, clear RDRF to 0.
3. Receive error handling: if a receive error occurs (figure 13.12 (cont)), read the ORER and FER bits in SSR to identify the error. After executing the necessary error handling, clear both ORER and FER to 0. Receiving cannot resume if ORER or FER remains set to 1.
4. SCI status check and data receiving: read SSR, check that RDRF is set to 1, then read data from the receive data register (RDR).




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.12 Sample Flowchart for Receiving Multiprocessor Serial Data**

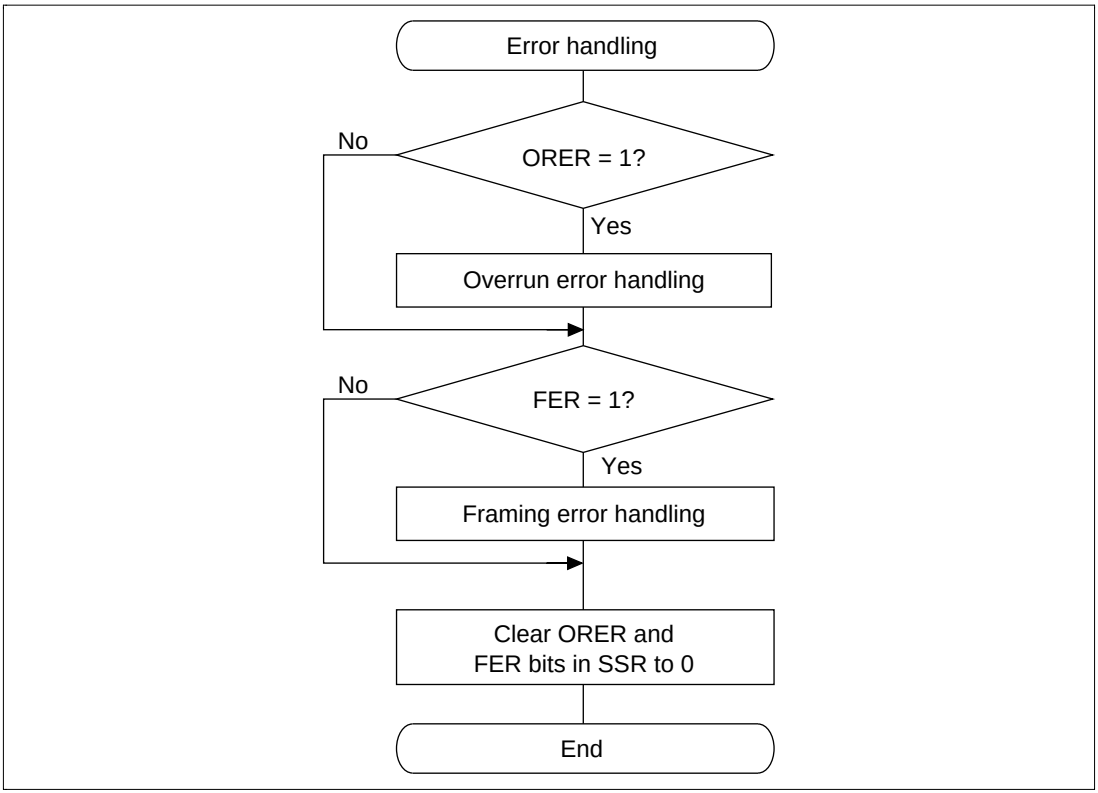



**Figure 13.12 Sample Flowchart for Receiving Multiprocessor Serial Data (cont)**

Figure 13.13 shows an example of SCI receive operation using a multiprocessor format.

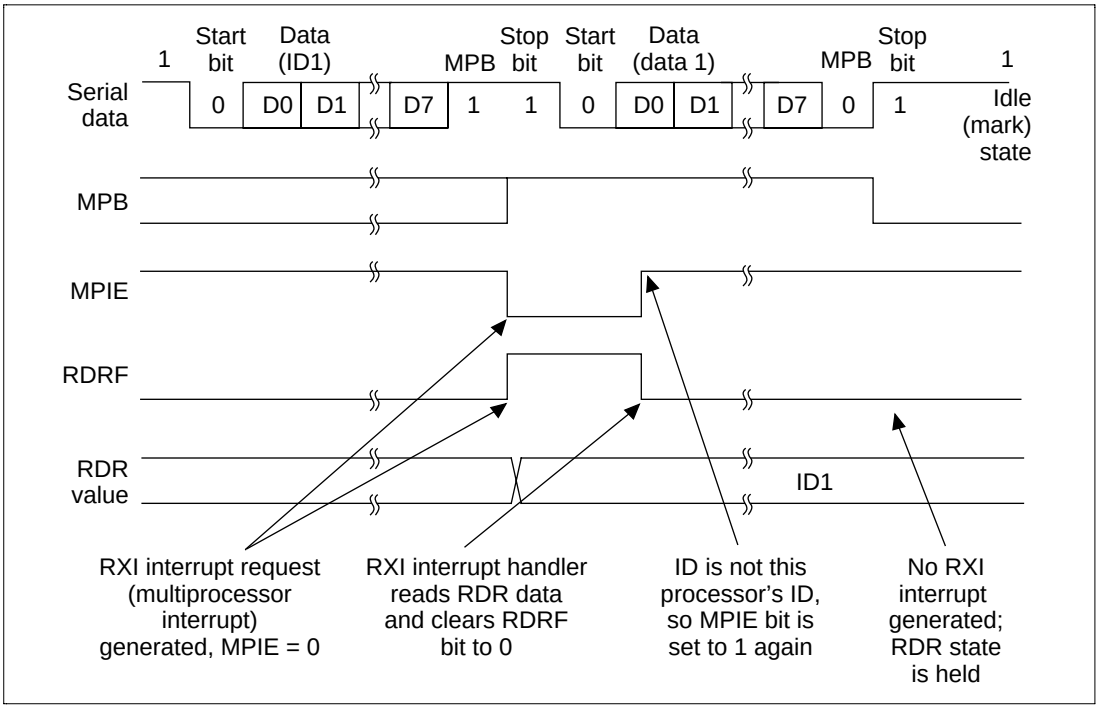



**Figure 13.13 Example of SCI Receive Operation**
**(Own ID Does Not Match Data, 8-Bit Data with Multiprocessor Bit and One Stop Bit)**

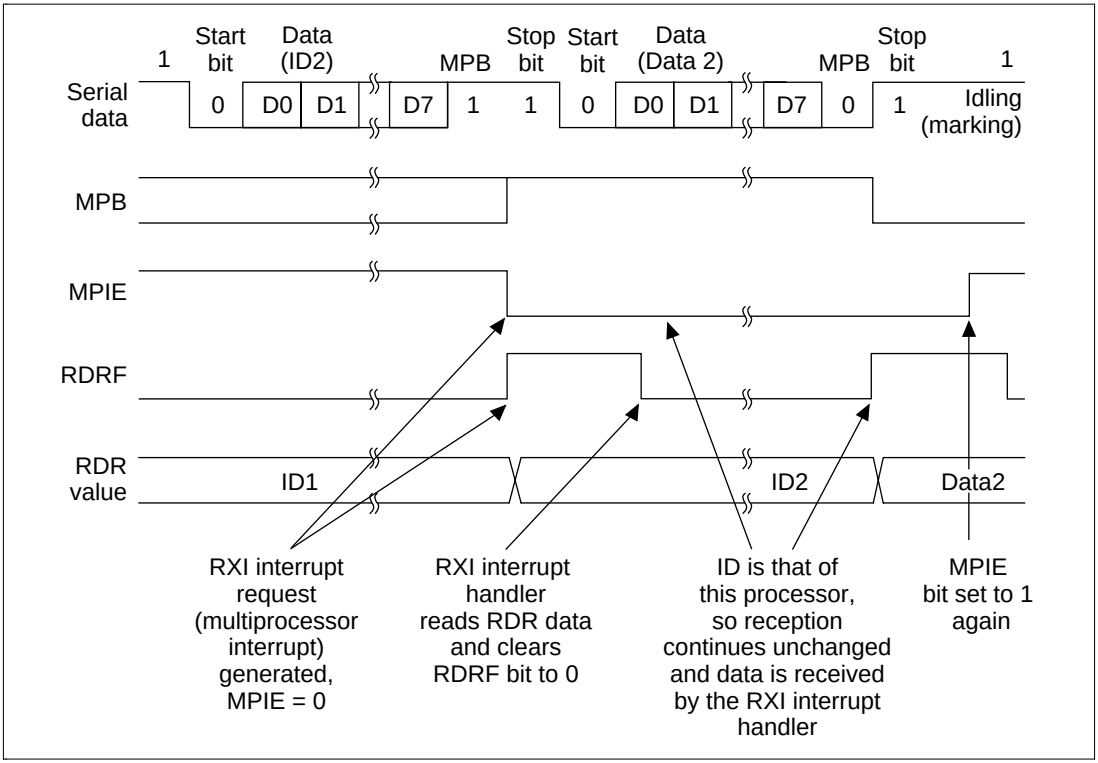



**Figure 13.13 Example of SCI Receive Operation**
**(Own ID Matches Data, 8-Bit Data with Multiprocessor Bit and One Stop Bit) (cont)**

### 13.3.4 Clocked Synchronous Operation

In clocked synchronous mode, the SCI transfers data in synchronization with clock pulses. This mode is suitable for high-speed serial communication.

The SCI transmitter and receiver are independent, so full duplex communication is possible while sharing the same clock. The transmitter and receiver are also double buffered, so continuous transfer is possible by reading or writing data while transfer is in progress.

Figure 13.14 shows the general format in clocked synchronous serial communication.

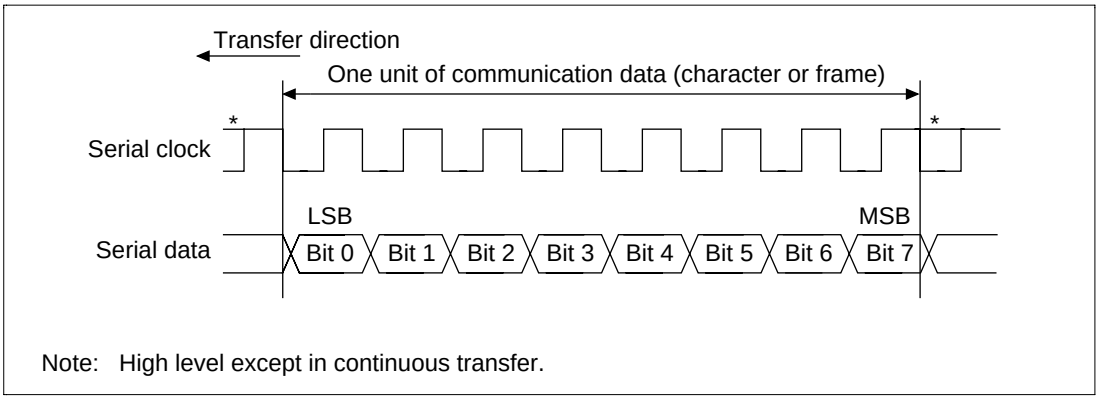



**Figure 13.14 Data Format in Clocked Synchronous Communication**

In clocked synchronous serial communication, each data bit is output on the communication line from one falling edge of the serial clock to the next. Data is guaranteed valid at the rising edge of the serial clock. In each character, the serial data bits are transmitted in order from LSB (first) to MSB (last). After output of the MSB, the communication line remains in the state of the MSB. In clocked synchronous mode, the SCI receives data by synchronizing with the rising edge of the serial clock.

**Transfer Format:** The data length is fixed at eight bits. No parity bit or multiprocessor bit can be added.

**Clock:** An internal clock generated by the built-in baud rate generator or an external clock input from the SCK pin can be selected as the SCI clock source. The clock source is selected by the C/$\overline{A}$ bit in the serial mode register (SMR) and bits CKE1 and CKE0 in the serial control register (SCR). See table 13.9.

When the SCI operates on an internal clock, it outputs the clock signal at the SCK pin. Eight clock pulses are output per transferred character. When the SCI is not transmitting or receiving, the clock signal remains in the high state.

In receive-only operations, the SCI performs receive operations with two characters as one unit, and therefore a 16-pulse serial clock is output. To perform receive operations using a one-character unit, an external clock should be selected as the clock source.

Figure 13.15 shows an example of SCI transmit operation. In transmitting serial data, the SCI operates as follows.

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data and loads this data from TDR into the transmit shift register (TSR).

2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) in SCR is set to 1, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

   If clock output mode is selected, the SCI outputs eight synchronous clock pulses. If an external clock source is selected, the SCI outputs data in synchronization with the input clock. Data is output from the TxD pin in order from LSB (bit 0) to MSB (bit 7).
3. The SCI checks the TDRE bit when it outputs the MSB (bit 7). If TDRE is 0, the SCI loads data from TDR into TSR, transmits the MSB, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit in SSR to 1, transmits the MSB (bit 7), then holds the transmit data pin (TxD) in the MSB state. If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested at this time.
4. After the end of serial transmission, the SCK pin is held in the high state.

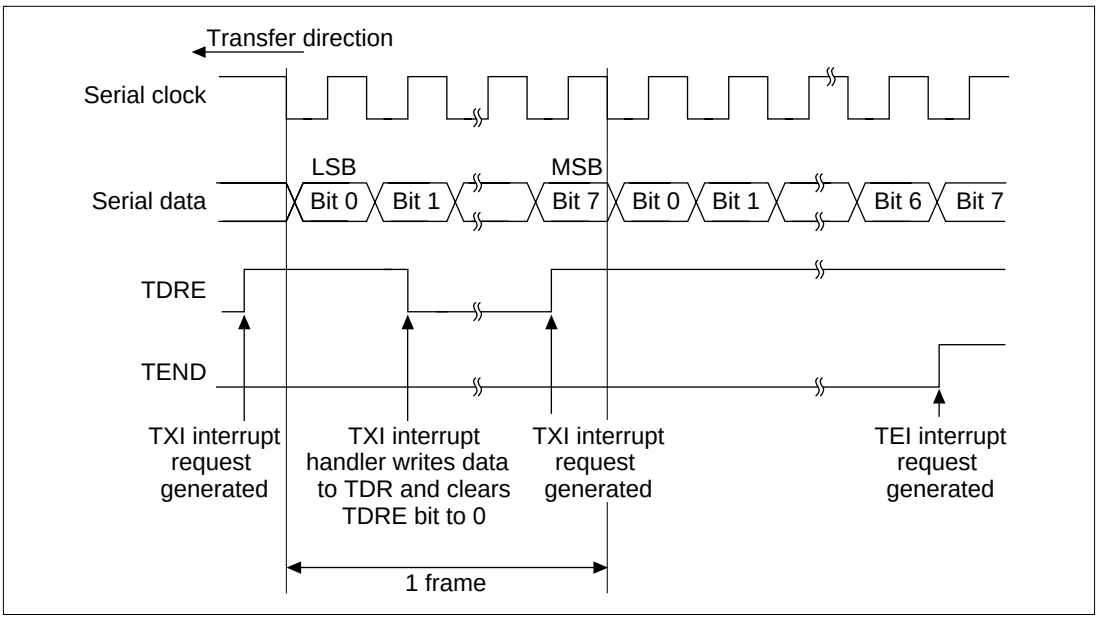



**Figure 13.15 Example of SCI Transmit Operation**

**Transferring Data**

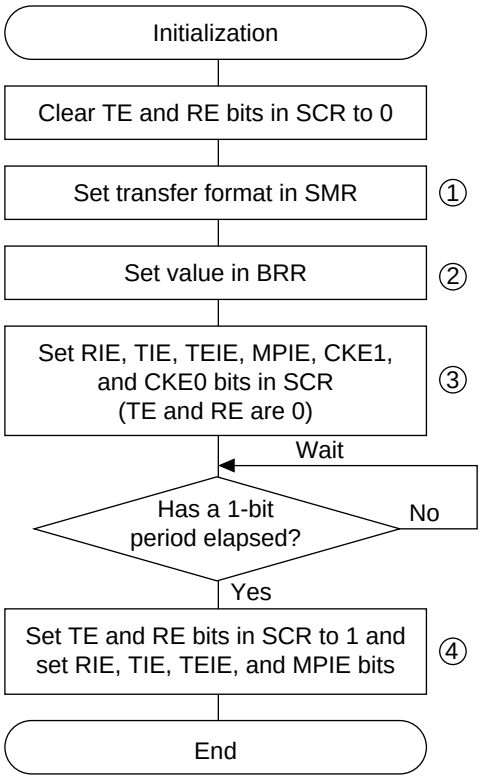
**SCI Initialization (Clocked Synchronous Mode):** Before transmitting or receiving, software must clear the TE and RE bits to 0 in the serial control register (SCR), then initialize the SCI as follows.

When changing the mode or communication format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 sets TDRE to 1 and initializes the transmit shift register (TSR). Clearing RE to 0, however, does not initialize the RDRF, PER, FER, and ORER flags and receive data register (RDR), which retain their previous contents.

Figure 13.16 shows a sample flowchart for initializing the SCI. The procedure for initializing the SCI is as follows.

1. Select the transfer format in the serial mode register (SMR).
2. Write the value corresponding to the bit rate in the bit rate register (BRR) unless an external clock is used.
3. Select the clock source in the serial control register (SCR). Leave RIE, TIE, TEIE, MPIE, TE and RE cleared to 0.
4. Wait for at least the interval required to transmit or receive one bit, then set TE or RE in the serial control register (SCR) to 1. Also set RIE, TIE, TEIE and MPIE.

Note: In simultaneous transmission/reception, the TE bit and RE bit should be cleared to 0 or set to 1 simultaneously.

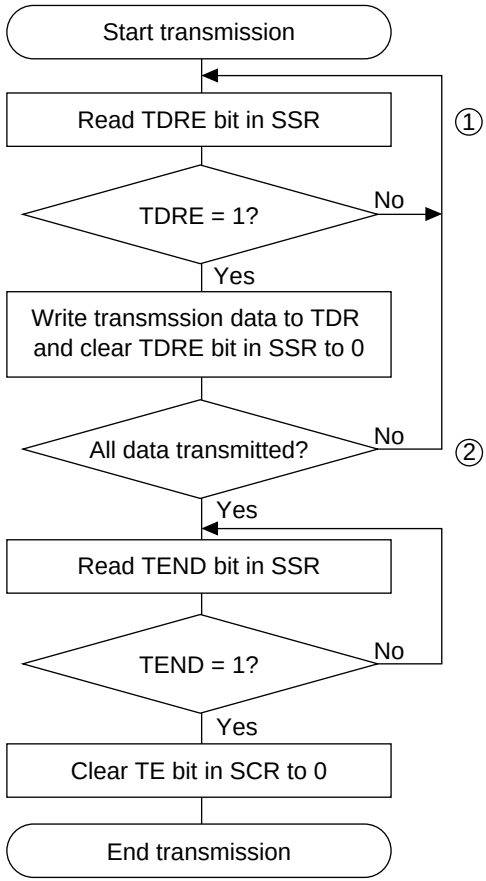



Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.16 Sample Flowchart for SCI Initialization**

**Transmitting Serial Data (Clocked Synchronous Mode):** Figure 13.17 shows a sample flowchart for transmitting serial data. The procedure for transmitting serial data is as follows.

1. SCI status check and transmit data write: read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR) and clear TDRE to 0.
2. Serial transmission continuation procedure: to continue transmitting serial data, read the TDRE bit to check whether it is safe to write (if it reads 1); if so, write data in TDR, then clear TDRE to 0.



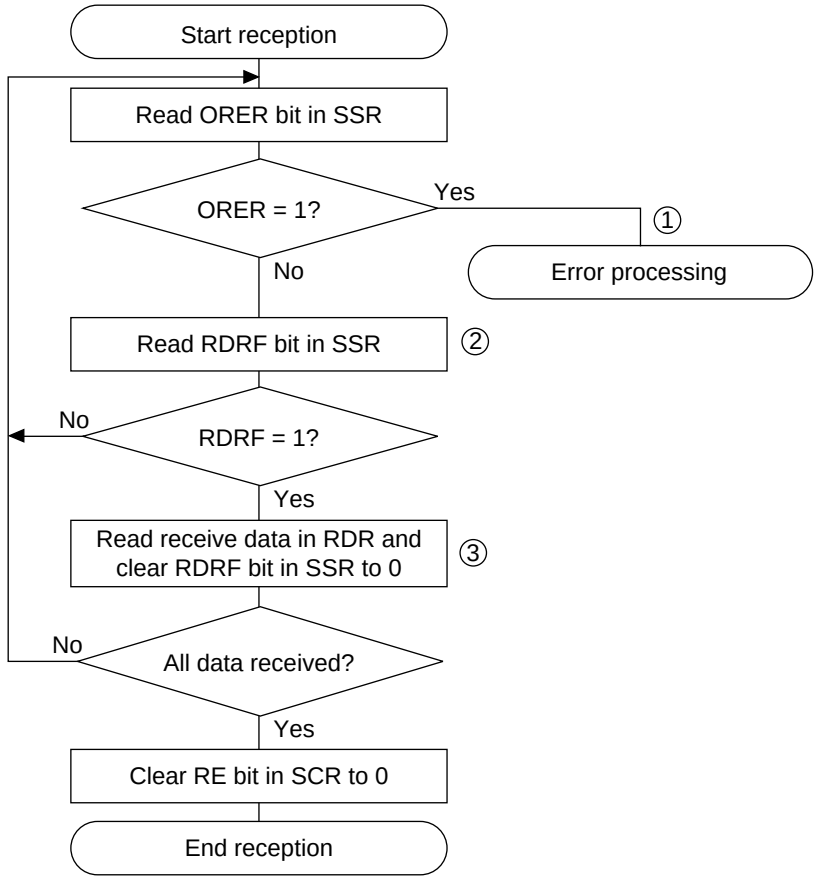

Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.17 Sample Flowchart for Serial Transmitting**

**Receiving Serial Data (Clocked Synchronous Mode):** Figure 13.18 shows a sample flowchart for receiving serial data. Use the following procedure for serial data reception. When switching from asynchronous mode to clocked synchronous mode, make sure that ORER, PER, and FER are cleared to 0. If PER or FER is set to 1, the RDRF bit will not be set and transfer will be disabled. Figure 13.19 shows an example of the SCI receive operation.

The procedure for receiving serial data is as follows:

1. Receive error handling: if a receive error occurs, read the ORER bit in SSR to identify the error. After executing the necessary error handling, clear ORER to 0. Transfer cannot resume if ORER remains set to 1.
2. SCI status check and receive data read: read the serial status register (SSR), check that RDRF is set to 1, then read receive data from the receive data register (RDR) and clear RDRF to 0. The RXI interrupt can also be used to determine if the RDRF bit has changed from 0 to 1.
3. To continue receiving serial data: read the receive data register (RDR) and clear the RDRF bit to 0 before the MSB (bit 7) of the current frame is received.




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.18 Sample Flowchart for Serial Receiving**

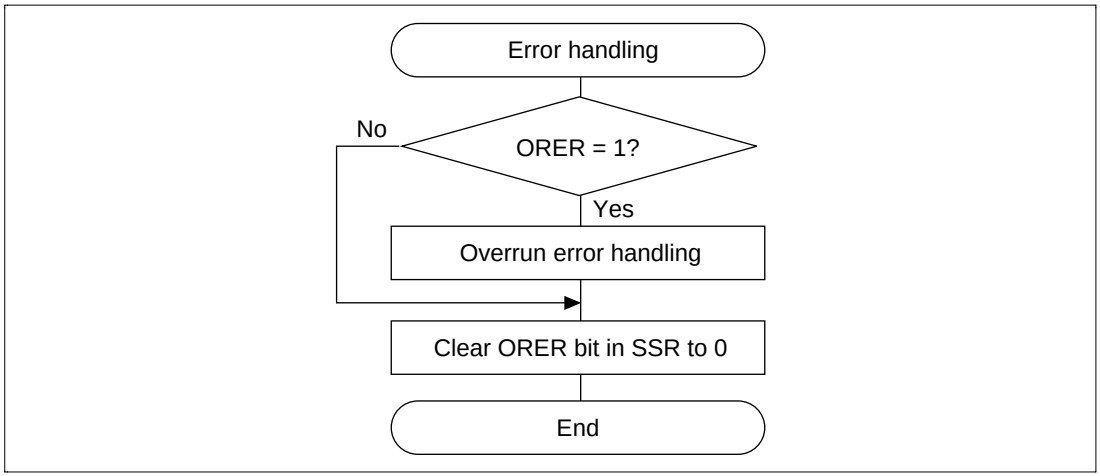



**Figure 13.18 Sample Flowchart for Serial Receiving (cont)**

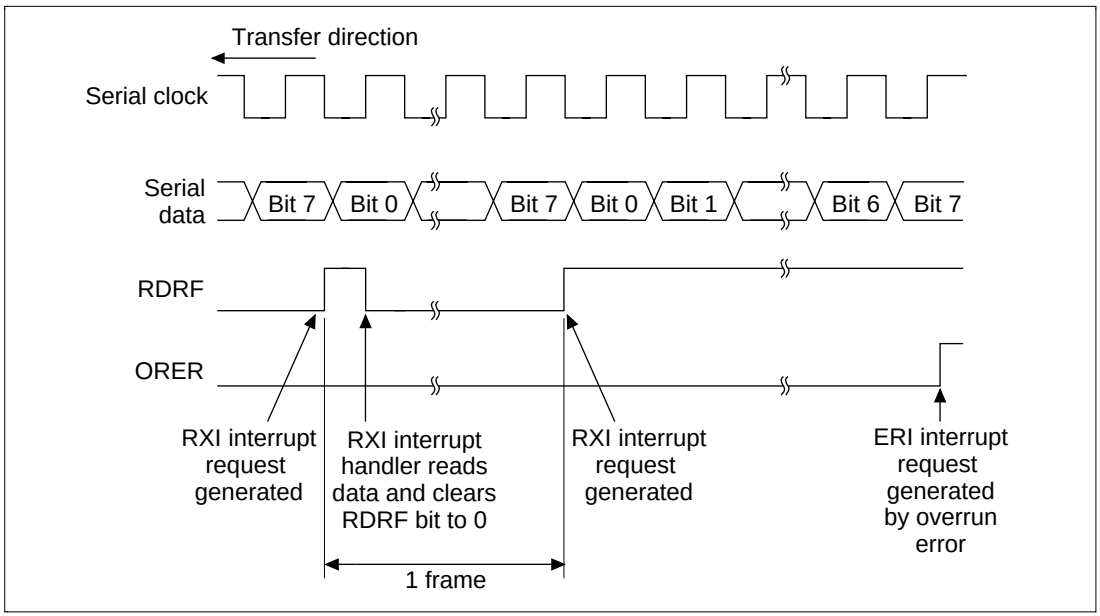


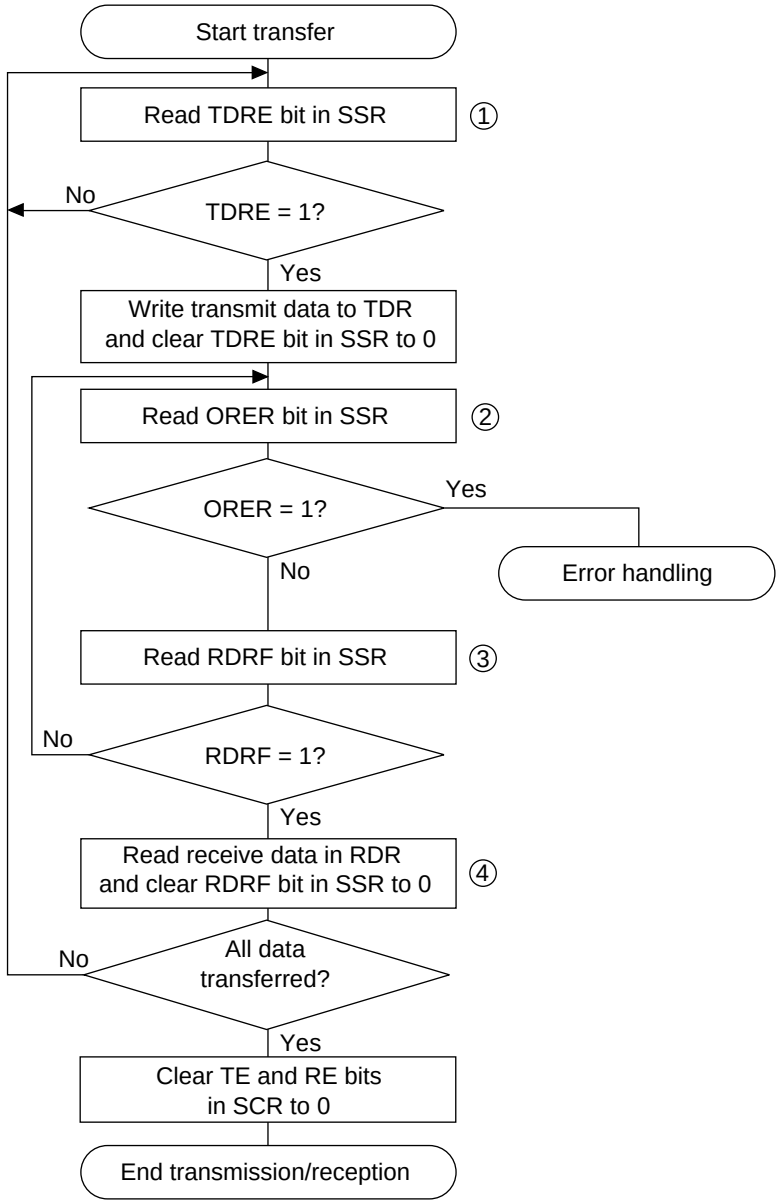

**Figure 13.19 Example of SCI Receive Operation**

In receiving, the SCI operates as follows:

1. The SCI synchronizes with serial clock input or output and initializes internally.
2. Receive data is shifted into RSR in order from LSB to MSB. After receiving the data, the SCI checks that RDRF is 0 so that receive data can be loaded from RSR into RDR. If this check passes, the SCI sets RDRF to 1 and stores the received data in RDR. If the check does not pass (receive error), the SCI operates as indicated in table 13.8. No further transmit or receive operations are possible in this state. The RDRF bit is not set to 1. Be sure to clear the error flag.
3. After setting RDRF to 1, if the receive-data-full interrupt enable bit (RIE) is set to 1 in SCR, the SCI requests a receive-data-full interrupt (RXI). If the ORER bit is set to 1 and the receive-data-full interrupt enable bit (RIE) in SCR is also set to 1, the SCI requests a receive-error interrupt (ERI).

**Transferring Serial Data Simultaneously (Clocked Synchronous Mode):** Figure 13.20 shows a sample flowchart for transferring serial data simultaneously. The procedure for transmitting and receiving serial data simultaneously is as follows:

1. SCI status check and transmit data write: read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR) and clear TDRE to 0. The TXI interrupt can also be used to determine if the TDRE bit has changed from 0 to 1.
2. Receive error handling: if a receive error occurs, read the ORER bit in SSR to identify the error. After executing the necessary error handling, clear ORER to 0. Transfer cannot resume if ORER remains set to 1.
3. SCI status check and receive data read: read the serial status register (SSR), check that RDRF is set to 1, then read receive data from the receive data register (RDR) and clear RDRF to 0. The RXI interrupt can also be used to determine if the RDRF bit has changed from 0 to 1.
4. Serial transfer continuation procedure: to continue transferring serial data, read the RDRF bit and RDR, and clear RDRF to 0 before the MSB (bit 7) of the current frame is received. Also read the TDRE bit to check whether it is safe to write (if it reads 1); if so, write data in TDR, then clear TDRE to 0 before the MSB (bit 7) of the current frame is transmitted.




Note: When switching from transmitting or receiving to simultaneous transfer, clear both TE and RE to 0, then set both TE and RE to 1.
Circled numbers refer to the preceding description of the procedure in the text.

**Figure 13.20 Sample Flowchart for Serial Transmitting/Receiving**

## 13.4 SCI Interrupt Sources and the DMAC

The SCI has four interrupt sources in each channel: transmit-end (TEI), receive-error (ERI), receive-data-full (RXI), and transmit-data-empty (TXI). Table 13.13 lists the interrupt sources and indicates their priority. These interrupts can be enabled and disabled by the TIE, RIE, and TEIE bits in the serial control register (SCR). Each interrupt request is sent separately to the interrupt controller.

TXI is requested when the TDRE bit in SSR is set to 1.

RXI is requested when the RDRF bit in SSR is set to 1.

ERI is requested when the ORER, PER, or FER bit in SSR is set to 1.

TEI is requested when the TEND bit in SSR is set to 1. Where the TXI interrupt indicates that transmit data writing is enabled, the TEI interrupt indicates that the transmit operation is complete.

**Table 13.13 SCI Interrupt Sources**

| Interrupt Source | Description | DMAC Availability | Priority |
|---|---|---|---|
| ERI | Receive error (ORER, PER, or FER) | | High |
| RXI | Receive data register full (RDRF) | | ↑ |
| TXI | Transmit data register empty (TDRE) | | ↓ |
| TEI | Transmit end (TEND) | | Low |

See section 4, Exception Handling, for information on the priority order and relationship to non-SCI interrupts.

## 13.5 Usage Notes

Note the following points when using the SCI.

### 13.5.1 TDR Write and TDRE Flag

The TDRE bit in the serial status register (SSR) is a status flag indicating loading of transmit data from TDR into TSR. The SCI sets TDRE to 1 when it transfers data from TDR to TSR. Data can be written to TDR regardless of the TDRE bit status. If new data is written in TDR when TDRE is 0, however, the old data stored in TDR will be lost because the data has not yet been transferred to TSR. Before writing transmit data to TDR, be sure to check that TDRE is set to 1.

### 13.5.2 Simultaneous Multiple Receive Errors

Table 13.14 indicates the state of the SSR status flags when multiple receive errors occur simultaneously. When an overrun error occurs, the RSR contents cannot be transferred to RDR, so receive data is lost.

**Table 13.14 SSR Status Flags and Transfer of Receive Data**

| | SSR Status Flags | | | | Receive Data Transfer |
|---|---|---|---|---|---|
| **Receive Error Status** | **RDRF** | **ORER** | **FER** | **PER** | **RSR → RDR** |
| Overrun error | 1 | 1 | 0 | 0 | X |
| Framing error | 0 | 0 | 1 | 0 | O |
| Parity error | 0 | 0 | 0 | 1 | O |
| Overrun error + framing error | 1 | 1 | 1 | 0 | X |
| Overrun error + parity error | 1 | 1 | 0 | 1 | X |
| Framing error + parity error | 0 | 0 | 1 | 1 | O |
| Overrun error + framing error + parity error | 1 | 1 | 1 | 1 | X |

Note: O: Receive data is transferred from RSR to RDR.
X: Receive data is not transferred from RSR to RDR.

### 13.5.3 Break Detection and Processing

In the break state, the input from the RxD pin consists of all 0s, so FER is set and the parity error flag (PER) may also be set. In the break state, the SCI receiver continues to operate, so if the FER bit is cleared to 0, it will be set to 1 again.

### 13.5.4 Receive Error Flags and Transmitter Operation (Clocked Synchronous Mode Only)

When a receive error flag (ORER, PER, or FER) is set to 1, the SCI will not start transmitting even if TDRE is set to 1. Be sure to clear the receive error flags to 0 before starting to transmit. Note that clearing RE to 0 does not clear the receive error flags.

### 13.5.5 Receive Data Sampling Timing and Receive Margin in Asynchronous Mode

In asynchronous mode, the SCI operates on a base clock of 16 times the bit rate frequency. In receiving, the SCI synchronizes internally with the falling edge of the start bit, which it samples on the base clock. Receive data is latched on the rising edge of the eighth base clock pulse. See figure 13.21.

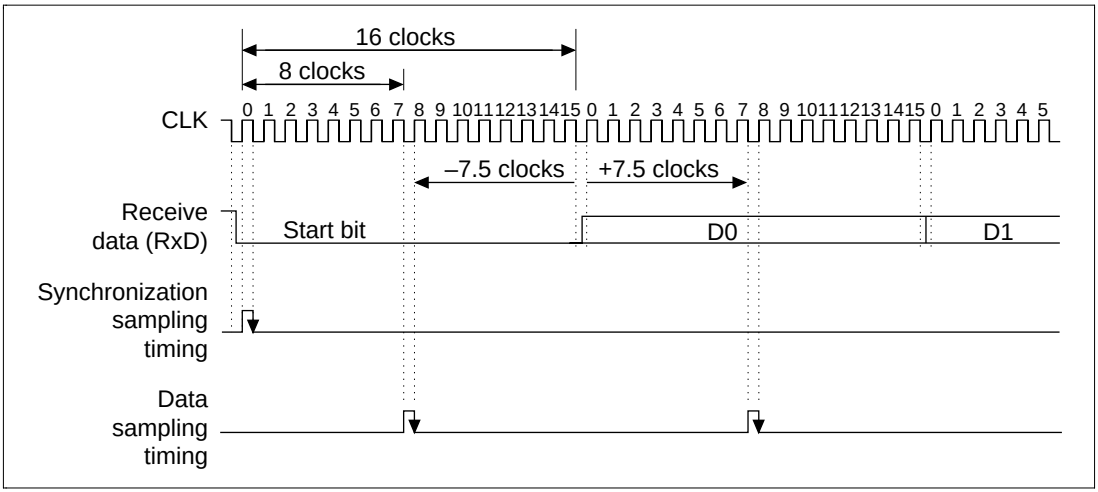



**Figure 13.21 Receive Data Sampling Timing in Asynchronous Mode**

The receive margin in asynchronous mode can therefore be expressed as in equation 1.

Equation 1:

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

M : Receive margin (%)
N : Ratio of clock frequency to bit rate (N = 16)
D : Clock duty cycle (D = 0–1.0)
L : Frame length (L = 9–12)
F : Absolute deviation of clock frequency

From equation (1), if F = 0 and D = 0.5 the receive margin is 46.875%, as given by equation 2.

Equation 2:

$D = 0.5, F = 0$

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

This is a theoretical value. A reasonable margin to allow in system designs is 20–30%.

### 13.5.6 Cautions for Clocked Synchronous External Clock Mode

- Set TE = RE = 1 only when external clock SCK is 1.
- Do not set TE = RE = 1 until at least four clock cycles after external clock SCK has changed from 0 to 1.
- When receiving, RDRF is set to 1 when RE is cleared to 0, 2.5–3.5 clocks after the rising edge of the RxD D7 bit SCK input, but it cannot be copied to RDR.

### 13.5.7 Caution for Clocked Synchronous Internal Clock Mode

When receiving, RDRF is set to 1 when RE is cleared to 0, 1.5 clocks after the rising edge of the RxD D7 bit SCK output, but it cannot be copied to RDR.

# Section 14 Smart Card Interface

## 14.1 Overview

An IC card (smart card) interface conforming to the ISO/IEC 7816-3 (Identification Card) data transmission protocol format (T = 0; asynchronous full-duplex character transmission protocol) is supported as a serial communication interface (SCI) extension function. Register settings are used to switch between the ordinary serial communication interface and the smart card interface.

### 14.1.1 Features

The smart card interface has the following features:

- Asynchronous mode
  - Data length: 8 bits
  - Parity bit generation and check
  - Receive mode error signal detection (parity error)
  - Error signal detection and automatic data retransmission in transmit
  - Direct convention and inverse convention both supported
- On-chip baud rate generator any bit rate to be supported.
- Three interrupt sources: Transmit-data-empty, receive-data-full, and communication-error interrupts are requested independently.

### 14.1.2 Block Diagram

Figure 14.1 shows a block diagram of the smart card interface.

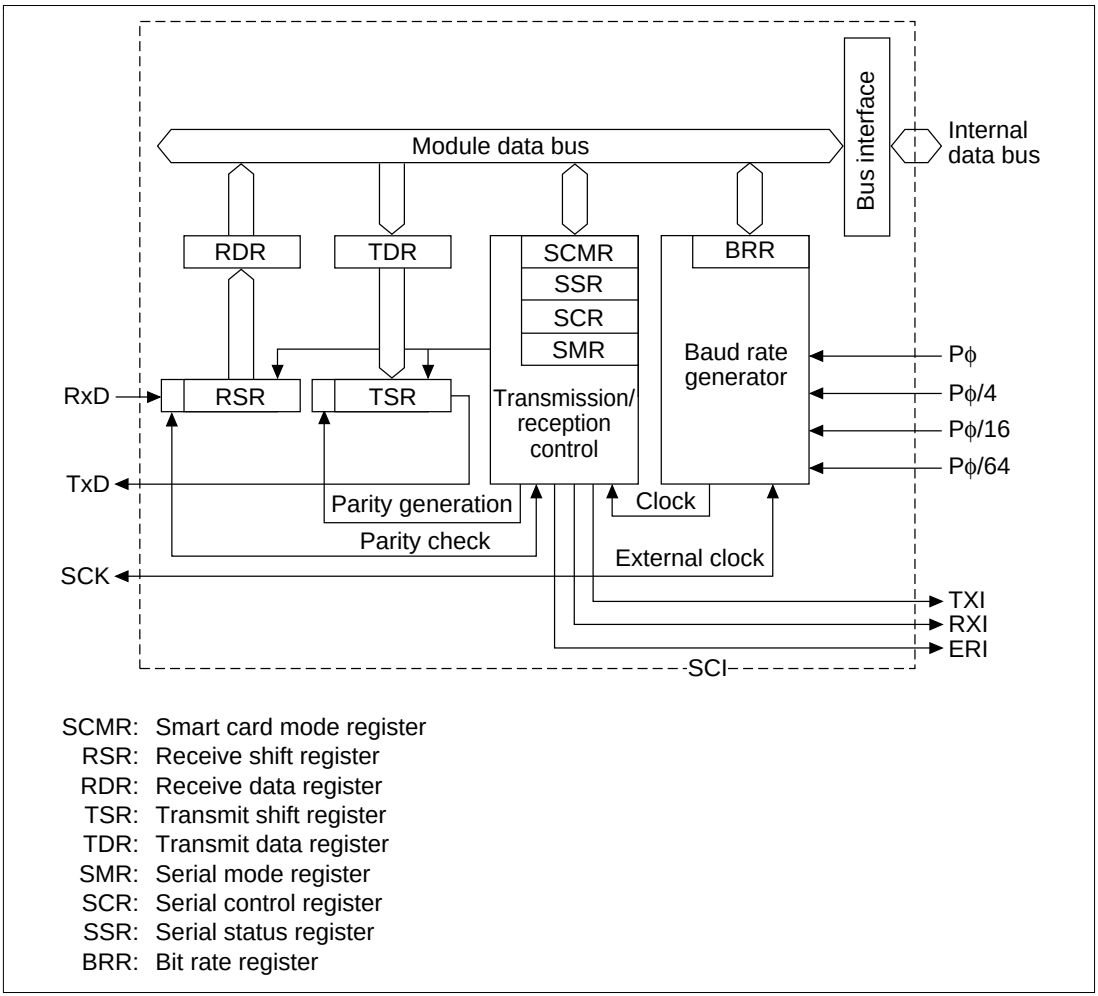



**Figure 14.1 Smart Card Interface Block Diagram**

### 14.1.3 Pin Configuration

Table 14.1 shows the smart card interface pins.

**Table 14.1 Pin Configuration**

| Pin Name | Abbreviation | Input/Output | Function |
|---|---|---|---|
| Serial clock pin | SCK | Output | Clock output |
| Receive data pin | RxD | Input | Receive data input |
| Transmit data pin | TxD | Output | Transmit data output |

### 14.1.4 Register Configuration

Table 14.2 shows the registers used by the smart card interface. The SMR, BRR, SCR, TDR, and RDR registers are the same as in the ordinary SCI function. They are described in section 13, Serial Communication Interface.

**Table 14.2 Register Configuration**

| Name | Abbreviation | R/W | Initial Value*[3] | Address | Access Size |
|---|---|---|---|---|---|
| Serial mode register | SMR | R/W | H'00 | H'FFFFFE00 | 8 |
| Bit rate register | BRR | R/W | H'FF | H'FFFFFE01 | 8 |
| Serial control register | SCR | R/W | H'00 | H'FFFFFE02 | 8 |
| Transmit data register | TDR | R/W | H'FF | H'FFFFFE03 | 8 |
| Serial status register | SSR | R/(W)*[1] | H'84 | H'FFFFFE04 | 8 |
| Receive data register | RDR | R | H'00 | H'FFFFFE05 | 8 |
| Smart card mode register | SCMR | R/W | *[2] | H'FFFFFE06 | 8 |

Notes: 1. Only 0 can be written, to clear the flags.
2. Bits 0, 2, and 3 are cleared. The value of the other bits is undefined.
3. Initialized by a power-on or manual reset.

## 14.2 Register Descriptions

This section describes the registers added for the smart card interface and the bits whose functions are changed.

### 14.2.1 Smart Card Mode Register (SCMR)

The smart card mode register (SCMR) is an 8-bit read/write register that selects smart card interface functions. SMR bits 0, 2, and 3 are initialized to 0 by a reset and in standby mode.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | SDIR | SINV | — | SMIF |
| Initial value: | — | — | — | — | 0 | 0 | — | 0 |
| R/W: | — | — | — | — | R/W | R/W | — | R/W |

Bits 7 to 4 and 1—Reserved: An undefined value will be returned if these bits are read.

Bit 3—Smart Card Data Transfer Direction (SDIR): Selects the serial/parallel conversion format.

| Bit 3: SDIR | Description |
|---|---|
| 0 | Contents of TDR are transferred LSB first, receive data is stored in RDR LSB first (Initial value) |
| 1 | Contents of TDR are transferred MSB first, receive data is stored in RDR MSB first |

Bit 2—Smart Card Data Inversion (SINV): Specifies whether to invert the logic level of the data. This function is used in combination with bit 3 for transmitting and receiving with an inverse convention card. SINV does not affect the logic level of the parity bit. See section 14.3.4, Register Settings, for information on how parity is set.

| Bit 2: SINV | Description |
|---|---|
| 0 | Contents of TDR are transferred unchanged, receive data is stored in RDR unchanged (Initial value) |
| 1 | Contents of TDR are inverted before transfer, receive data is inverted before storage in RDR |

Bit 0—Smart Card Interface Mode Select (SMIF): Enables the smart card interface function.

| Bit 0: SMIF | Description | |
|---|---|---|
| 0 | Smart card interface function disabled | (Initial value) |
| 1 | Smart card interface function enabled | |

### 14.2.2 Serial Mode Register (SMR)

In the smart card interface mode, the function of SMR bit 7 is changed.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | GM(C/$\overline{A}$) | CHR | PE | O/$\overline{E}$ | STOP | MP | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bit 7—GSM Mode: Sets the smart card interface function to GSM mode. With the normal smart card interface, this bit is cleared to 0. Setting this bit to 1 selects GSM mode, an additional mode for controlling the timing for setting the TEND flag that indicates completion of transmission, and the type of clock output used. The details of the additional clock output control mode are specified by the CKE1 and CKE0 bits in the serial control register (SCR).

| Bit 7: GM | Description | |
|---|---|---|
| 0 | Normal smart card interface mode operation | (Initial value) |
| | 1. The TEND flag is set 12.5 etu after the beginning of the start bit | |
| | 2. Clock output on/off control only | |
| 1 | GSM mode smart card interface mode operation | |
| | 1. The TEND flag is set 11.0 etu after the beginning of the start bit | |
| | 2. Clock output on/off and fixed-high/fixed-low control (set in SCR) | |

Bits 6 to 0: These bits have the same function as in the ordinary SCI. See section 13, Serial Communication Interface, for more information.

### 14.2.3 Serial Control Register (SCR)

In the smart card interface mode, the function of SCR bits 1 and 0 is changed.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 7 to 2: These bits have the same function as in the ordinary SCI. See section 13, Serial Communication Interface, for more information.

Bits 1 and 0—Clock Enable 1 and 0 (CKE1, CKE0): These bits specify the function of the SCK pin. In the smart card interface mode, an internal clock is always used as the internal clock source. In this mode, it is possible to specify a fixed high level or fixed low level for the clock output, in addition to the usual switching between enabling and disabling of the clock output.

| SMR: GM (C/$\overline{A}$) | Bit 1: CKE1 | Bit 0: CKE0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | Input pin (input ignored) |
| | | 1 | Clock output as SCK output pin |
| | 1 | 0 | Invalid setting; do not set |
| | | 1 | Invalid setting; do not set |
| 1 | 0 | 0 | Fixed low output as output pin |
| | | 1 | Clock output as output pin |
| | 1 | 0 | Fixed high output as output pin |
| | | 1 | Clock output as output pin |

### 14.2.4 Serial Status Register (SSR)

In the smart card interface mode, the function of SSR bit 4 is changed. The setting conditions for bit 2, the TEND bit, are also changed.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TDRE | RDRF | ORER | FER/ERS | PER | TEND | MPB | MPBT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R | R | R/W |

Note: * Only 0 can be written, to clear the flag.

Bits 7 to 5: These bits have the same function as in the ordinary SCI. See section 13, Serial Communication Interface, for more information.

Bit 4—Error Signal Status (ERS): In the smart card interface mode, bit 4 indicates the status of the error signal returned from the receiving side during transmission. The smart card interface cannot detect framing errors.

| Bit 4: ERS | Description |
|---|---|
| 0 | Receiving ended normally with no error signal (Initial value)<br>ERS is cleared to 0 when the chip is reset or enters standby mode, or when software reads ERS after it has been set to 1, then writes 0 in ERS |
| 1 | An error signal indicating a parity error was transmitted from the receiving side<br>ERS is set to 1 if the error signal sampled is low |

Note: The ERS flag maintains its status even when the TE bit in SCSCR1 is cleared to 0.

Bit 3—Parity Error (PER): This bit has the same function as in the ordinary SCI. See section 13, Serial Communication Interface, for more information.

Bit 2—Transmit End (TEND): The setting conditions for bit 2, Transmit End (TEND), are shown below.

| Bit 2: TEND | Description |
|---|---|
| 0 | Transmission is in progress<br>TEND is cleared to 0 when software reads TDRE after it has been set to 1, then writes 0 in TDRE |
| 1 | Transmission has ended (Initial value)<br>TEND is set to 1 when:<br>• the chip is reset or enters standby mode<br>• the TE bit in SCR is 0 and the FER/ERS bit is also 0<br>• the GM bit in SMR is 0, and TDRE = 1 and FER/ERS = 0 (normal transmission) 2.5 etu after a one-byte serial character is transmitted<br>• the GM bit in SMR is 1, and TDRE = 1 and FER/ERS = 0 (normal transmission) 1.0 etu after a one-byte serial character is transmitted |

etu: Elementary time unit (time for transfer of 1 bit)

The TEND generation timing is shown in figure 14.2.

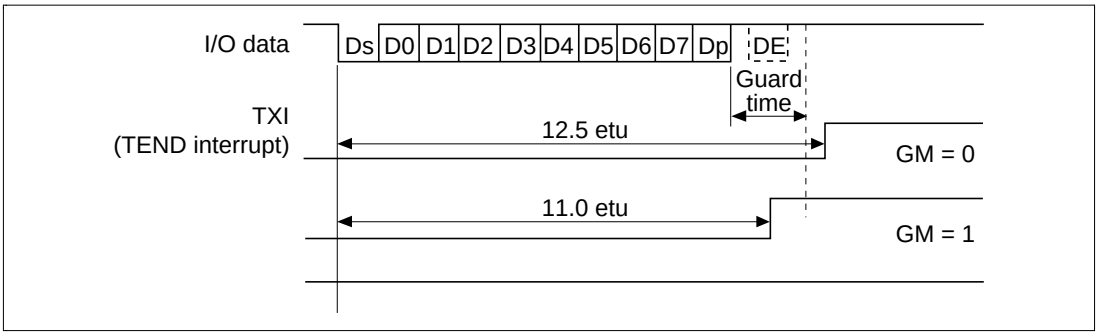



**Figure 14.2 TEND Generation Timing**

Bits 1 and 0: These bits have the same function as in the ordinary SCI. See section 13, Serial Communication Interface, for more information.

## 14.3 Operation

### 14.3.1 Overview

The smart card interface supports a protocol system T = 0; asynchronous duplex character transmission protocol conforming to the ISO/IEC 7816-3 standard. The transmission protocol configuration is shown in figure 14.3. Its primary functions are described below.

1. Each frame consists of 8 data bits and 1 parity bit.
2. During transmission, a guard time of at least 2 etu (elementary time units: time for transfer of 1 bit) is left from the end of the parity bit to the start of the next frame.
3. During reception, a low error signal level is output for 1 etu after 10.5 etu has elapsed from the start bit if a parity error was detected.
4. During transmission, it automatically transmits the same data after allowing at least 2 etu from the time the error signal is sampled.
5. Only start-stop type asynchronous communication functions are supported; no synchronous communication functions are available.

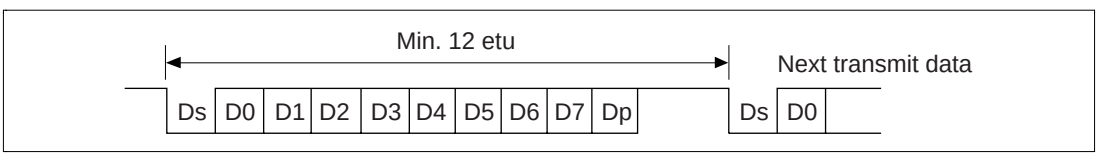



**Figure 14.3 Transmission Protocol Configuration**

### 14.3.2 Pin Connections

Figure 14.4 shows the pin connection diagram for the smart card interface. During communication with an IC card, transmission and reception are both carried out over the same data transfer line, so connect the TxD and RxD pins on the chip. Pull up the data transfer line to the power supply $V_{CC}$ side with a resistor.

When using the clock generated by the smart card interface on an IC card, input the SCK pin output to the IC card's CLK pin. This connection is not necessary when the internal clock is used on the IC card.

Use the chip's port output as the reset signal. Apart from these pins, the power and ground pin connections are usually also required.

Note: When the IC card is not connected and both RE and TE are set to 1, closed transfer is possible and auto-diagnosis can be performed.

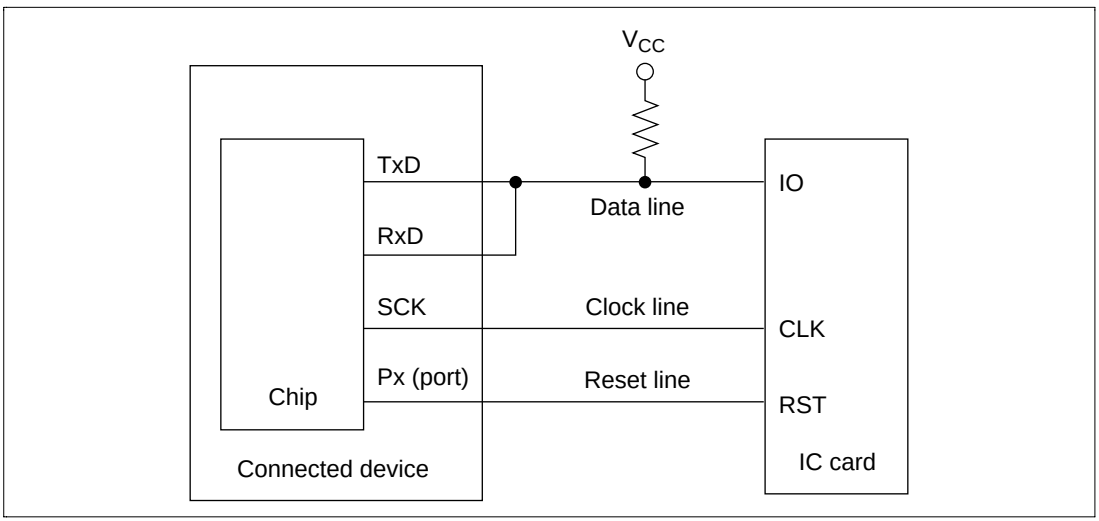



**Figure 14.4 Pin Connection Diagram for the Smart Card Interface**

### 14.3.3 Data Format

Figure 14.5 shows the data format for the smart card interface. In this mode, parity is checked every frame while receiving and error signals sent back to the transmitting side whenever an error is detected so that retransmission of the data is requested. During transmission, error signals are sampled and data re-transmitted whenever an error signal is sampled.

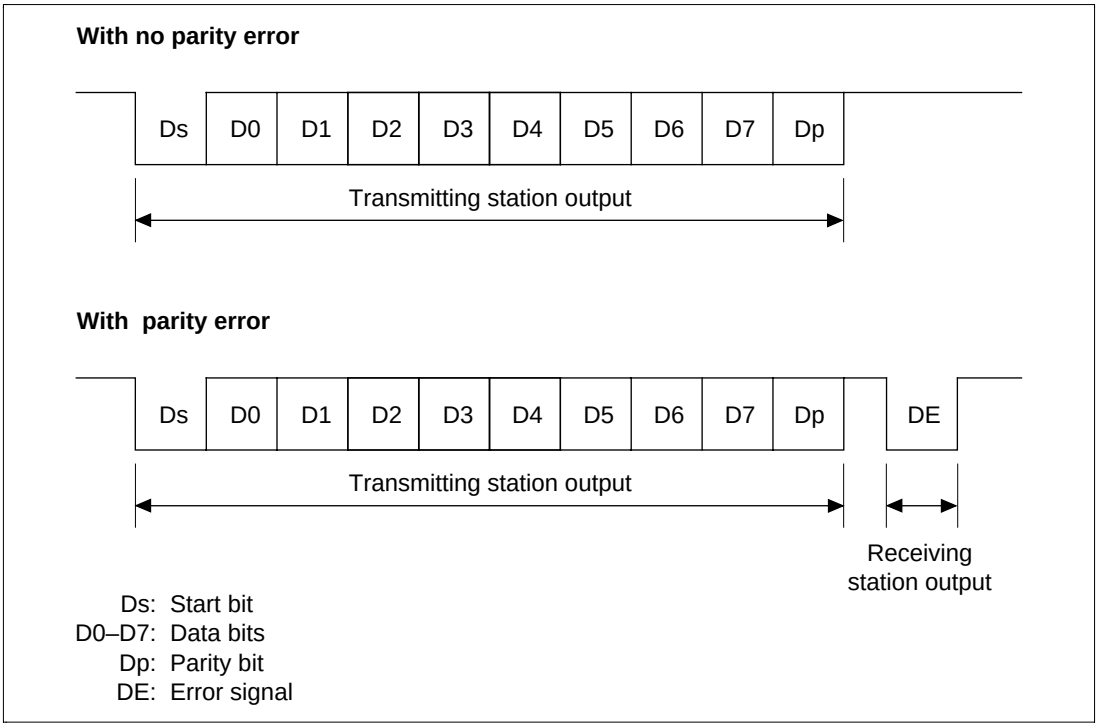



**Figure 14.5 Data Format for Smart Card Interface**

The operating sequence is:

1. The data line is high impedance when not in use and is fixed high with a pull-up resistor.
2. The transmitting side starts one frame of data transmission. The data frame starts with a start bit (Ds, low level). The start bit is followed by eight data bits (D0–D7) and a parity bit (Dp).
3. On the smart card interface, the data line returns to high impedance after this. The data line is pulled high with a pull-up resistor.
4. The receiving side checks parity. When the data is received normally with no parity errors, the receiving side then waits to receive the next data. When a parity error occurs, the receiving side outputs an error signal (DE, low level) and requests re-transfer of data. The receiving station returns the signal line to high impedance after outputting the error signal for a specified period. The signal line is pulled high with a pull-up resistor.
5. The transmitting side transmits the next frame of data unless it receives an error signal. If it does receive an error signal, it returns to step 2 to re-transmit the erroneous data.

### 14.3.4 Register Settings

Table 14.3 shows the bit map of the registers that the smart card interface uses. Bits shown as 1 or 0 must be set to the indicated value. The settings for the other bits are described below.

**Table 14.3 Register Settings for the Smart Card Interface**

| Register | Address | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| SMR | H'FFFFFE80 | GM | 0 | 1 | O/$\overline{E}$ | 1 | 0 | CKS1 | CKS0 |
| BRR | H'FFFFFE82 | BRR7 | BRR6 | BRR5 | BRR4 | BRR3 | BRR2 | BRR1 | BRR0 |
| SCR | H'FFFFFE84 | TIE | RIE | TE | RE | 0 | 0 | CKE1 | CKE0 |
| TDR | H'FFFFFE86 | TDR7 | TDR6 | TDR5 | TDR4 | TDR3 | TDR2 | TDR1 | TDR0 |
| SSR | H'FFFFFE88 | TDRE | RDRF | ORER | FER/ ERS | PER | TEND | 0 | 0 |
| RDR | H'FFFFFE8A | RDR7 | RDR6 | RDR5 | RDR4 | RDR3 | RDR2 | RDR1 | RDR0 |
| SCMR | H'FFFFFE8C | — | — | — | — | SDIR | SINV | — | SMIF |

Note: Dashes indicate unused bits.

1. Setting the serial mode register (SMR): The GM bit selects the TEND flag setting timing, and, in combination with the CKE1 and CKE0 bits in the serial control register (SCR), the clock output state. Set the O/$\overline{E}$ bit to 0 when the IC card uses the direct convention or to 1 when it uses the inverse convention. Select the on-chip baud rate generator clock source with the CKS1 and CKS0 bits (see section 14.3.5, Clock).
2. Setting the bit rate register (BRR): Set the bit rate. See section 14.3.5, Clock, to see how to calculate the set value.
3. Setting the serial control register (SCR): The TIE, RIE, TE and RE bits function as they do for the ordinary SCI. See section 13, Serial Communication Interface, for more information. The CKE0 and CKE1 bits select the clock output state. See section 14.3.5, Clock, for more information.
4. Setting the smart card mode register (SCMR): The SDIR and SINV bits are both set to 0 for IC cards that use the direct convention and both to 1 when the inverse convention is used. The SMIF bit is set to 1 for the smart card interface.

Figure 14.6 shows sample waveforms for register settings of the two types of IC cards (direct convention and inverse convention) and their start characters.

In the direct convention type, the logical 1 level is state Z, the logical 0 level is state A, and communication is LSB first. The start character data is H'3B. The parity bit is even (from the smart card standards), and thus a 1.

In the inverse convention type, the logical 1 level is state A, the logical 0 level is state Z, and communication is MSB first. The start character data is H'3F. The parity bit is even (from the smart card standards), and thus a 0, which corresponds to state Z.

Only data bits D7–D0 are inverted by the SINV bit. To invert the parity bit, set the O/$\overline{E}$ bit in SMR to odd parity mode. This applies to both transmission and reception.

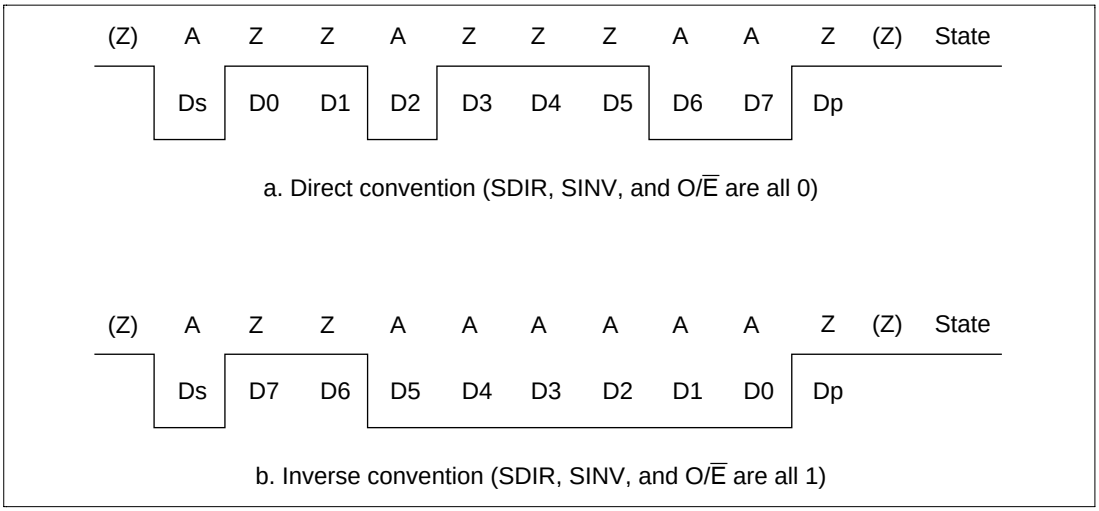



**Figure 14.6 Waveform of Start Character**

### 14.3.5 Clock

Only the internal clock generated by the on-chip baud rate generator can be used as the communication clock in the smart card interface. The bit rate for the clock is set by the bit rate register (BRR) and the CKS1 and CKS0 bits in the serial mode register (SMR), and is calculated using the equation below. Table 14.5 shows sample bit rates. If clock output is then selected by setting CKE0 to 1, a clock with a frequency 372 times the bit rate is output from the SCK pin.

$$B = \frac{P\phi}{1488 \times 2^{2n-1} \times (N + 1)} \times 10^6$$

Where:

N = BRR setting ($0 \leq N \leq 255$)
B = Bit rate (bit/s)
$P\phi$ = Peripheral module clock (MHz)
n = 0–3 (table 14.4)

**Table 14.4 Relationship of n, CKS1, and CKS0**

| n | CKS1 | CKS0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |

**Table 14.5 Examples of Bit Rate B (Bit/s) for BRR Settings (n = 0)**

| | Pϕ (MHz) | | | | | | |
|---|---|---|---|---|---|---|---|
| **N** | **7.1424** | **10.00** | **10.7136** | **13.00** | **14.2848** | **16.00** | **18.00** |
| 0 | 9600.0 | 13440.9 | 14400.0 | 17473.1 | 19200.0 | 21505.4 | 24193.5 |
| 1 | 4800.0 | 6720.4 | 7200.0 | 8736.6 | 9600.0 | 10752.7 | 12096.8 |
| 2 | 3200.0 | 4480.3 | 4800.0 | 5824.4 | 6400.0 | 7168.5 | 8064.5 |

Note: The bit rate is rounded to two decimal places.

Calculate the value to be set in the bit rate register (BRR) from the peripheral module clock and the bit rate. N is an integer in the range $0 \le N \le 255$, specifying a smaller error.

$$N = \frac{P\phi}{1488 \times 2^{2n-1} \times B} \times 10^6 - 1$$

**Table 14.6 Examples of BRR Settings for Bit Rate B (Bit/s) (n = 0)**

| Pϕ (MHz) (9600 Bits/s) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **7.1424** | | **10.00** | | **10.7136** | | **13.00** | | **14.2848** | | **16.00** | | **18.00** | |
| **N** | **Error** | **N** | **Error** | **N** | **Error** | **N** | **Error** | **N** | **Error** | **N** | **Error** | **N** | **Error** |
| 0 | 0.00 | 1 | 30.00 | 1 | 25.00 | 1 | 8.99 | 1 | 0.00 | 1 | 12.01 | 2 | 15.99 |

**Table 14.7 Maximum Bit Rates for Frequencies (Smart Card Interface Mode)**

| **Pϕ (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 |
| 14.2848 | 19200 | 0 | 0 |
| 16.00 | 21505 | 0 | 0 |
| 18.00 | 24194 | 0 | 0 |

The bit rate error is found as follows:

$$\text{Error}(\%) = \left(\frac{P\phi}{1488 \times 2^{2n-1} \times B \times (N+1)} \times 10^6 - 1\right) \times 100$$

Table 14.8 shows the relationship between transmit/receive clock register set values and output states on the smart card interface. Figure 14.7 shows the difference in clock output according to the setting of the GM bit.

**Table 14.8 Register Set Values and SCK Pin**

| | Register Value | | | | SCK Pin | |
|---|---|---|---|---|---|---|
| **Setting** | **SMIF** | **GM** | **CKE1** | **CKE0** | **Output** | **State** |
| 1*[1] | 1 | 0 | 0 | 0 | Port | Input pin (input ignored) |
| | 1 | 0 | 0 | 1 | ⎍ | SCK (serial clock) output state |
| 2*[2] | 1 | 1 | 0 | 0 | Low output | Low output state |
| | 1 | 1 | 0 | 1 | ⎍ | SCK (serial clock) output state |
| 3*[2] | 1 | 1 | 1 | 0 | High output | High output state |
| | 1 | 1 | 1 | 1 | ⎍ | SCK (serial clock) output state |

Notes: 1. The SCK output state changes as soon as the CKE0 bit is modified. The CKE1 bit should be cleared to 0.

2. The clock duty remains constant despite stopping and starting of the clock by modification of the CKE0 bit.

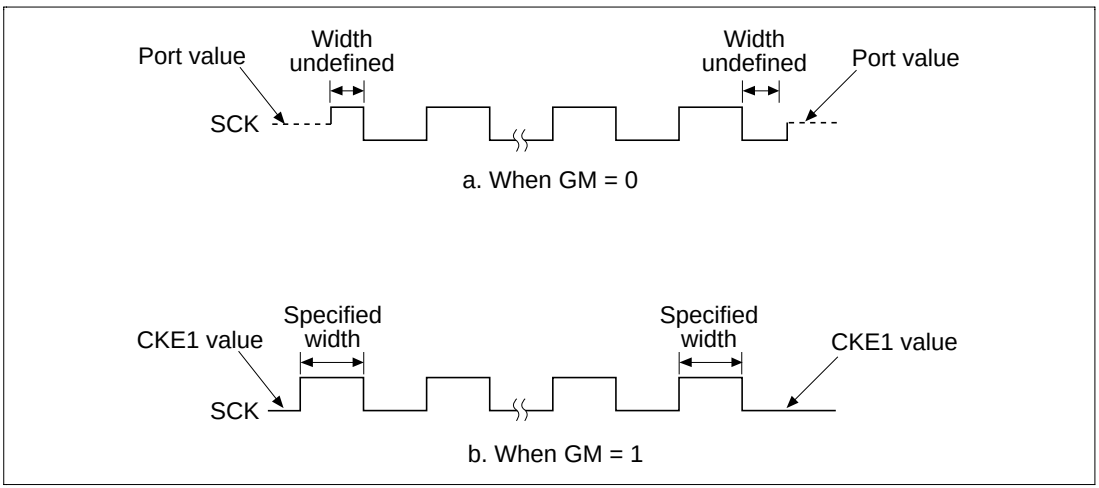
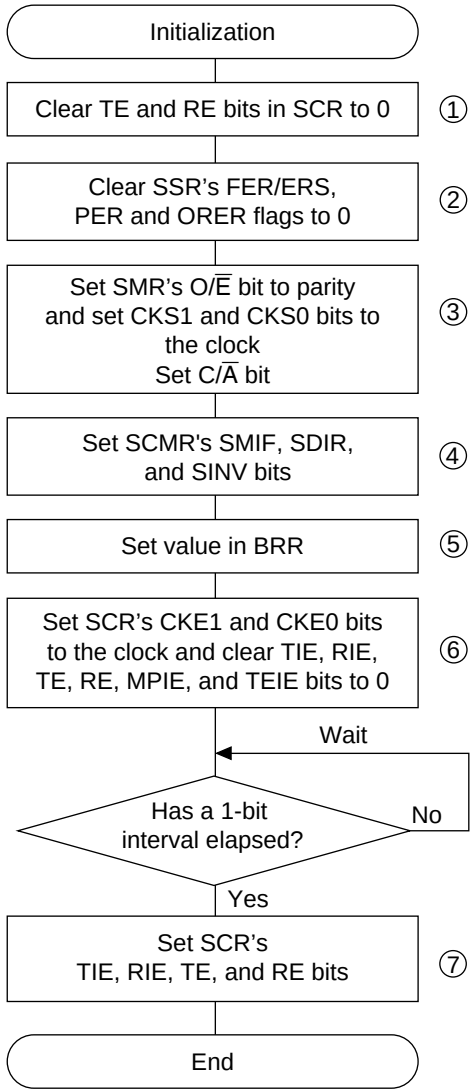



**Figure 14.7 Difference in Clock Output According to GM Bit Setting**

### 14.3.6 Data Transfer Operations

**Initialization:** Initialize the SCI using the following procedure before transmitting and receiving data. Initialization is also required for switching from transmit mode to receive mode or from receive mode to transmit mode. Figure 14.8 shows a flowchart of the initialization process.

1. Clear TE and RE in the serial control register (SCR) to 0.
2. Clear error flags FER/ERS, PER, and ORER to 0 in the serial status register (SSR).
3. Set the C/$\overline{A}$ bit, parity bit (O/$\overline{E}$ bit), and baud rate generator select bits (CKS1 and CKS0 bits) in the serial mode register (SMR). At this time also clear the CHR and MP bits to 0 and set the STOP and PE bits to 1.
4. Set the SMIF, SDIR, and SINV bits in the smart card mode register (SCMR). When the SMIF bit is set to 1, the TxD and RxD pins become high impedance.
5. Set the value corresponding to the bit rate in the bit rate register (BRR).
6. Set the clock source select bits (CKE1 and CKE0 bits) in the serial control register (SCR). Clear the TIE, RIE, TE, RE, MPIE, and TEIE bits to 0. When the CKE0 bit is set to 1, a clock is output from the SCK pin.
7. After waiting at least 1 bit, set the TIE, RIE, TE, and RE bits in SCR. Do not set the TE and RE bits simultaneously unless performing auto-diagnosis.



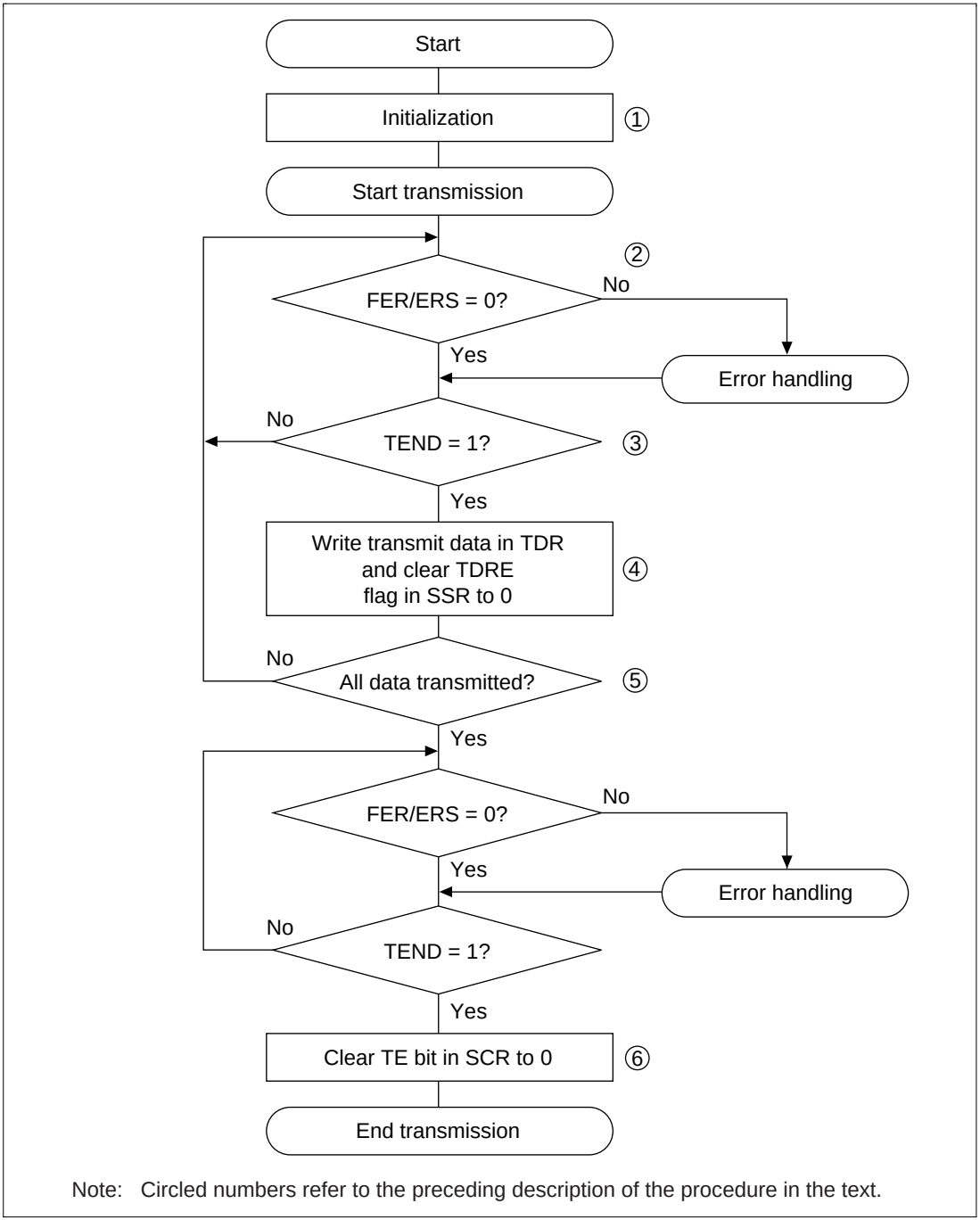

Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 14.8 Initialization Flowchart (Example)**

**Serial Data Transmission:** The handling procedures in the smart card mode differ from ordinary SCI processing because data is retransmitted when an error signal is sampled during a data transmission. This results in the transmission processing flowchart shown in figure 14.9.

1. Initialize the smart card interface mode as described in initialization above and figure 14.8.
2. Check that the FER/ERS bit in SSR is cleared to 0.
3. Repeat steps 2 and 3 until the TEND flag in SSR is set to 1.
4. Write the transmit data into TDR, clear the TDRE flag to 0 and start transmitting. The TEND flag will be cleared to 0.
5. To transmit more data, return to step 2.
6. To end transmission, clear the TE bit to 0.

This processing can be interrupted. When the TIE bit is set to 1 and interrupt requests are enabled, a transmit-data-empty interrupt (TXI) will be requested when the TEND flag is set to 1 at the end of the transmission. When the RIE bit is set to 1 and interrupt requests are enabled, a communication error interrupt (ERI) will be requested when the ERS flag is set to 1 when an error occurs in transmission. See Interrupt Operation below for more information.




Note: Circled numbers refer to the preceding description of the procedure in the text.

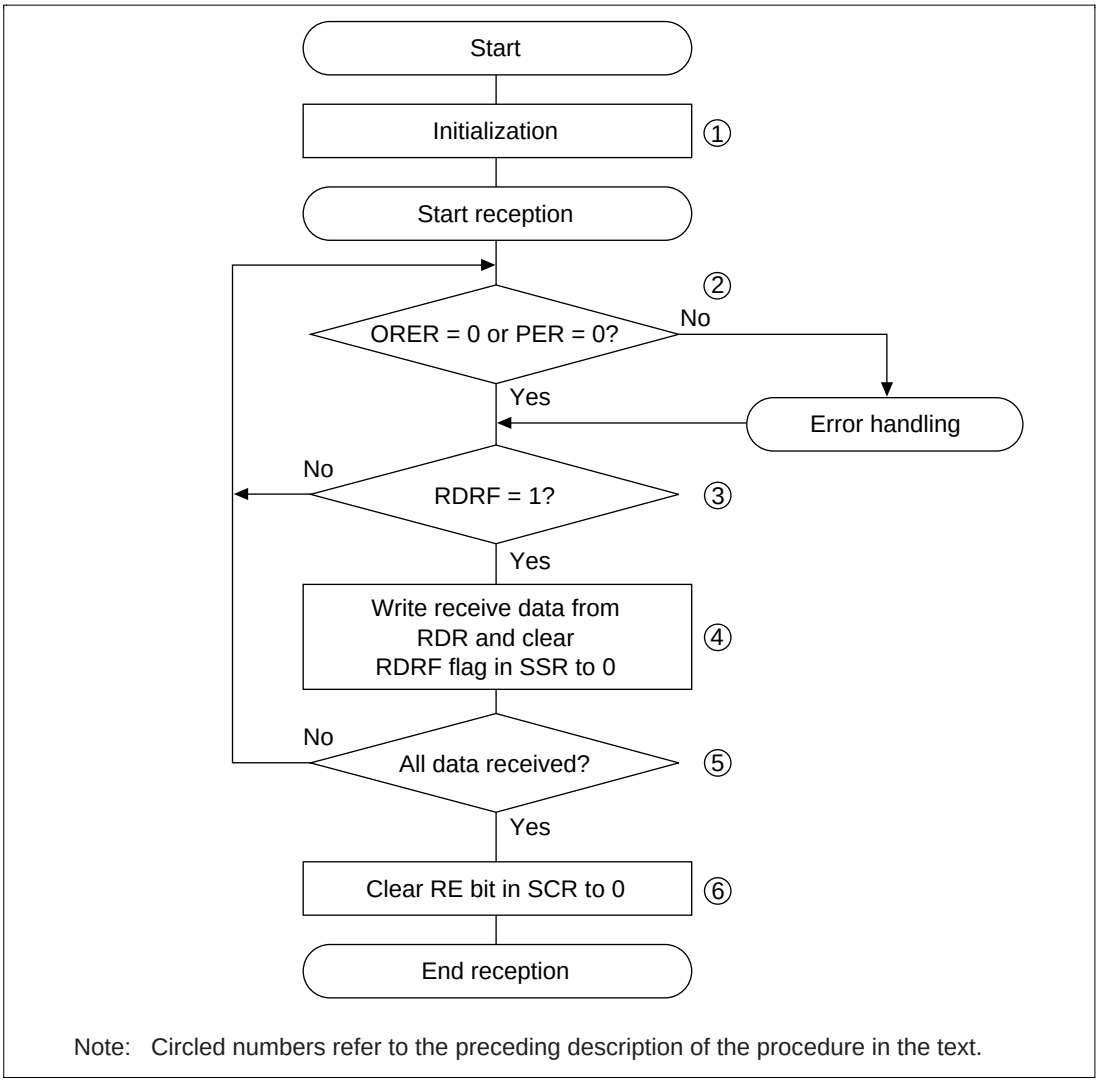
**Figure 14.9 Sample Flowchart for Transmission**

**Serial Data Reception:** The handling procedures in the smart card mode are the same as in ordinary SCI processing. The reception processing flowchart is shown in figure 14.10.

1. Initialize the smart card interface mode as described above in Initialization and in figure 14.5.
2. Check that the ORER and PER flags in SSR are cleared to 0. If either flag is set, clear both to 0 after performing the appropriate error handling procedures.
3. Repeat steps 2 and 3 until the RDRF flag is set to 1.
4. Read the receive data from RDR.
5. To receive more data, clear the RDRF flag to 0 and return to step 2.
6. To end reception, clear the RE bit to 0.

This processing can be interrupted. When the RIE bit is set to 1 and interrupt requests are enabled, a receive-data-full interrupt (RXI) will be requested when the RDRF flag is set to 1 at the end of the reception. When an error occurs during reception and either the ORER or PER flag is set to 1, a communication error interrupt (ERI) will be requested. See Interrupt Operation below for more information.

The received data will be transferred to RDR even when a parity error occurs during reception and PER is set to 1, so this data can still be read.




Note: Circled numbers refer to the preceding description of the procedure in the text.

**Figure 14.10 Sample Flowchart for Reception**

**Switching Modes:** When switching from receive mode to transmit mode, check that the receive operation is completed before starting initialization and setting RE to 0 and TE to 1. The RDRF, PER, and ORER flags can be used to check if reception is completed. When switching from transmit mode to receive mode, check that the transmit operation is completed before starting initialization and setting TE to 0 and RE to 1. The TEND flag can be used to check if transmission is completed.

**Interrupt Operation:** In the smart card interface mode, there are three types of interrupts: transmit-data-empty (TXI), communication error (ERI) and receive-data-full (RXI). In this mode, the transmit-end interrupt (TEI) cannot be requested.

Set the TEND flag in SSR to 1 to request a TXI interrupt. Set the RDRF flag in SSR to 1 to request an RXI interrupt. Set the ORER, PER, or FER/ERS flag in SSR to 1 to request an ERI interrupt (table 14.9).

**Table 14.9 Smart Card Mode Operating Status and Interrupt Sources**

| Mode | Status | Flag | Mask Bit | Interrupt Source |
|---|---|---|---|---|
| Transmit mode | Normal | TEND | TIE | TXI |
| | Error | FER/ERS | RIE | ERI |
| Receive mode | Normal | RDRF | RIE | RXI |
| | Error | PER, ORER | RIE | ERI |

## 14.4 Usage Notes

The following points in section 14.4.1 and 14.4.2 should be noted when using the SCI as a small card interface.

### 14.4.1 Receive Data Timing and Receive Margin in Asynchronous Mode

In asynchronous mode, the SCI runs on a basic clock with a frequency of 372 times the transfer rate. During reception, the SCI samples the fall of the start bit using the base clock to achieve internal synchronization. Receive data is latched internally on the rising edge of the 186th basic clock cycle (figure 14.11).

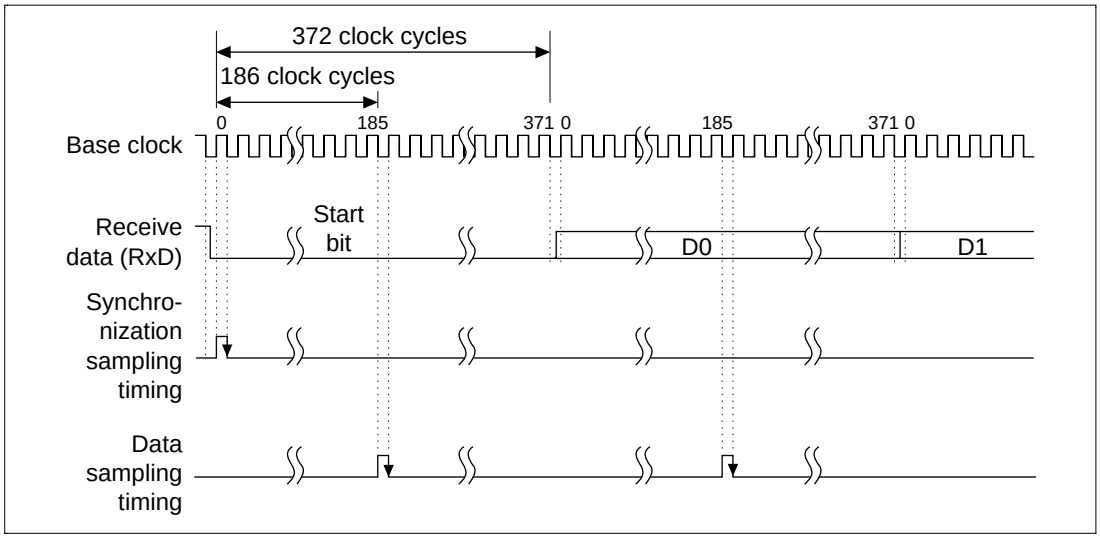



**Figure 14.11 Receive Data Sampling Timing in Smart Card Mode**

The receive margin is found from the following equation:

For smart card mode:

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

Where:

M = Receive margin (%)
N = Ratio of bit rate to clock (N = 372)
D = Clock duty (D = 0 to 1.0)
L = Frame length (L = 10)
F = Absolute value of clock frequency deviation

Using this equation, the receive margin when F = 0 and D = 0.5 is as follows:

$$M = (0.5 - 1/2 \times 372) \times 100\% = 49.866\%$$

### 14.4.2 Retransfer (Receive and Transmit Modes)

SCI retransmission operations in receive mode and transmit mode are described below.

**Retransfer by the SCI in Receive Mode:** Figure 14.12 shows the retransmission operation in the SCI receive mode.

1. When the received parity bit is checked and an error is found, the PER bit in SSR is automatically set to 1. If the RIE bit in SCR is enabled at this time, an ERI interrupt is requested. Be sure to clear the PER bit in SSR to 0 before the next parity bit is sampled.
2. The RDRF bit in SSR is not set in the frame that caused the error.
3. When the received parity bit is checked and no error is found, the PER bit in SSR is not set.
4. When the received parity bit is checked and no error is found, reception is considered to have been completed normally and the RDRF bit in SSR is automatically set to 1. If the RIE bit in SCR is enabled at this time, an RXI interrupt is requested.
5. When a normal frame is received, the pin maintains a three-state status when it transmits the error signal.

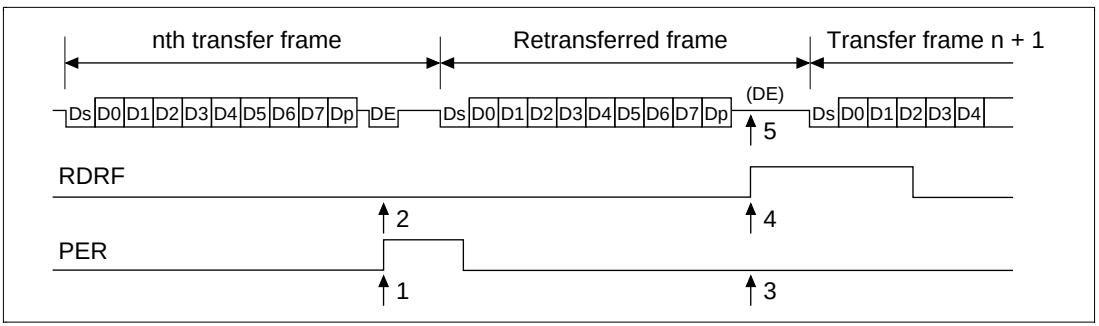



**Figure 14.12 Retransfer in SCI Receive Mode**

**Retransfer by the SCI in Transmit Mode:** Figure 14.13 shows the retransmission operation in the SCI transmit mode.

1. After transmission of one frame is completed, the FER/ERS bit in SSR is set to 1 when a error signal is returned from the receiving side. If the RIE bit in SCR is enabled at this time, an ERI interrupt is requested. Be sure to clear the FER/ERS bit in SSR to 0 before the next parity bit is sampled.
2. The TEND bit in SSR is not set in the frame that received the error signal that indicated the error.
3. The FER/ERS bit in SSR is not set when no error signal is returned from the receiving side.
4. When no error signal is returned from the receiving side, the TEND bit in SSR is set to 1 when the transmission of the frame that includes the retransmission is considered completed. If the TIE bit in SCR is enabled at this time, a TXI interrupt will be requested.

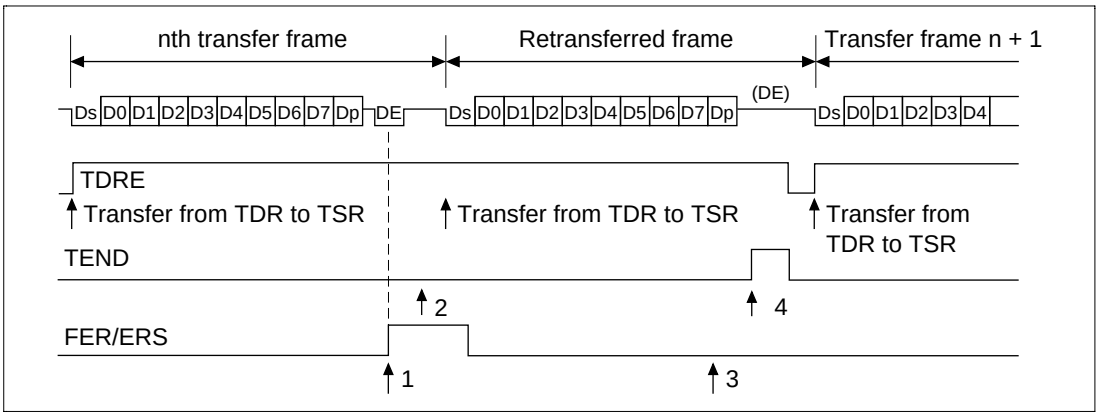



**Figure 14.13 Retransfer in SCI Transmit Mode**

# Section 15 Serial Communication Interface with FIFO (SCIF)

## 15.1 Overview

A two-channel on-chip serial communication interface with FIFO (SCIF) is provided that supports asynchronous serial communication. Sixteen-stage FIFO registers are provided for both transmission and reception that enables efficient, high-speed, continuous communication.

### 15.1.1 Features

The SCIF has the following features:

- Asynchronous serial communication
  - — Serial data communication is executed using an asynchronous system in which synchronization is achieved in character units. The SCIF can communicate with a Universal Asynchronous Receiver/Transmitter (UART), an Asynchronous Communication Interface Adapter (ACIA), or any other communications chip that employs a standard asynchronous serial system. There are eight selectable serial data communication formats.
  - — Data length: 7 or 8 bits
  - — Stop bit length: 1 or 2 bits
  - — Parity: Even/odd/none
  - — Receive error detection: Parity and framing errors
  - — Break detection: Break is detected when the receive data following the generated framing error is the space 0 level, indicating a framing error. It is also detected by reading the RxD level directly from the port data register (PBDR) when a framing error occurs.
- Full duplex communication

  The transmitting and receiving units are independent, so the SCI can transmit and receive simultaneously. Both sections use 16-stage FIFO buffering, so high-speed continuous data transfer is possible in both the transmit and receive directions.
- On-chip baud rate generator with selectable bit rates
- Internal or external transmit/receive clock source: From either baud rate generator (internal) or SCK pin (external)
- Four interrupt sources interrupts

  Transmit-FIFO-data-empty, break, receive-FIFO-data-full, and receive-error interrupts are requested independently.
- When the SCIF is not in use, it can be stopped by halting its clock supply to reduce power consumption.
- Modem control functions ($\overline{\text{RTS}}$ and $\overline{\text{CTS}}$)

- The quantity of data in the transmit and receive FIFO registers and the number of receive errors of the receive data in the receive FIFO register can be ascertained.
- A time-out error (DR) can be detected in receiving.

### 15.1.2 Block Diagram

Figure 15.1 shows a block diagram of the SCIF.

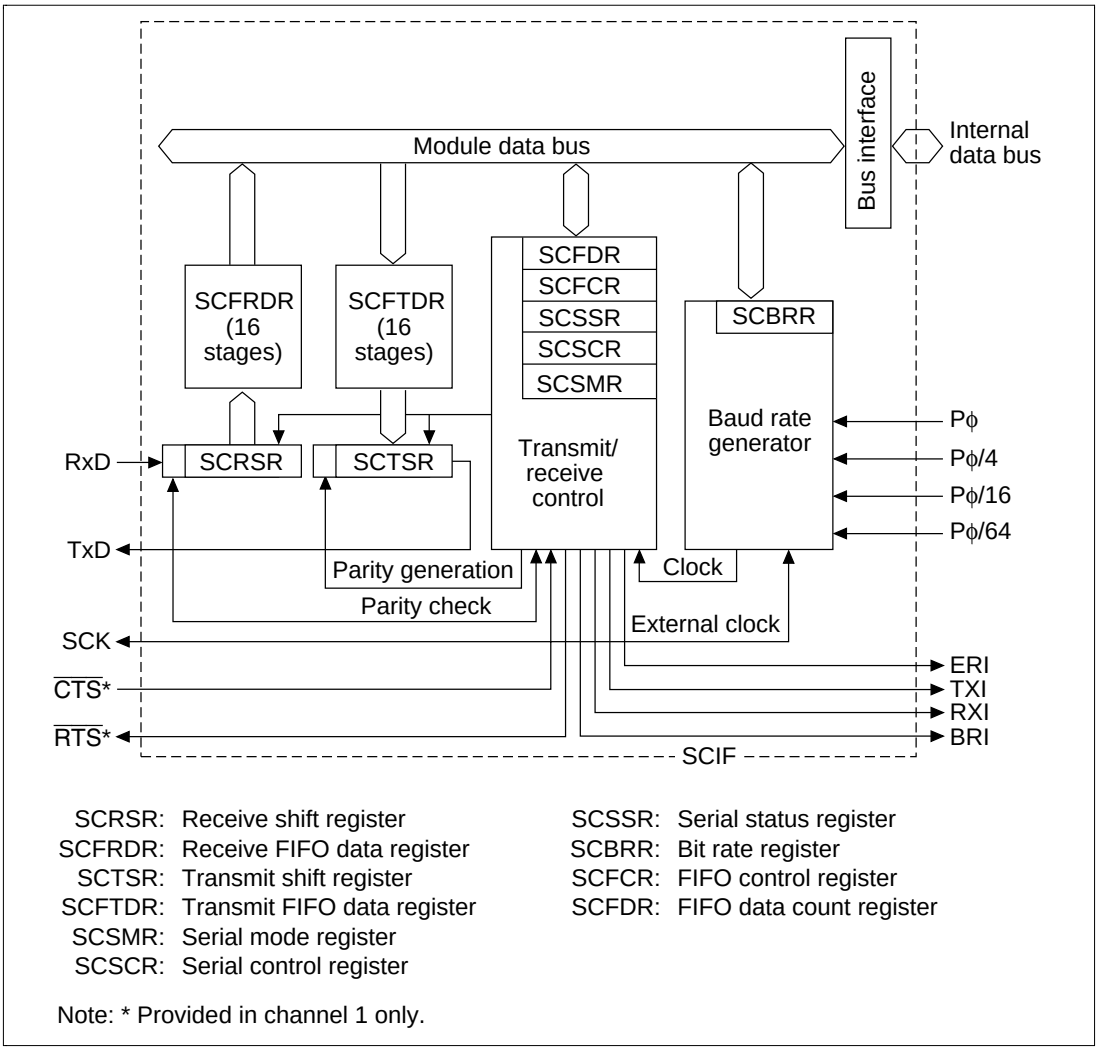



**Figure 15.1 SCIF Block Diagram**

### 15.1.3 Pin Configuration

Table 15.1 shows the pin configuration of the SCIF.

**Table 15.1 Pin Configuration**

| Channel | Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|---|
| 1 | Serial clock pin | SCK1 | Input/output | Clock input/output |
| | Receive data pin | RxD1 | Input | Receive data input |
| | Transmit data pin | TxD1 | Output | Transmit data output |
| | Transmit request pin | $\overline{\text{RTS}}$ | Output | Transmit request |
| | Transmit enable pin | $\overline{\text{CTS}}$ | Input | Transmit enable |
| 2 | Serial clock pin | SCK2 | Input/output | Clock input/output |
| | Receive data pin | RxD2 | Input | Receive data input |
| | Transmit data pin | TxD2 | Output | Transmit data output |

### 15.1.4 Register Configuration

Table 15.2 shows the SCIF internal registers. These registers select the communication mode (asynchronous or synchronous), specify the data format and bit rate, and control the transmitter and receiver sections.

**Table 15.2 Register Configuration**

| Channel | Register Name | Abbre-viation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|---|
| 1 | Serial mode register 1 | SCSMR1 | R/W | H'00 | H'FFFFFCC0 | 8 bits |
| | Bit rate register 1 | SCBRR1 | R/W | H'FF | H'FFFFFCC2 | 8 bits |
| | Serial control register 1 | SCSCR1 | R/W | H'00 | H'FFFFFCC4 | 8 bits |
| | Transmit FIFO data register 1 | SCFTDR1 | W | — | H'FFFFFCC6 | 8 bits |
| | Serial status register 1 | SCSSR1 | R/(W)* | H'0060 | H'FFFFFCC8 | 16 bits |
| | Receive data FIFO register 1 | SCFRDR1 | R | Undefined | H'FFFFFCCA | 8 bits |
| | FIFO control register 1 | SCFCR1 | R/W | H'00 | H'FFFFFCCC | 8 bits |
| | FIFO data count register 1 | SCFDR1 | R | H'0000 | H'FFFFFCCE | 16 bits |
| 2 | Serial mode register 2 | SCSMR2 | R/W | H'00 | H'FFFFFCD0 | 8 bits |
| | Bit rate register 2 | SCBRR2 | R/W | H'FF | H'FFFFFCD2 | 8 bits |
| | Serial control register 2 | SCSCR2 | R/W | H'00 | H'FFFFFCD4 | 8 bits |
| | Transmit FIFO data register 2 | SCFTDR2 | W | — | H'FFFFFCD6 | 8 bits |
| | Serial status register 2 | SCSSR2 | R/(W)* | H'0060 | H'FFFFFCD8 | 16 bits |
| | Receive data FIFO register 2 | SCFRDR2 | R | Undefined | H'FFFFFCDA | 8 bits |
| | FIFO control register 2 | SCFCR2 | R/W | H'00 | H'FFFFFCDC | 8 bits |
| | FIFO data count register 2 | SCFDR2 | R | H'0000 | H'FFFFFCDE | 16 bits |

Note: * Only 0 can be written, to clear the flags.

## 15.2 Register Descriptions

### 15.2.1 Receive Shift Register (SCRSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | — | — | — | — | — | — | — | — |

The receive shift register (SCRSR) is an 8-bit register that receives serial data. Data input at the RxD pin is loaded into SCRSR in the order received, LSB (bit 0) first, converting the data to parallel form. When one byte has been received, it is automatically transferred to SCFRDR, which is a receive FIFO data register. The CPU cannot read or write to SCRSR directly.

### 15.2.2 Receive FIFO Data Register (SCFRDR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | R | R | R | R | R | R | R | R |

The 16-stage receive FIFO data register (SCFRDR) is a FIFO register comprising sixteen 8-bit stages that stores serial receive data. The SCIF completes the reception of one byte of serial data by moving the received data from the receive shift register (SCRSR) into SCFRDR for storage. The SCRSR is then enabled for reception, and consecutive receive operations are performed until the receive FIFO register is filled with 16 bytes of data.

The SCFRDR is a read-only register, and cannot be written by the CPU. If the SCFRDR is read when there is no receive data in the receive FIFO register, an undefined value will be returned. When the receive FIFO register is full of receive data, subsequent serial data is lost.

### 15.2.3 Transmit Shift Register (SCTSR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | — | — | — | — | — | — | — | — |

The transmit shift register (SCTSR) is an 8-bit register that transmits serial data. The SCIF loads transmit data from the transmit FIFO data register (SCFTDR) into SCTSR, then transmits the data serially from the TxD pin, LSB (bit 0) first. After transmitting one data byte, the SCIF automatically loads the next transmit data from SCFTDR into SCTSR and starts transmitting again. The CPU cannot read or write to SCTSR directly.

### 15.2.4 Transmit FIFO Data Register (SCFTDR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| R/W: | W | W | W | W | W | W | W | W |

The transmit FIFO data register (SCFTDR) is a FIFO register comprising sixteen 8-bit stages that stores data for serial transmission. When the SCIF detects that the transmit shift register (SCTSR) is empty, it moves transmit data written in SCFTDR into SCTSR and starts serial transmission. Continuous serial transmission is performed until the SCFTDR is empty. The CPU can always write to SCFTDR.

When the SCFTDR is filled with 16 bytes of transmit data, no further data can be written. If data is written, it will be ignored.

### 15.2.5 Serial Mode Register (SCSMR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R | R/W | R/W |

The serial mode register (SCSMR) is an 8-bit register that specifies the SCIF serial communication format and selects the clock source for the baud rate generator.

The CPU can always read and write to SCSMR. SCSMR is initialized to H'00 by a reset and in standby and module standby modes.

Bit 7—Reserved bit: This bit always reads 0. The write value should always be 0.

Bit 6—Character Length (CHR): Selects 7-bit or 8-bit data in asynchronous mode.

| Bit 6: CHR | Description | |
|---|---|---|
| 0 | 8-bit data | (Initial value) |
| 1 | 7-bit data<br>When seven-bit data is selected, the MSB (bit 7) of the transmit FIFO data register is not transmitted | |

Bit 5—Parity Enable (PE): Selects whether to add a parity bit to transmit data and to check the parity of receive data.

| Bit 5: PE | Description | |
|---|---|---|
| 0 | Parity bit not added or checked | (Initial value) |
| 1 | Parity bit added and checked<br>When PE is set to 1, an even or odd parity bit is added to transmit data, depending on the parity mode (O/$\overline{E}$) setting. Receive data parity is checked according to the even/odd (O/$\overline{E}$) mode setting | |

Bit 4—Parity Mode (O/$\overline{E}$): Selects even or odd parity when parity bits are added and checked. The O/$\overline{E}$ setting is used only when the parity enable bit (PE) is set to 1 to enable parity addition and check. The O/$\overline{E}$ setting is ignored when parity addition and check is disabled.

| Bit 4: O/$\overline{E}$ | Description |
|---|---|
| 0 | Even parity (Initial value)<br>If even parity is selected, the parity bit is added to transmit data to make an even number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an even number of 1s in the received character and parity bit combined |
| 1 | Odd parity<br>If odd parity is selected, the parity bit is added to transmit data to make an odd number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an odd number of 1s in the received character and parity bit combined |

Bit 3—Stop Bit Length (STOP): Selects one or two bits as the stop bit length.

In receiving, only the first stop bit is checked, regardless of the STOP bit setting. When a 2-bit stop bit length is set, if the second stop bit is 1 it is treated as a stop bit, but if it is 0 it is treated as the start bit of the next transmit character.

| Bit 3: STOP | Description |
|---|---|
| 0 | One stop bit (Initial value)<br>In transmitting, a single 1-bit is added at the end of each transmitted character |
| 1 | Two stop bits<br>In transmitting, two 1-bits are added at the end of each transmitted character |

Bit 2—Reserved bit: This bit always reads 0. The write value should always be 0.

Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source of the built-in baud rate generator. Four clock sources are available: Pϕ, Pϕ/4, Pϕ/16, and Pϕ/64. For further information on the clock source, bit rate register settings, and baud rate, see section 15.2.8, Bit Rate Register.

| Bit 1: CKS1 | Bit 0: CKS0 | Description |
|---|---|---|
| 0 | 0 | Pϕ clock (Initial value) |
| | 1 | Pϕ/4 clock |
| 1 | 0 | Pϕ/16 clock |
| | 1 | Pϕ/64 clock |

Note: Pϕ: Peripheral module clock

### 15.2.6 Serial Control Register (SCSCR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R/W | R/W |

The serial control register (SCSCR) operates SCIF transfer operations, selects the serial clock output in asynchronous mode, enables/disables interrupt requests, and selects the transmit/receive clock source. The CPU can always read and write to SCSCR. SCSCR is initialized to H'00 by a reset and in standby and module standby modes.

Bit 7—Transmit Interrupt Enable (TIE): Enables or disables the transmit-FIFO-data-empty interrupt (TXI) requested when the serial transmit data is transferred from the transmit FIFO data register (SCFTDR) to the transmit shift register (SCTSR), when the quantity of data in the transmit FIFO register falls to or below the transmit trigger setting, and when the TDFE flag in the serial status register (SCSSR) is set to1.

| Bit 7: TIE | Description |
|---|---|
| 0 | Transmit-FIFO-data-empty interrupt request (TXI) is disabled (Initial value)<br>The TXI interrupt request can be cleared by writing a quantity of transmit data than the specified transmission trigger number to SCFTDR and by clearing TDFE to 0 after reading 1 from TDFE, or can be cleared by clearing TIE to 0 |
| 1 | Transmit-FIFO-data-empty interrupt request (TXI) is enabled |

Bit 6—Receive Interrupt Enable (RIE): Enables or disables the receive-data-full (RXI) and receive-error (ERI) interrupts requested when serial receive data is transferred from the receive shift register (SCRSR) to the receive FIFO data register (SCFRDR), when the quantity of data in the receive FIFO data register reaches or exceeds the receive trigger setting, and when the RDF flag in SCSSR is set to1.

| Bit 6: RIE | Description |
|---|---|
| 0 | Receive-data-full interrupt (RXI), receive-error interrupt (ERI), and receive break interrupt (BRI) requests are disabled (Initial value)<br>RXI and ERI interrupt requests can be cleared by reading the DR, ER, or RDF flag after it has been set to 1, then clearing the flag to 0, or by clearing RIE to 0. When RDF is set, read 1 from the RDF flag and clear it to 0, after reading the received data from SCFRDR until the quantity of received data is less than the specified receive trigger number |
| 1 | Receive-data-full interrupt (RXI) and receive-error interrupt (ERI) requests are enabled |

Bit 5—Transmit Enable (TE): Enables or disables the SCIF serial transmitter.

| Bit 5: TE | Description |
|---|---|
| 0 | Transmission disabled (Initial value) |
| 1 | Transmission enabled<br>Serial transmission starts after writing transmit data into SCFTDR. Before setting TE to 1, select the communication format in SCSMR and SCFCR, then set FTRST to 1 in SCFCR to reset the FIFO control register (SCFTDR). Then clear TFRST to 0 and set TE to 1. |

Bit 4—Receive Enable (RE): Enables or disables the SCIF serial receiver.

| Bit 4: RE | Description |
|---|---|
| 0 | Reception disabled (Initial value)<br>Clearing RE to 0 does not affect the receive flags (DR, ER, BRK, FER, PER, and RDF). These flags retain their previous values |
| 1 | Reception enabled<br>Serial reception starts when a start bit is detected. Select the receive format in SCSMR before setting RE to 1 |

Bits 3 and 2—Reserved bits: These bits always read 0. The write value should always be 0.

Bits 1 and 0—Clock Enable 1 and 0 (CKE1 and CKE0): These bits select the SCIF clock source and enable or disable clock output from the SCK pin. Depending on the combination of CKE1 and CKE0, the SCK pin can be used for serial clock output or serial clock input.

The CKE0 setting is valid only when the SCIF is operating with the internal clock (CKE1 = 0). The CKE0 setting is ignored when an external clock source is selected (CKE1 = 1). Set CKE1 and CKE0 before selecting the SCIF operating mode in the serial mode register (SCSMR). For further details on selection of the SCIF clock source, see table 15.8 in section 15.3, Operation.

| Bit 1: CKE1 | Bit 0: CKE0 | Description |
|---|---|---|
| 0 | 0 | Internal clock, SCK pin used for input pin (input signal is ignored) (Initial value) |
| | 1 | Internal clock, SCK pin used for clock output*[1] |
| 1 | 0 | External clock, SCK pin used for clock input*[2] |
| | 1 | External clock, SCK pin used for clock input*[2] |

Notes: 1. The output clock frequency is 16 times the bit rate.
2. The input clock frequency is 16 times the bit rate.

### 15.2.7 Serial Status Register (SCSSR)

The serial status register (SCSSR) is a 16-bit register. The upper 8 bits indicate the number of receive errors in the receive FIFO register data, and the lower 8 bits indicate the SCIF operating status.

The CPU can always read and write to SCSSR, but cannot write 1 in the status flags (ER, TEND, TDFE, BRK, OPER, and DR). These flags can be cleared to 0 only if they have first been read (after being set to 1). Bits 3 (FER) and 2 (PER) are read-only bits. SCSSR is initialized to H'0060 by a reset and in standby and module standby modes.

| Lower 8 bits: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | ER | TEND | TDFE | BRK | FER | PER | RDF | DR |
| Initial value: | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R | R | R/(W)* | R/(W)* |

Note: * Only 0 can be written, to clear the flag.

Bit 7—Receive Error (ER): Indicates that a framing error or (when receiving data including parity) a parity error has occurred.*

Note: * Clearing the RE bit to 0 in SCSCR does not affect the ER bit, which retains its previous value. Even if a receive error occurs, the receive data is transferred to SCFRDR and the receive operation is continued. Whether or not the data read from SCFRDR includes a receive error can be detected by the FER and PER bits in SCSSR.

| Bit 7: ER | Description |
|---|---|
| 0 | Receiving is in progress or has ended normally (Initial value)<br>ER is cleared to 0 when the chip is reset or enters standby mode, or when 0 is written after 1 is read from ER |
| 1 | A framing error or a parity error has occurred<br>ER is set to 1 when the stop bit is 0 at the end of reception of one data byte*, or when the total number of 1s in the receive data and in the parity bit does not match the even/odd parity specification by the O/$\overline{E}$ bit in SCSMR |

Note: * In stop-bit mode, only the first stop bit is checked.

Bit 6—Transmit End (TEND): Indicates that when the last bit of a serial character was transmitted, the SCFTDR did not contain valid data, so transmission has ended.

| Bit 6: TEND | Description |
|---|---|
| 0 | Transmission is in progress<br>TEND is cleared to 0 when data is written in SCFTDR |
| 1 | End of transmission (Initial value)<br>TEND is set to 1 when the chip is reset or enters standby mode, the TE bit is 0 in SCSCR, or SCFTDR does not contain transmit data when the last bit of a one-byte serial character is transmitted |

Bit 5—Transmit FIFO Data Register Empty (TDFE): Indicates that data has been transferred from the transmit FIFO data register (SCFTDR) to the transmit shift register (SCTSR), the quantity of data in SCFTDR is equal to or less than the transmission trigger number specified by the TTRG1 and TTRG0 bits in the FIFO control register (SCFCR), and writing of transmit data to SCFTDR is enabled.

| Bit 5: TDFE | Description |
|---|---|
| 0 | The quantity of transmit data written to SCFTDR is greater than the specified transmission trigger number<br>TDFE is cleared to 0 when data exceeding the specified transmission trigger number is written to SCFTDR, software reads TDFE after it has been set to 1, then writes 0 in TDFE |
| 1 | The quantity of transmit data in SCFTDR is equal to or less than the specified transmission trigger number (Initial value)<br>TDFE is set to 1 by a reset or in standby mode, or when the quantity of transmit data in SCFTDR is equal to or less than the specified transmission trigger number as a result of transmission* |

Note: * Since SCFTDR is a 16-byte FIFO register, the maximum quantity of data which can be written when TDFE is 1 is "16 minus the specified transmission trigger number". Even if additional data is written, the data is ignored. The quantity of data in SCFTDR is indicated by the upper 8 bits of SCFDR.

Bit 4—Break Detection (BRK): Indicates that a break signal has been detected in the receive data.

| Bit 4: BRK | Description |
|---|---|
| 0 | No break signal is being received (Initial value)<br>BRK is cleared to 0 when the chip is reset or enters standby mode, or software reads BRK after it has been set to 1, then writes 0 in BRK |
| 1 | The break signal is received*<br>BRK is set to 1 when data associated with a framing error is received and a framing error occurs with the next data comprising all space “0” |

Note: * When a break is detected, transfer of the received data (H'00) to SCFRDR stops. When the break ends and the receive signal goes to mark (1), the transfer of the received data resumes. The received data of a frame in which a break signal is detected is transferred to SCFRDR. After this, however, no received data is transferred until a break ends with the received signal at mark (1), and the next data is received.

Bit 3—Framing Error (FER): Indicates a framing error in the data read from the receive FIFO data register (SCFRDR).

| Bit 3: FER | Description |
|---|---|
| 0 | No framing error occurred in the data read from SCFRDR (Initial value)<br>FER is cleared to 0 when the chip is reset or enters standby mode, or when no framing error is present in the data read from SCFRDR |
| 1 | A framing error occurred in the data read from SCFRDR<br>FER is set to 1 when a framing error is present in the data read from SCFRDR |

Bit 2—Parity Error (PER): Indicates a parity error in the data read from the receive FIFO data register (SCFRDR).

| Bit 2: PER | Description |
|---|---|
| 0 | No parity error occurred in the data read from SCFRDR (Initial value)<br>PER is cleared to 0 when the chip is reset or enters standby mode, or when no parity error is present in the data read from SCFRDR |
| 1 | A framing error occurred in the data read from SCFRDR<br>PER is set to 1 when a parity error is present in the data read from SCFRDR |

Bit 1—Receive FIFO Data Register Full (RDF): Indicates that received data has been transferred to the receive FIFO data register (SCFRDR), and the quantity of data in SCFRDR reaches or exceeds the receive trigger number specified by the RTRG1 and RTRG0 bits in FIFO control register (SCFCR).

| Bit 1: RDF | Description |
|---|---|
| 0 | The quantity of receive data written to SCFRDR is less than the specified number of receive trigger number (Initial value)<br>RDF is cleared to 0 by a reset or enters standby mode, or cleared when data in SCFRDR is read until the quantity of receive data in SCFRDR is less than the specified receive trigger number, and 1 is read from RDF, and 0 is then written |
| 1 | The quantity of receive data in SCFRDR is equal to or greater than the specified receive trigger number<br>RDF is set to 1 when a quantity of receive data which is equal to or greater than the specified receive trigger number is stored in SCFRDR* |

Note: * Since SCFRDR is a 16-byte FIFO register, the maximum quantity of data which can be read when RDF is 1 is the specified receive trigger number. If reading is attempted after all data in SCFRDR has been read, the data will be undefined. The quantity of receive data in SCFRDR is indicated by the lower 8 bits of SCFDR.

Bit 0—Receive Data Ready (DR): Indicates that the receive FIFO data register (SCFRDR) holds a quantity of data less than the specified receive trigger number (where receive data quantity > 0), and that the next data has not yet been received after the elapse of 15 etu since the last stop bit.

| Bit 0: DR | Description |
|---|---|
| 0 | Receiving is in progress, or no received data remains in SCFRDR after reception has ended normally (Initial value)<br>DR is cleared to 0 when the chip is reset or enters standby mode, or when software reads DR after it has been set to 1, then writes 0 in DR |
| 1 | Next receive data has not arrived<br>DR is set to 1 when the quantity of receive data in SCFRDR is less than the receive trigger number, and the next data has not yet been received 15 etu* after the last stop bit |

Note: * This is equivalent to 1.5 frames with the 8-bit, 1-stop-bit format. (etu: element time unit)

| Upper 8 bits: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PER3 | PER2 | PER1 | PER0 | FER3 | FER2 | FER1 | FER0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

Bits 15 to 12—Number of Parity Errors (PER): Indicates the number of receive data bytes in which a parity error occurred that are stored in the receive FIFO data register (SCFRDR). The value indicated by bits 15 to 12 represents the number of SCFRDR parity errors.

Bits 11 to 8—Number of Framing Errors (FER): Indicates the number of receive data bytes in which a framing error occurred that are stored in the receive FIFO data register (SCFRDR). The value indicated by bits 11 to 8 represents the number of SCFRDR framing errors.

### 15.2.8 Bit Rate Register (SCBRR)

The bit rate register (SCBRR) is an 8-bit register that, together with the baud rate generator clock source selected by the CKS1 and CKS0 bits in the serial mode register (SCSMR), determines the serial transmit/receive bit rate.

The CPU can always read and write to SCBRR. SCBRR is initialized to H'FF by a reset and in module standby and standby modes. Each channel has independent baud rate generator control, so different values can be set in the two channels.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The SCBRR setting is calculated as follows:

Asynchronous mode:

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

B: Bit rate (bit/s)
N: SCBRR setting for baud rate generator ($0 \leq N \leq 255$)
P$\phi$: Operating frequency for supporting modules (MHz)
n: Baud rate generator clock source (n = 0, 1, 2, 3) (for the clock sources and values of n, see table 15.3.)

**Table 15.3 SCSMR Settings**

| | | SCSMR Settings | |
|---|---|---|---|
| **n** | **Clock Source** | **CKS1** | **CKS0** |
| 0 | Pϕ | 0 | 0 |
| 1 | Pϕ/4 | 0 | 1 |
| 2 | Pϕ/16 | 1 | 0 |
| 3 | Pϕ/64 | 1 | 1 |

Note: The the bit rate error is given by the following equation:

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

Table 15.4 lists examples of SCBRR settings.

**Table 15.4 Bit Rates and SCBRR Settings**

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

| Bit Rate (bits/s) | Pϕ (MHz) 3 | | | 3.6864 | | | 4 | | |
|---|---|---|---|---|---|---|---|---|---|
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 1 | 212 | 0.03 | 2 | 64 | 0.70 | 2 | 70 | 0.03 |
| 150 | 1 | 155 | 0.16 | 1 | 191 | 0.00 | 1 | 207 | 0.16 |
| 300 | 1 | 77 | 0.16 | 1 | 95 | 0.00 | 1 | 103 | 0.16 |
| 600 | 0 | 155 | 0.16 | 0 | 191 | 0.00 | 0 | 207 | 0.16 |
| 1200 | 0 | 77 | 0.16 | 0 | 95 | 0.00 | 0 | 103 | 0.16 |
| 2400 | 0 | 38 | 0.16 | 0 | 47 | 0.00 | 0 | 51 | 0.16 |
| 4800 | 0 | 19 | –2.34 | 0 | 23 | 0.00 | 0 | 25 | 0.16 |
| 9600 | 0 | 9 | –2.34 | 0 | 11 | 0.00 | 0 | 12 | 0.16 |
| 19200 | 0 | 4 | –2.34 | 0 | 5 | 0.00 | 0 | 6 | –6.99 |
| 31250 | 0 | 2 | 0.00 | 0 | 3 | –7.84 | 0 | 3 | 0.00 |
| 38400 | — | — | — | 0 | 2 | 0.00 | 0 | 2 | 8.51 |

**Table 15.4 Bit Rates and SCBRR Settings (cont)**

| Bit Rate (bits/s) | Pϕ (MHz) 4.9152 n | N | Error (%) | 5 n | N | Error (%) | 6 n | N | Error (%) |
|---|---|---|---|---|---|---|---|---|---|
| 110 | 2 | 86 | 0.31 | 2 | 88 | –0.25 | 2 | 106 | –0.44 |
| 150 | 1 | 255 | 0.00 | 2 | 64 | 0.16 | 2 | 77 | 0.16 |
| 300 | 1 | 127 | 0.00 | 1 | 129 | 0.16 | 1 | 155 | 0.16 |
| 600 | 0 | 255 | 0.00 | 1 | 64 | 0.16 | 1 | 77 | 0.16 |
| 1200 | 0 | 127 | 0.00 | 0 | 129 | 0.16 | 0 | 155 | 0.16 |
| 2400 | 0 | 63 | 0.00 | 0 | 64 | 0.16 | 0 | 77 | 0.16 |
| 4800 | 0 | 31 | 0.00 | 0 | 32 | –1.36 | 0 | 38 | 0.16 |
| 9600 | 0 | 15 | 0.00 | 0 | 15 | 1.73 | 0 | 19 | –2.34 |
| 19200 | 0 | 7 | 0.00 | 0 | 7 | 1.73 | 0 | 9 | –2.34 |
| 31250 | 0 | 4 | –1.70 | 0 | 4 | 0.00 | 0 | 5 | 0.00 |
| 38400 | 0 | 3 | 0.00 | 0 | 3 | 1.73 | 0 | 4 | –2.34 |

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

**Table 15.4 Bit Rates and SCBRR Settings (cont)**

| Bit Rate (bits/s) | Pϕ (MHz) 9.8304 n | N | Error (%) | 10 n | N | Error (%) | 12 n | N | Error (%) | 12.288 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 |

| Bit Rate (bits/s) | Pϕ (MHz) 14.7456 n | N | Error (%) | 16 n | N | Error (%) | 19.6608 n | N | Error (%) | 20 n | N | Error (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 110 | 3 | 64 | 0.70 | 3 | 70 | 0.03 | 3 | 86 | 0.31 | 3 | 88 | –0.25 |
| 150 | 2 | 191 | 0.00 | 2 | 207 | 0.16 | 2 | 255 | 0.00 | 3 | 64 | 0.16 |
| 300 | 2 | 95 | 0.00 | 2 | 103 | 0.16 | 2 | 127 | 0.00 | 2 | 129 | 0.16 |
| 600 | 1 | 191 | 0.00 | 1 | 207 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 |
| 1200 | 1 | 95 | 0.00 | 1 | 103 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 |
| 2400 | 0 | 191 | 0.00 | 0 | 207 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 |
| 4800 | 0 | 95 | 0.00 | 0 | 103 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 |
| 9600 | 0 | 47 | 0.00 | 0 | 51 | 0.16 | 0 | 63 | 0.00 | 0 | 64 | 0.16 |
| 19200 | 0 | 23 | 0.00 | 0 | 25 | 0.16 | 0 | 31 | 0.00 | 0 | 32 | –1.36 |
| 31250 | 0 | 14 | -1.70 | 0 | 15 | 0.00 | 0 | 19 | –1.70 | 0 | 19 | 0.00 |
| 38400 | 0 | 11 | 0.00 | 0 | 12 | 0.16 | 0 | 15 | 0.00 | 0 | 15 | 1.73 |
| 115200 | 0 | 3 | 0.00 | 0 | 3 | 8.51 | 0 | 4 | 6.67 | 0 | 4 | 8.51 |
| 500000 | 0 | 0 | -7.84 | 0 | 0 | 0.00 | 0 | 0 | 22.9 | 0 | 0 | 25.0 |

**Table 15.4 Bit Rates and SCBRR Settings (cont)**

| | Pϕ (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 24 | | | 24.576 | | | 28.7 | | | 30 | | |
| Bit Rate (bits/s) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 106 | –0.44 | 3 | 108 | 0.08 | 3 | 126 | 0.31 | 3 | 132 | 0.13 |
| 150 | 3 | 77 | 0.16 | 3 | 79 | 0.00 | 3 | 92 | 0.46 | 3 | 97 | –0.35 |
| 300 | 2 | 155 | 0.16 | 2 | 159 | 0.00 | 2 | 186 | –0.08 | 2 | 194 | 0.16 |
| 600 | 2 | 77 | 0.16 | 2 | 79 | 0.00 | 2 | 92 | 0.46 | 2 | 97 | –0.35 |
| 1200 | 1 | 155 | 0.16 | 1 | 159 | 0.00 | 1 | 186 | –0.08 | 1 | 194 | 0.16 |
| 2400 | 1 | 77 | 0.16 | 1 | 79 | 0.00 | 1 | 92 | 0.46 | 1 | 97 | –0.35 |
| 4800 | 0 | 155 | 0.16 | 0 | 159 | 0.00 | 0 | 186 | –0.08 | 0 | 194 | 0.16 |
| 9600 | 0 | 77 | 0.16 | 0 | 79 | 0.00 | 0 | 92 | 0.46 | 0 | 97 | –0.35 |
| 19200 | 0 | 38 | 0.16 | 0 | 39 | 0.00 | 0 | 46 | –0.61 | 0 | 48 | –0.35 |
| 31250 | 0 | 23 | 0.00 | 0 | 24 | –1.70 | 0 | 28 | –1.03 | 0 | 29 | 0.00 |
| 38400 | 0 | 19 | –2.34 | 0 | 19 | 0.00 | 0 | 22 | 1.55 | 0 | 23 | 1.73 |
| 115200 | 0 | 6 | –6.99 | 0 | 6 | –4.76 | 0 | 7 | –2.68 | 0 | 7 | 1.73 |
| 500000 | 0 | 1 | –25.0 | 0 | 1 | –23.2 | 0 | 1 | –10.3 | 0 | 1 | –6.25 |

Note: Settings with an error of 1% or less are recommended.

—: Setting available, but error occurs

*: Continuous transmission/reception not possible

Table 15.5 indicates the maximum bit rates in asynchronous mode when the baud rate generator is being used. Table 15.6 list the maximum rates for external clock input.

**Table 15.5 Maximum Bit Rates for Various Frequencies with Baud Rate Generator (Asynchronous Mode)**

| | | Settings | |
|---|---|---|---|
| **Pϕ (MHz)** | **Maximum Bit Rate (bits/s)** | **n** | **N** |
| 2 | 62500 | 0 | 0 |
| 2.097152 | 65536 | 0 | 0 |
| 2.4576 | 76800 | 0 | 0 |
| 3 | 93750 | 0 | 0 |
| 3.6864 | 115200 | 0 | 0 |
| 4 | 125000 | 0 | 0 |
| 4.9152 | 153600 | 0 | 0 |
| 8 | 250000 | 0 | 0 |
| 9.8304 | 307200 | 0 | 0 |
| 12 | 375000 | 0 | 0 |
| 14.7456 | 460800 | 0 | 0 |
| 16 | 500000 | 0 | 0 |
| 19.6608 | 614400 | 0 | 0 |
| 20 | 625000 | 0 | 0 |
| 24 | 750000 | 0 | 0 |
| 24.576 | 768000 | 0 | 0 |
| 28.7 | 896875 | 0 | 0 |
| 30 | 937500 | 0 | 0 |

**Table 15.6 Maximum Bit Rates during External Clock Input (Asynchronous Mode)**

| Pϕ (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bits/s) |
|---|---|---|
| 2 | 0.5000 | 31250 |
| 2.097152 | 0.5243 | 32768 |
| 2.4576 | 0.6144 | 38400 |
| 3 | 0.7500 | 46875 |
| 3.6864 | 0.9216 | 57600 |
| 4 | 1.0000 | 62500 |
| 4.9152 | 1.2288 | 76800 |
| 8 | 2.0000 | 125000 |
| 9.8304 | 2.4576 | 153600 |
| 12 | 3.0000 | 187500 |
| 14.7456 | 3.6864 | 230400 |
| 16 | 4.0000 | 250000 |
| 19.6608 | 4.9152 | 307200 |
| 20 | 5.0000 | 312500 |
| 24 | 6.0000 | 375000 |
| 24.576 | 6.1440 | 384000 |
| 28.7 | 7.1750 | 448436 |
| 30 | 7.5000 | 468750 |

### 15.2.9 FIFO Control Register (SCFCR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | RTRG1 | RTRG0 | TTRG1 | TTRG0 | MCE | TFRST | RFRST | LOOP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The FIFO control register (SCFCR) resets the quantity of data in the transmit and receive FIFO data registers, sets the trigger data number and contains the enable bit for loop-back testing. SCFCR can read and written by the CPU at all times. It is initialized to H'00 by a reset, by the module standby function, and in standby mode.

Bits 7 and 6—Receive FIFO Data Trigger (RTRG1, RTRG0): These bits set the quantity of receive data which sets the receive data full (RDF) flag in the serial status register (SCSSR). The RDF flag is set when the quantity of receive data stored in the receive FIFO data register (SCFRDR) exceeds the trigger number as shown below.

| Bit 7: RTRG1 | Bit 6: RTRG0 | Receive Trigger Number | |
|---|---|---|---|
| 0 | 0 | 1 | (Initial value) |
| | 1 | 4 | |
| 1 | 0 | 8 | |
| | 1 | 14 | |

Bits 5 and 4—Transmit FIFO Data Trigger (TTRG1, TTRG0): These bits set the quantity of remaining transmit data which sets the transmit FIFO data register empty (TDFE) flag in the serial status register (SCSSR). The TDFE flag when the quantity of transmit data in the transmit FIFO data register (SCFTDR) falls below the trigger number specified by these bits, as shown below.

| Bit 5: TTRG1 | Bit 4: TTRG0 | Transmit Trigger Number | |
|---|---|---|---|
| 0 | 0 | 8 (8)* | (Initial value) |
| | 1 | 4 (12) | |
| 1 | 0 | 2 (14) | |
| | 1 | 1 (15) | |

Note: * Values in parentheses indicate the number of empty bits in the SCFTDR register when the TDFE flag is set to 1.

Bit 3—Modem Control Enable (MCE): Enables modem control signals $\overline{\text{CTS}}$ and $\overline{\text{RTS}}$.

| Bit 3: MCE | Description | |
|---|---|---|
| 0 | Modem signals disabled* | (Initial value) |
| 1 | Modem signals enabled | |

Note: * $\overline{\text{CTS}}$ is fixed at active-0 regardless of the input value, and $\overline{\text{RTS}}$ is also fixed at 0.

Bit 2—Transmit FIFO Data Register Reset (TFRST): Disables the transmit data in the transmit FIFO data register and resets the data to the empty state.

| Bit 2: TFRST | Description | |
|---|---|---|
| 0 | Reset operation disabled* | (Initial value) |
| 1 | Reset operation enabled | |

Note: Resetting operates in resets and in standby mode.

Bit 1—Receive FIFO Data Register Reset (RFRST): Disables the receive data in the receive FIFO data register and resets the data to the empty state.

| Bit 1: RFRST | Description | |
|---|---|---|
| 0 | Reset operation disabled* | (Initial value) |
| 1 | Reset operation enabled | |

Note: Resetting operates in resets and in standby mode.

Bit 0—Loop-Back Test (LOOP): Internally connects the transmit output pin (TXD) and receive input pin (RXD) and enables loop-back testing.

| Bit 0: LOOP | Description | |
|---|---|---|
| 0 | Loop-back testing disabled | (Initial value) |
| 1 | Loop-back testing enabled | |

### 15.2.10 FIFO Data Count Register (SCFDR)

SCFDR is a 16-bit register which indicates the quantity of data stored in the transmit FIFO data register (SCFTDR) and the receive FIFO data register (SCFRDR). It indicates the quantity of transmit data in SCFTDR in the upper eight bits, and the quantity of receive data in SCFRDR in the lower eight bits. SCFDR can always be read by the CPU.

| Lower 8 Bits: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | R4 | R3 | R2 | R1 | R0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

The lower 8 bits of SCFDR indicate the quantity of receive data stored in SCFRDR. H'00 means that there is no receive data, and H'10 means that the SCFRDR is full of receive data.

| Upper 8 Bits: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | T4 | T3 | T2 | T1 | T0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

The upper 8 bits of SCFDR indicate the quantity of non-transmitted data stored in SCFTDR. H'00 means that there is no transmit data, and H'10 means that the SCFTDR is full of transmit data.

## 15.3 Operation

### 15.3.1 Overview

For serial communication, the SCIF uses an asynchronous mode in which characters are synchronized individually. Refer to section 13.3.2, Serial Communication Interface, for details of asynchronous mode operation. The SCIF has a 16-stage FIFO buffer for both transmit and receive operations, reducing CPU overhead and enabling continuous high-speed communication. It also uses $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ modem control signals. The transmission format is selected in the serial mode register (SCSMR), as shown in table 15.7. The SCIF clock source is selected by the combination of the CKE1 and CKE0 bits in the serial control register (SCSCR), as shown in table 15.8.

- Data length is selectable: seven or eight bits.
- Choice of parity addition and addition of 1 or 2 stop bits. The combination of the preceding selections constitutes the communication format and character length.
- In receiving, it is possible to detect framing errors (FER), parity errors (PER), receive FIFO data full, receive data ready, and breaks.
- In transmitting, it is possible to detect transmit FIFO data empty.

- The number of stored data bytes is displayed for both the transmit and receive FIFO registers.
- An internal or external clock can be selected as the SCIF clock source.
  - When an internal clock is selected, the SCIF operates using the built-in baud rate generator, and can output a serial clock signal with a frequency 16 times the bit rate.
  - When an external clock is selected, the external clock input must have a frequency of 16 times the bit rate. (The built-in baud rate generator is not used.)

**Table 15.7 SMR Settings and SCIF Transfer Format Selection**

| | SCSMR Settings | | | | | |
|---|---|---|---|---|---|---|
| **Mode** | **Bit 6 CHR** | **Bit 5 PE** | **Bit 3 STOP** | **Data Length** | **Parity Bit** | **SCIF Communication Format Stop Bit Length** |
| Asynchronous | 0 | 0 | 0 | 8-bit | Not set | 1 bit |
| | | | 1 | | | 2 bits |
| | | 1 | 0 | | Set | 1 bit |
| | | | 1 | | | 2 bits |
| | 1 | 0 | 0 | 7-bit | Not set | 1 bit |
| | | | 1 | | | 2 bits |
| | | 1 | 0 | | Set | 1 bit |
| | | | 1 | | | 2 bits |

**Table 15.8 SCSMR and SCSCR Settings and SCIF Clock Source Selection**

| | SCSCR Settings | | SCIF Transmit/Receive Clock | |
|---|---|---|---|---|
| **Mode** | **Bit 1 CKE1** | **Bit 0 CKE0** | **Clock Source** | **SCK Pin Function** |
| Asynchronous | 0 | 0 | Internal | SCIF does not use the SCK pin |
| | | 1 | | Outputs a clock with a frequency 16 times the bit rate |
| | 1 | 0 | External | Inputs a clock with frequency 16 times the bit rate |
| | | 1 | | |

### 15.3.2 Serial Operation

**Transmit/Receive Formats:** Table 15.9 lists the eight communication formats that can be selected. The format is selected by settings in the serial mode register (SCSMR).

**Table 15.9 Serial Communication Formats**

| SCSMR Bits | | | Serial Transmit/Receive Format and Frame Length | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHR | PE | STOP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | S | 8-bit data | | | | | | | | STOP | | |
| | | 1 | S | 8-bit data | | | | | | | | STOP | STOP | |
| | 1 | 0 | S | 8-bit data | | | | | | | | P | STOP | |
| | | 1 | S | 8-bit data | | | | | | | | P | STOP | STOP |
| 1 | 0 | 0 | S | 7-bit data | | | | | | | STOP | | | |
| | | 1 | S | 7-bit data | | | | | | | STOP | STOP | | |
| | 1 | 0 | S | 7-bit data | | | | | | | P | STOP | | |
| | | 1 | S | 7-bit data | | | | | | | P | STOP | STOP | |

Notes: S: Start bit
STOP: Stop bit
P: Parity bit

**Clock:** An internal clock generated by the built-in baud rate generator or an external clock input from the SCK pin can be selected as the SCIF transmit/receive clock. The clock source is selected by the bits CKE1 and CKE0 in the serial control register (SCSCR) (table 15.8).

When an external clock is input at the SCK pin, it must have a frequency equal to 16 times the desired bit rate.

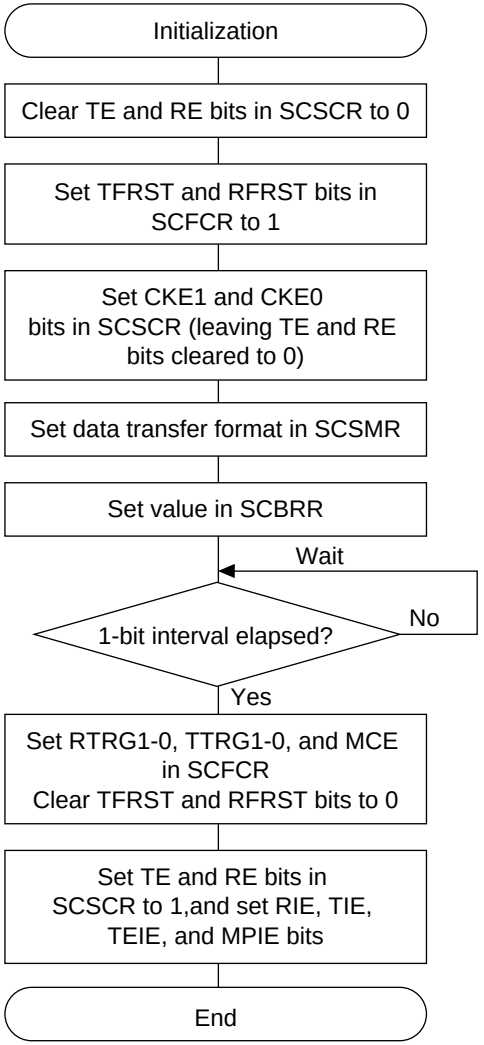
When the SCIF operates on an internal clock, it can output a clock signal at the SCK pin. The frequency of this output clock is 16 times the bit rate.

**Data Transfer Operations (SCIF Initialization):** Before transmitting or receiving data, clear the TE and RE bits to 0 in the serial control register (SCSCR), then initialize the SCIF as follows.

When changing the communication format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 initializes the transmit shift register (SCTSR). Clearing TE and RE to 0, however, does not initialize the serial status register (SCSSR), transmit FIFO data register (SCFTDR), or receive FIFO data register (SCFRDR), which retain their previous contents. Clear TE to 0 after all transmit data are transmitted and the TEND flag in SCSSR is set. The transmitting data enters the high impedance state after clearing to 0 although the bit can be cleared to 0 in transmitting. Set the TFRST bit in SCFCR to 1 and reset SCFTDR before TE is set again to start transmission.

When an external clock is used, the clock should not be stopped during initialization or subsequent operation. SCIF operation becomes unreliable if the clock is stopped.

Figure 15.2 shows a sample flowchart for initializing the SCIF. The procedure for initializing the SCIF is shown in the figure.




1. Set the clock selection in SCSCR.
   Be sure to clear bits RIE TIE, TE, and RE to 0.
   When clock output is selected, the clock is output immediately after SCSCR settings are made.

2. Set the data transfer format in SCSMR.

3. Write a value corresponding to the bit rate into the bit rate register (SCBRR).
   (Not necessary if an external clock is used.)

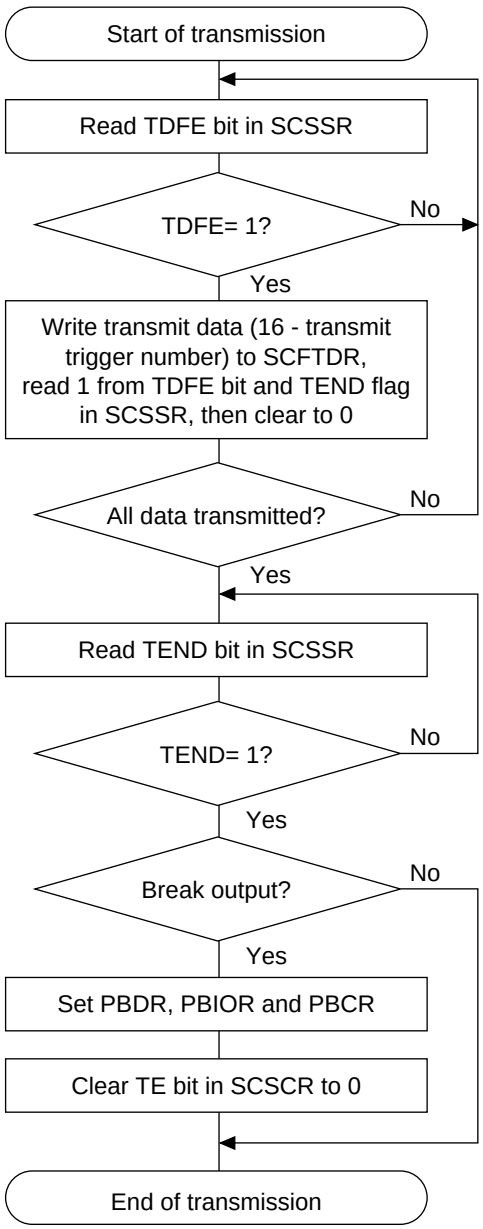
4. Wait at least one bit interval, then set the TE bit or RE bit in SCSCR to 1. Also set the RIE and TIE bits.
   Setting the TE and RE bits enables the TxD and RxD pins to be used. When transmitting, the SCIF will go to the mark state; when receiving, it will go to the idle state, waiting for a start bit.

**Figure 15.2 Sample Flowchart for SCIF Initialization**

- Serial data transmission

Figure 15.3 shows a sample flowchart for serial transmission.

Use the following procedure for serial data transmission after enabling the SCIF for transmission.




1. SCIF status check and transmit data write: Read serial status register (SCSSR) and check that the TDFE flag is set to 1, then write transmit data to the transmit FIFO data register (SCFTDR), read 1 from the TDFE and TEND flags, then clear these flags to 0.
   The number of transmit data bytes that can be written is 16 – transmit trigger number.

2. Serial transmission continuation procedure: To continue serial transmission, read 1 from the TDFE flag to confirm that writing is possible, then write data to SCFTDR, and then clear the TDFE flag to 0.

3. Break output at the end of serial transmission: To output a break in serial transmission, clear port register PBDR corresponding to the TxD pin to 0, set 1 in the PBIOR register, and set the PBCR register to general port use. Then clear the TE bit to 0 in the serial control register (SCSCR).

   In steps 1 and 2, it is possible to ascertain the number of data bytes that can be written from the number of transmit data bytes in SCFTDR indicated by the upper 8 bits of the FIFO data register (SCFDR).

**Figure 15.3 Sample Flowchart for Serial Transmission**

In serial transmission, the SCIF operates as described below.

1. When data is written into the transmit FIFO data register (SCFTDR), the SCIF transfers the data from SCFTDR to the transmit shift register (SCTSR) and starts transmitting. Confirm that the TDFE flag in the serial status register (SCSSR) is set to 1 before writing transmit data to SCFTDR. The number of data bytes that can be written is (16 – transmit trigger number).

2. When data is transferred from SCFTDR to SCTSR and transmission is started, consecutive transmit operations are performed until there is no transmit data left in SCFTDR. When the number of transmit data bytes in SCFTDR falls below the transmit trigger number set in the FIFO control register (SCFCR), the TDFE flag is set. If the TIE bit in the serial control register (SCSCR) is set to 1 at this time, a transmit-FIFO-data-empty interrupt (TXI) request is generated.

   The serial transmit data is sent from the TxD pin in the following order.

   a. Start bit: One 0-bit is output.
   b. Transmit data: 8-bit or 7-bit data is output in LSB-first order.
   c. Parity bit: One parity bit (even or odd parity) is output. (A format in which a parity bit is not output can also be selected.)
   d. Stop bit(s): One or two 1-bits (stop bits) are output.
   e. Mark state: 1 is output continuously until the start bit that starts the next transmission is sent.

3. The SCIF checks the SCFTDR transmit data at the timing for sending the stop bit. If data is present, the data is transferred from SCFTDR to SCTSR, the stop bit is sent, and then serial transmission of the next frame is started.

   If there is no transmit data, the TEND flag in SCSSR is set to 1, the stop bit is sent, and then the line goes to the mark state in which 1 is output continuously.

   Figure 15.4 shows an example of the operation for transmission.

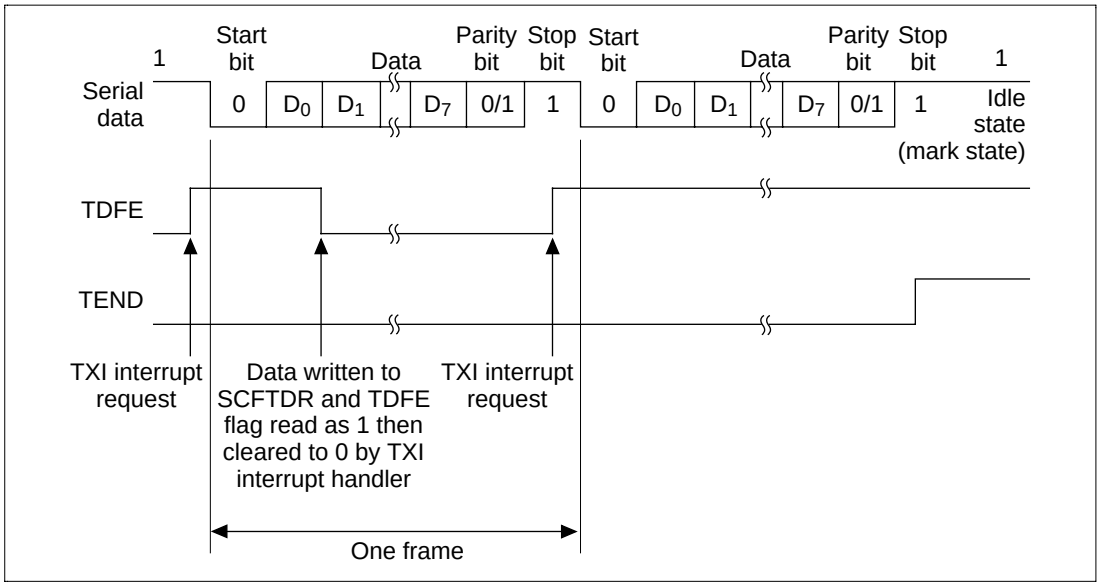



**Figure 15.4 Example of Transmit Operation (Example with 8-Bit Data, Parity, One Stop Bit)**

4. When modem control is enabled, transmission can be stopped and restarted in accordance with the $\overline{\text{CTS}}$ input value. When $\overline{\text{CTS}}$ is 1, if transmission is in progress, the line goes to the mark state after transmission of one frame. When $\overline{\text{CTS}}$ is 0, the next transmit data is output starting from the start bit.

   Figure 15.5 shows an example of the operation when modem control is used.

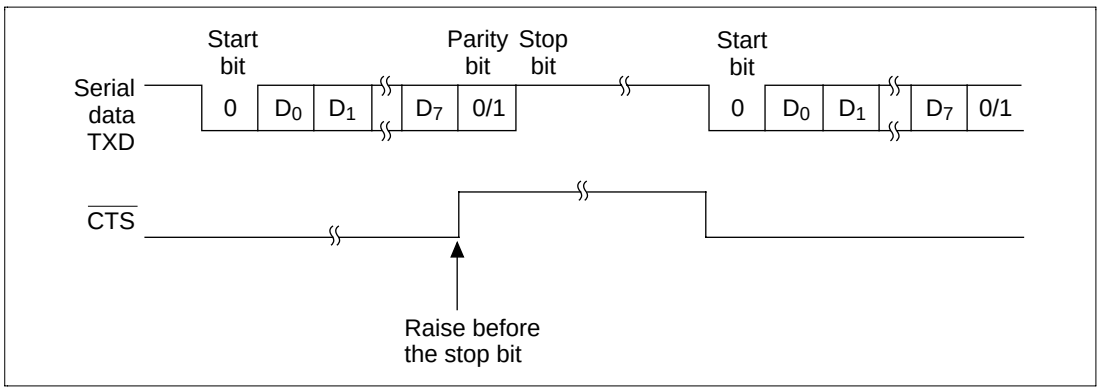
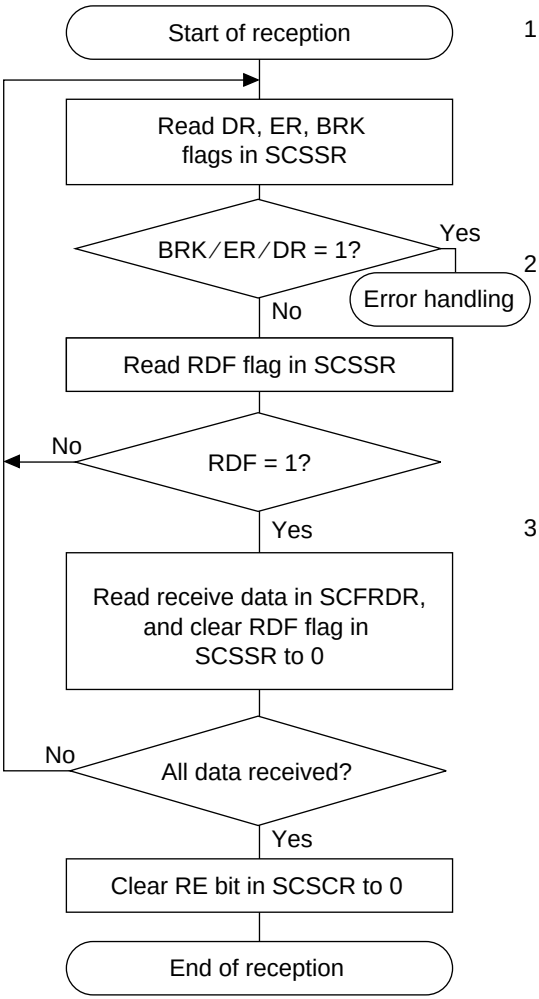



**Figure 15.5 Example of Operation Using Modem Control ($\overline{\text{CTS}}$)**

- Serial data reception

Figure 15.6 shows a sample flowchart for serial reception.

Use the following procedure for serial data reception after enabling the SCIF for reception.

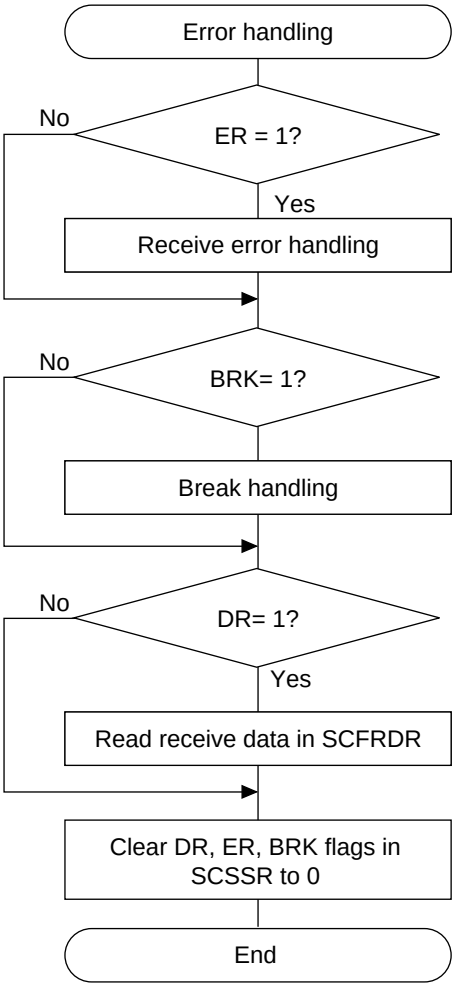



1. Receive error handling and break detection: Read the DR, ER, and BRK flags in SCSSR to identify any error, perform the appropriate error handling, then clear the DR, ER, and BRK flags to 0. In the case of a framing error, a break can also be detected by reading the value of the RxD pin.

2. SCIF status check and receive data read : Read the serial status register (SCSSR) and check that RDF = 1, then read the receive data in the receive FIFO data register (SCFRDR), read 1 from the RDF flag, and then clear the RDF flag to 0. The transition of the RDF flag from 0 to 1 can be identified by an RXI interrupt.

3. Serial reception continuation procedure: To continue serial reception, read at least the number of receive data bytes set as the receive trigger from SCFRDR, read 1 from the RDF flag, then clear the RDF flag to 0. The number of receive data bytes in SCFRDR can be ascertained by reading the lower bits of SCFDR.

**Figure 15.6 Sample Flowchart for Serial Reception**




1. Whether a framing error or parity error has occurred in the receive data read from SCFRDR can be ascertained from the FER and PER bits in SCSSR.

2. When a break signal is received, receive data is not transferred to SCFRDR while the BRK flag is set. However, note that the last data in SCFRDR is H'00 and the break data in which a framing error occurred is stored.

**Figure 15.6 Sample Flowchart for Serial Reception (cont)**

In serial reception, the SCIF operates as described below.

1. The SCIF monitors the transmission line, and if a 0 start bit is detected, performs internal synchronization and starts reception.
2. The received data is stored in SCRSR in LSB-to-MSB order.
3. The parity bit and stop bit are received.

   After receiving these bits, the SCIF carries out the following checks.

   a. Stop bit check: The SCIF checks whether the stop bit is 1. If there are two stop bits, only the first is checked.
   b. The SCIF checks whether receive data can be transferred from the receive shift register (SCRSR) to SCFRDR.
   c. Break check: The SCIF checks that the BRK flag is 0, indicating that the break state is not set.

   If all the above checks are passed, the receive data is stored in SCFRDR.

   Note: Reception is not suspended when a receive error occurs.

4. If the RIE bit in SCSCR is set to 1 when the RDF or DR flag changes to 1, a receive-FIFO-data-full interrupt (RXI) request is generated.

   If the RIE bit in SCSCR is set to 1 when the ER flag changes to 1, a receive-error interrupt (ERI) request is generated.

   If the RIE bit in SCSCR is set to 1 when the BRK flag changes to 1, a break reception interrupt (BRI) request is generated.

   Figure 15.7 shows an example of the operation for reception.

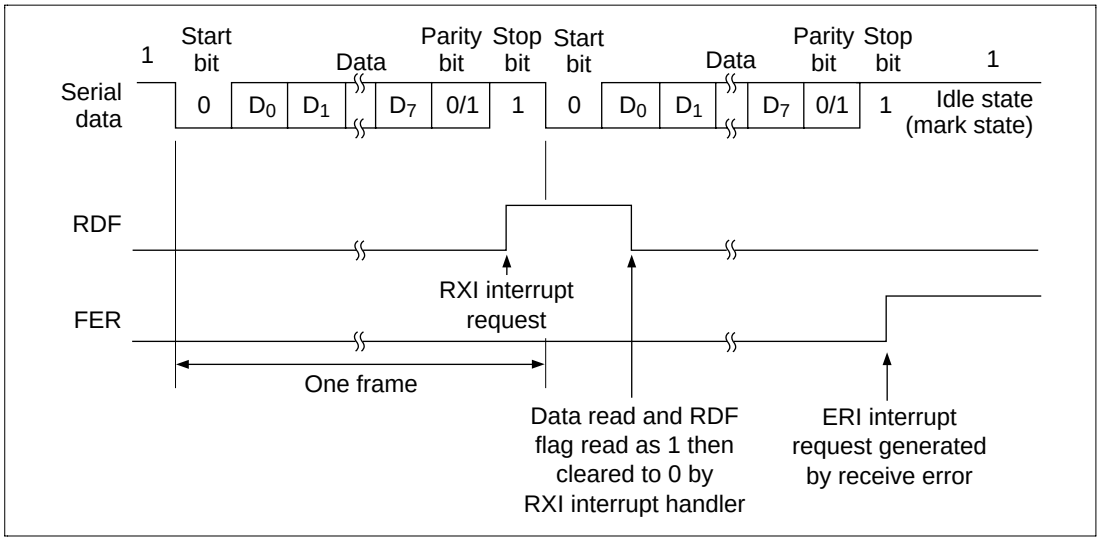



**Figure 15.7 Example of SCIF Receive Operation (Example with 8-Bit Data, Parity, One Stop Bit)**

5. When modem control is enabled, the $\overline{\text{RTS}}$ signal is output when SCFRDR is empty. When $\overline{\text{RTS}}$ is 0, reception is possible. When $\overline{\text{RTS}}$ is 1, this indicates that SCFRDR is full and reception is not possible.

   Figure 15.8 shows an example of the operation when modem control is used.

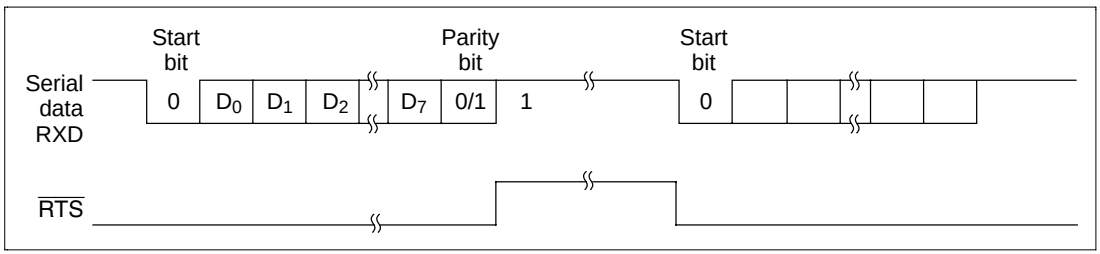



**Figure 15.8 Example of Operation Using Modem Control ($\overline{\text{RTS}}$)**

## 15.4 SCIF Interrupts

The SCIF has four interrupt sources: transmit-FIFO-data-empty (TXI), receive-error (ERI), receive-data-full (RXI), and break (BRI).

Table 15.10 shows the interrupt sources and their order of priority. The interrupt sources are enabled or disabled by means of the TIE and RIE bits in SCSCR. A separate interrupt request is sent to the interrupt controller for each of these interrupt sources.

When the TDFE flag in the serial status register (SCSSR) is set to 1, a TXI interrupt request is generated.

When the RDF flag in SCSSR is set to 1, an RXI interrupt request is generated.

When the ER flag in SCSSR is set to 1, an ERI interrupt request is generated.

When the BRK flag in SCSSR is set to 1, a BRI interrupt request is generated.

The TXI interrupt indicates that transmit data can be written, and the RXI interrupt indicates that there is receive data in SCFRDR.

**Table 15.10 SCIF Interrupt Sources**

| Interrupt Source | Description | Priority |
|---|---|---|
| ERI | Interrupt initiated by receive error flag (ER) | High |
| RXI | Interrupt initiated by receive FIFO data register full flag (RDF) or data ready flag (DR) | ↑ |
| BRI | Interrupt initiated by break flag (BRK) | |
| TXI | Interrupt initiated by transmit FIFO data register empty flag (TDFE) | ↓ Low |

See section 4, Exception Handling, for priorities and the relationship with non-SCIF interrupts.

## 15.5 Usage Notes

Note the following when using the SCIF.

### 15.5.1 SCFTDR Writing and the TDFE Flag

The TDFE flag in the serial status register (SCSSR) is set when the number of transmit data bytes written in the transmit FIFO data register (SCFTDR) has fallen to or below the transmit trigger number set by bits TTRG1 and TTRG0 in the FIFO control register (SCFCR). After TDFE is set, transmit data up to the number of empty data in SCFTDR can be written, allowing efficient continuous transmission.

However, if the number of data bytes written in SCFTDR is equal to or less than the transmit trigger number, the TDFE flag will be set to 1 again after being read as 1 and cleared to 0. TDFE clearing should therefore be carried out when SCFTDR contains more than the transmit trigger number of transmit data bytes.

The number of transmit data bytes in SCFTDR can be found from the upper 8 bits of the FIFO data count register (SCFDR).

### 15.5.2 SCFRDR Reading and the RDF Flag

The RDF flag in the serial status register (SCSSR) is set when the number of receive data bytes in the receive FIFO data register (SCFRDR) has become equal to or greater than the receive trigger number set by bits RTRG1 and RTRG0 in the FIFO control register (SCFCR). After RDF is set, receive data equivalent to the trigger number can be read from SCFRDR, allowing efficient continuous reception.

However, if the number of data bytes in SCFRDR is equal to or greater than the trigger number, the RDF flag will be set to 1 again if it is cleared to 0. Therefore, clear RDF to 0 after it is read as 1 when all the receive data has been read.

The number of receive data bytes in SCFRDR can be found from the lower 8 bits of the FIFO data count register (SCFDR).

### 15.5.3 Break Detection and Processing

Break signals can be detected by reading the RxD pin directly when a framing error (FER) is detected. In the break state the input from the RxD pin consists of all 0s, so the FER flag is set and the parity error flag (PER) may also be set. Note that, although transfer of receive data to SCFRDR is halted in the break state, the SCIF receiver continues to operate, so if the BRK flag is cleared to 0 it will be set to 1 again.

### 15.5.4 Sending a Break Signal

The input/output condition and level of the TxD pin are determined by the port B data register (PBDR), port B function control register (PBCR), and port B I/O control register (PBIOR). This feature can be used to send a break signal. To send a break signal during serial transmission, clear PBDR corresponding to the TxD pin to 0, set PBIOR to 1, set PBCR to the port function, and then clear the TE bit to 0 (halting transmission). When the TE bit is cleared to 0, the transmitter is initialized regardless of the current transmission status, and 0 is output from the TxD pin.

### 15.5.5 TEND Flag and TE Bit Processing

The TEND flag is set to 1 during transmission of the stop bit of the last data. Consequently, if the TE bit is cleared to 0 immediately after setting of the TEND flag has been confirmed, the stop bit will be in the process of transmission and will not be transmitted normally. Therefore, the TE bit should not be cleared to 0 for at least 0.5 serial clock cycles (or 1.5 cycles if two stop bits are used) after setting of the TEND flag is confirmed.

### 15.5.6 Receive Data Sampling Timing and Receive Margin

The SCIF operates on a base clock with a frequency of 16 times the transfer rate. In reception, the SCIF synchronizes internally with the fall of the start bit, which it samples on the base clock. Receive data is latched at the rising edge of the eighth base clock pulse. The timing is shown in figure 15.9.

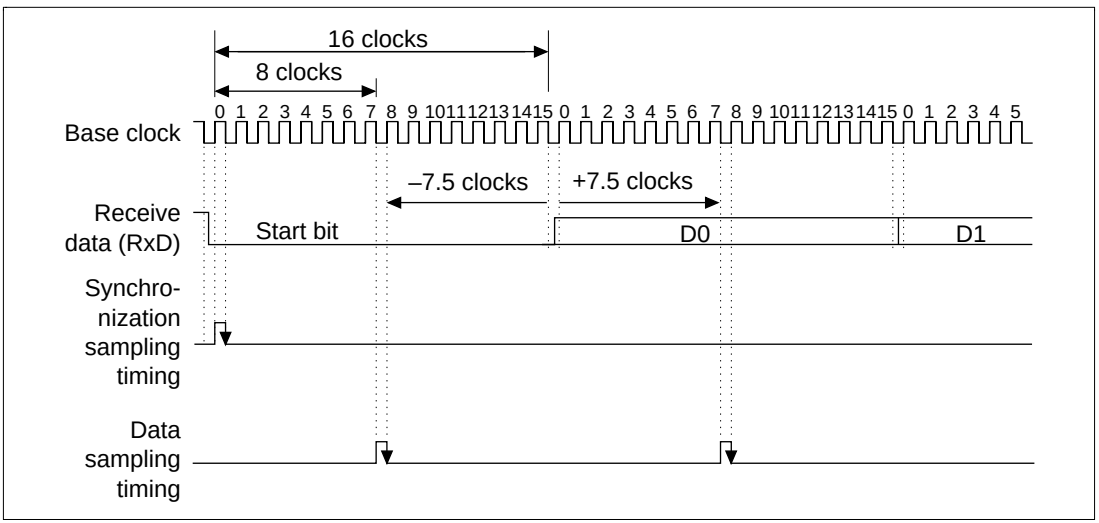



**Figure 15.9 Receive Data Sampling Timing in Asynchronous Mode**

The receive margin in asynchronous mode can therefore be expressed as shown in equation (1).

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

M: Receive margin (%)
N: Ratio of clock frequency to bit rate (N = 16)
D: Clock duty cycle (D = 0 to 1.0)
L: Frame length (L = 9 to 12)
F: Absolute deviation of clock frequency

From equation (1), if F = 0 and D = 0.5, the receive margin is 46.875%, as given by equation (2).

When D = 0.5 and F = 0:

$$M = (0.5 - 1/(2 \times 16)) \times 100\% = 46.875\% \quad \cdots\cdots (2)$$

This is a theoretical value. A reasonable margin to allow in system designs is 20% to 30%.

# Section 16 IrDA

## 16.1 Overview

An on-chip Infrared Data Association (IrDA) interface based on the IrDA 1.0 system is provided, enabling infrared communication. It can also be used as the SCIF by means of register settings.

### 16.1.1 Features

The IrDA has the following features:

- Conforms to the IrDA 1.0 system
- Asynchronous serial communication
  - Data length: 8 bits
  - Stop bit length: 1 bit
  - Parity bit: None
- Built-in 16-stage FIFO buffers for both transmission and reception
- On-chip baud rate generator with selectable bit rates
- Protection function that prevents receiver from being affected during transmission
- Clock supply halted to reduce power consumption when the IrDA is not in use

### 16.1.2 Block Diagram

Figure 16.1 shows a block diagram of the IrDA.

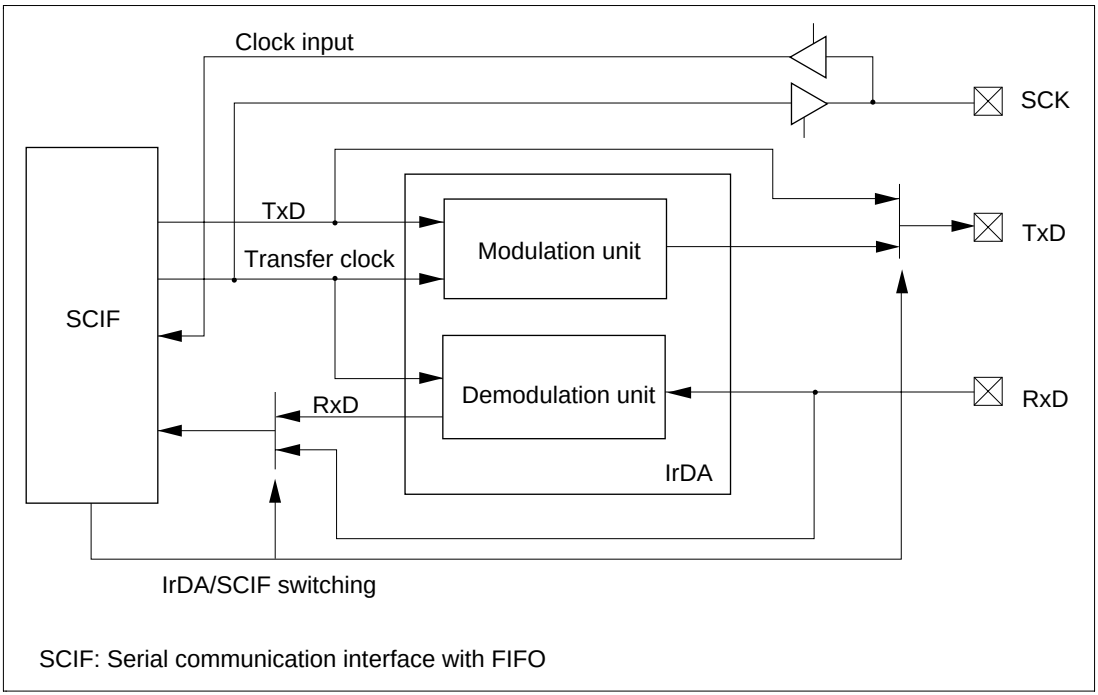



**Figure 16.1 IrDA Block Diagram**

### 16.1.3 Pin Configuration

Table 16.1 shows the pin configuration of the IrDA.

**Table 16.1 Pin Configuration**

| Channel | Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|---|
| 1 | Serial clock pin | SCK1 | Input/output | Clock input/output |
| | Receive data pin | RxD1 | Input | Receive data input |
| | Transmit data pin | TxD1 | Output | Transmit data output |
| 2 | Serial clock pin | SCK2 | Input/output | Clock input/output |
| | Receive data pin | RxD2 | Input | Receive data input |
| | Transmit data pin | TxD2 | Output | Transmit data output |

### 16.1.4 Register Configuration

The IrDA has the internal registers shown in table 16.2. By using these registers, the IrDA or SCIF mode can be selected, the data format and bit rate can be specified, and the transmit and receive units can be controlled.

**Table 16.2 Register Configuration**

| Channel | Register Name | Abbre-viation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|---|
| 1 | Serial mode register 1 | SCSMR1 | R/W | H'00 | H'FFFFFCC0 | 8 bits |
| | Bit rate register 1 | SCBRR1 | R/W | H'FF | H'FFFFFCC2 | 8 bits |
| | Serial control register 1 | SCSCR1 | R/W | H'00 | H'FFFFFCC4 | 8 bits |
| | Transmit FIFO data register 1 | SCFTDR1 | W | — | H'FFFFFCC6 | 8 bits |
| | Serial status register 1 | SCSSR1 | R/(W)* | H'0060 | H'FFFFFCC8 | 16 bits |
| | Receive FIFO data register 1 | SCFRDR1 | R | Undefined | H'FFFFFCCA | 8 bits |
| | FIFO control register 1 | SCFCR1 | R/W | H'00 | H'FFFFFCCC | 8 bits |
| | FIFO data count set register 1 | SCFDR1 | R | H'0000 | H'FFFFFCCE | 16 bits |
| 2 | Serial mode register 2 | SCSMR2 | R/W | H'00 | H'FFFFFCD0 | 8 bits |
| | Bit rate register 2 | SCBRR2 | R/W | H'FF | H'FFFFFCD2 | 8 bits |
| | Serial control register 2 | SCSCR2 | R/W | H'00 | H'FFFFFCD4 | 8 bits |
| | Transmit FIFO data register 2 | SCFTDR2 | W | — | H'FFFFFCD6 | 8 bits |
| | Serial status register 2 | SCSSR2 | R/(W)* | H'0060 | H'FFFFFCD8 | 16 bits |
| | Receive FIFO data register 2 | SCFRDR2 | R | Undefined | H'FFFFFCDA | 8 bits |
| | FIFO control register 2 | SCFCR2 | R/W | H'00 | H'FFFFFCDC | 8 bits |
| | FIFO data count set register 2 | SCFDR2 | R | H'0000 | H'FFFFFCDE | 16 bits |

Note: * Only 0 can be written, to clear the flags.

## 16.2 Register Description

Specifications of the registers in the IrDA are the same as those in the SCIF except for the serial mode register described below. Therefore, refer to section 15, Serial Communication Interface with FIFO, for these registers.

### 16.2.1 Serial Mode Register (SCSMR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IRMOD | ICK3 | ICK2 | ICK1 | ICK0 | PSEL | CKS1 | CKS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

SCSMR is an 8-bit register which can select IrDA or SCIF mode, specify the SCIF serial communication format, select the IrDA output pulse width, and select the baud rate generator clock source.

This module operates as IrDA by setting the IRMOD bit to 1. At this time, bits 3 to 6 are fixed at 0. This register functions in the same way as the SCSMR register in the SCIF by setting the IRMOD bit to 0; therefore, this module can also operates as the SCIF.

SCSMR is initialized to H'00 by a power-on reset, manual reset, stoppage by the module standby function, or standby mode.

Bit 7—IrDA Mode (IRMOD): Selects whether this module operates as an IrDA serial communication interface or as an SCIF.

| Bit 7: IRMOD | Description | |
|---|---|---|
| 0 | Operates as SCIF | (Initial value) |
| 1 | Operates as IrDA | |

Bits 6 to 3—IrDA Clock Select Bits (ICK3–ICK0)

Bit 2—Output Pulse Width Select (PSEL)

The output pulse width select bit (PSEL) selects an output pulse width of IrDA that is 3/16 of the bit length for 115 kbps or 3/16 of the bit length for the selected baud rate.

The IrDA clock select bits should be set properly to fix the output pulse width at 3/16 of the bit length for 115 kbps by setting the PSEL bit to 1.

| Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Description |
|---|---|---|---|---|---|
| ICK3 | ICK2 | ICK1 | ICK0 | PSEL | Pulse width: 3/16 of 115 kbps bit length |
| ICK3 | ICK2 | ICK1 | ICK0 | 1 | |
| Don't care | Don't care | Don't care | Don't care | 0 | Pulse width: 3/16 of bit length |

It is necessary to generate a fixed clock pulse, IRCLK, by dividing the Pϕ clock by 1/2N+2, with the value of N determined by the setting of ICK3–ICK0.

Example:

Pϕ clock: 14.7456 MHz
IRCLK: 921.6 kHz (fixed)
N: Setting of ICK3–ICK0 ($0 \leq N \leq 15$)

$$N \geq \frac{P\phi}{2 \times IRCLK} - 1 \geq 7$$

Thus, N is 7.

Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal baud rate generator clock source. Pϕ, Pϕ/4, Pϕ/16, or Pϕ/64 can be selected by setting the CKS1 and CKS0 bits.

Refer to section 15.2.8, Bit Rate Register, for the relationship between the clock source, the bit rate register set value, and the baud rate.

| Bit 1: CKS1 | Bit 0: CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | Pϕ clock | (Initial value) |
| | 1 | Pϕ/4 clock | |
| 1 | 0 | Pϕ/16 clock | |
| | 1 | Pϕ/64 clock | |

Note: Pϕ: Peripheral module clock

## 16.3 Operation

The IrDA module can perform infrared communication conforming to IrDA 1.0 by connecting an infrared transmit/receive unit. The serial communication interface unit includes a 16-stage FIFO buffer in the transmit section and the receive section, enabling CPU overhead to be reduced and continuous high-speed communication to be performed. The IrDA module differs from the SCIF described in section 15 in that it does not include modem control signals $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$.

Refer to section 15.3, Operation, for SCIF mode operation.

### 16.3.1 Overview

The IrDA module modifies TxD/RxD transmit/receive data waveforms to satisfy the IrDA 1.0 specification for infrared communication.

In the IrDA 1.0 specification, communication is first performed at the rate of 9,600 bps, and the communication rate is changed. However, the communication rate cannot be automatically changed in this module, and therefore the communication rate must be checked and the appropriate rate set in this module with software.

Note: In IrDA mode, reception cannot be performed when the TE bit in the serial control register (SCSCR) is set to 1 (enabling transmission). When performing reception, clear the TE bit in SCSCR to 0.

### 16.3.2 Transmission

The waveform of the serial output signal (UART frame) from the SCIF is modified and the signal is converted into the IR frame serial output signal by the IrDA module, as shown in figure 16.2.

When serial data is 0, a 3/16-bit width high pulse of the IR frame is generated and output. When serial data is 1, a pulse is not output.

An infrared LED is driven with high this signal, demodulated into 3/16 width.

### 16.3.3 Reception

The 3/16-bit width pulse of the IR frame that was received is demodulated and converted into a UART frame, as shown in figure 16.2.

Demodulation to 0 is performed for low pulse output; demodulation to 1 is not performed for pulse output.

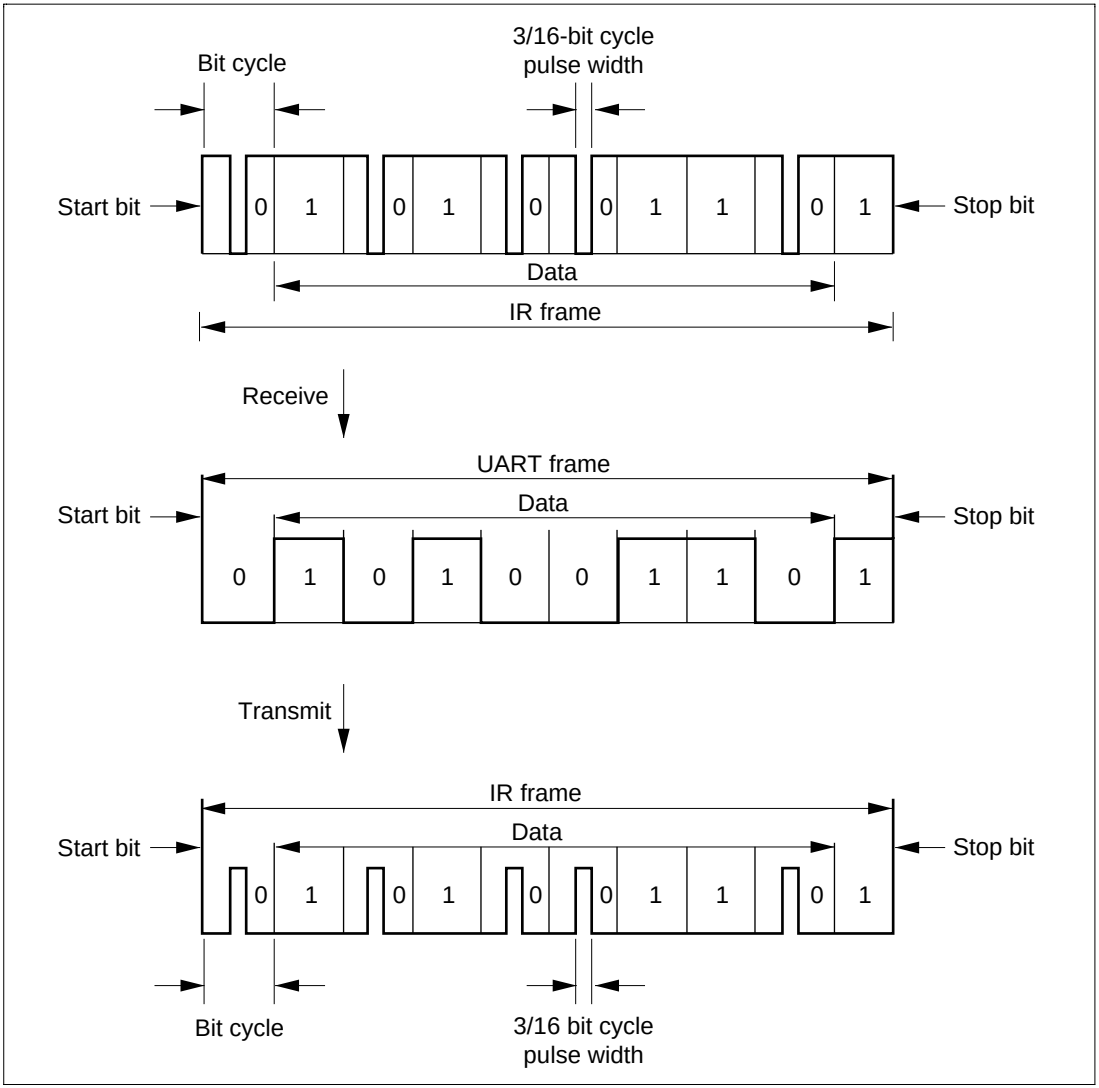



**Figure 16.2 Transmit/Receive Operation**

# Section 17 Serial I/O (SIO)

## 17.1 Overview

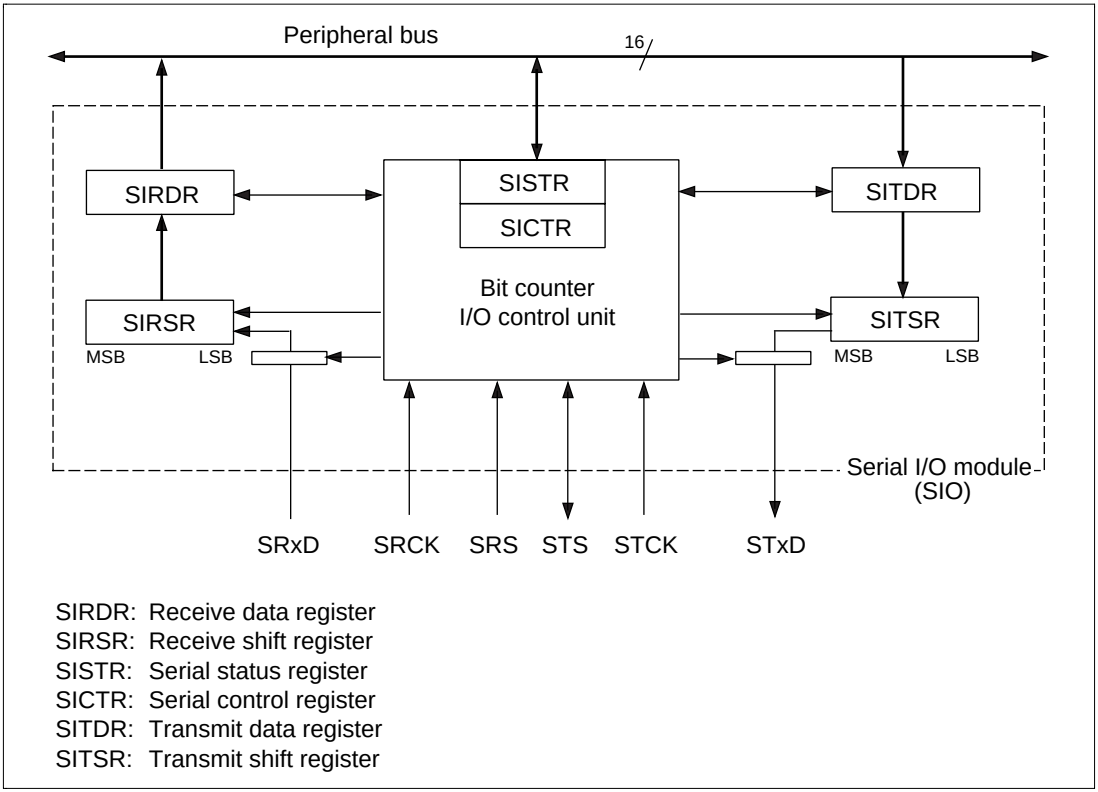
A three-channel simple synchronous serial I/O is provided on-chip. The serial I/O functions mainly as an interface between the chip and a codec or modem analog front-end.

### 17.1.1 Features

The serial I/O has the following features:

- Full-duplex operation
  Independent transmit/receive registers and independent transmit/receive clocks
- Double-buffered transmit/receive ports
  Continuous data transmission/reception possible
- Interval transfer mode and continuous transfer mode
- Memory-mapped receive register, transmit register, control register, and status register
  With the exception of SIRSR and SITSR, these registers are memory-mapped and can be accessed by a MOV instruction.
- Choice of 8- or 16-bit data length
- Data transfer communication by means of polling or interrupts
  Data transfer can be monitored by polling the receive data register full flag (RDRF) and transmit data register empty flag (TDRE) in the serial status register. Interrupt requests can be generated during data transfer by setting the receive interrupt request flag and transmit interrupt request flag.
- MSB-first transfer between SIO and data I/O

Figure 17.1 shows a block diagram of the serial I/O.




SIRDR: Receive data register
SIRSR: Receive shift register
SISTR: Serial status register
SICTR: Serial control register
SITDR: Transmit data register
SITSR: Transmit shift register

**Figure 17.1 SIO Block Diagram**

Table 17.1 shows the functions of the external pins. As the channels are independent, the channel numbers are omitted from the signal names in the rest of this section.

**Table 17.1 Serial I/O (SIO) External Pins**

| Channel | Name | Pin | I/O | Function |
|---|---|---|---|---|
| 0 | Serial receive data input pin | SRxD0 | Input | Serial data input port 0 |
| | Serial receive clock input pin | SRCK0 | Input | Serial receive clock port 0 |
| | Serial reception synchronization input pin | SRS0 | Input | Serial reception synchronization input port 0 |
| | Serial transmit data output pin | STxD0 | Output | Serial data output port 0 |
| | Serial transmit clock input pin | STCK0 | Input | Serial transmit clock port 0 |
| | Serial transmission synchronization input/output pin | STS0 | I/O | Serial transmission synchronization input/output port 0 |
| 1 | Serial receive data input pin | SRxD1 | Input | Serial data input port 1 |
| | Serial receive clock input pin | SRCK1 | Input | Serial receive clock port 1 |
| | Serial reception synchronization input pin | SRS1 | Input | Serial reception synchronization input port 1 |
| | Serial transmit data output pin | STxD1 | Output | Serial data output port 1 |
| | Serial transmit clock input pin | STCK1 | Input | Serial transmit clock port 1 |
| | Serial transmission synchronization input/output pin | STS1 | I/O | Serial transmission synchronization input/output port 1 |
| 2 | Serial receive data input pin | SRxD2 | Input | Serial data input port 2 |
| | Serial receive clock input pin | SRCK2 | Input | Serial receive clock port 2 |
| | Serial reception synchronization input pin | SRS2 | Input | Serial reception synchronization input port 2 |
| | Serial transmit data output pin | STxD2 | Output | Serial data output port 2 |
| | Serial transmit clock input pin | STCK2 | Input | Serial transmit clock port 2 |
| | Serial transmission synchronization input/output pin | STS2 | I/O | Serial transmission synchronization input/output port 2 |

Note: In a reset, all pins are initialized to the high-impedance state.

## 17.2 Register Configuration

Table 17.2 shows the SIO's registers. As the channels are independent, the channel numbers are omitted from the signal names in the rest of this section.

**Table 17.2 Register Configuration**

| Channel | Register | Abbreviation | R/W | Initial Value | Address | Access Size (Bits) |
|---|---|---|---|---|---|---|
| 0 | Receive shift register 0 | SIRSR0 | — | — | — | — |
| | Receive data register 0 | SIRDR0 | R | H'0000 | H'FFFFFC00 | 8, 16, 32 |
| | Transmit shift register 0 | SITSR0 | — | — | — | — |
| | Transmit data register 0 | SITDR0 | R/W | H'0000 | H'FFFFFC02 | 8, 16, 32 |
| | Serial control register 0 | SICTR0 | R/W | H'0000 | H'FFFFFC04 | 8, 16, 32 |
| | Serial status register 0 | SISTR0 | R/(W)* | H'0002 | H'FFFFFC06 | 8, 16, 32 |
| 1 | Receive shift register 1 | SIRSR1 | — | — | — | — |
| | Receive data register 1 | SIRDR1 | R | H'0000 | H'FFFFFC10 | 8, 16, 32 |
| | Transmit shift register 1 | SITSR1 | — | — | — | — |
| | Transmit data register 1 | SITDR1 | R/W | H'0000 | H'FFFFFC12 | 8, 16, 32 |
| | Serial control register 1 | SICTR1 | R/W | H'0000 | H'FFFFFC14 | 8, 16, 32 |
| | Serial status register 1 | SISTR1 | R/(W)* | H'0002 | H'FFFFFC16 | 8, 16, 32 |
| 2 | Receive shift register 2 | SIRSR2 | — | — | — | — |
| | Receive data register 2 | SIRDR2 | R | H'0000 | H'FFFFFC20 | 8, 16, 32 |
| | Transmit shift register 2 | SITSR2 | — | — | — | — |
| | Transmit data register 2 | SITDR2 | R/W | H'0000 | H'FFFFFC22 | 8, 16, 32 |
| | Serial control register 2 | SICTR2 | R/W | H'0000 | H'FFFFFC24 | 8, 16, 32 |
| | Serial status register 2 | SISTR2 | R/(W)* | H'0002 | H'FFFFFC26 | 8, 16, 32 |

Note: * Only 0 should be written, to clear flags (after reading 1 from the flag).

### 17.2.1 Receive Shift Register (SIRSR)

| Bit: | 15 | 14 | 13 | ... | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ... | | | | |
| Initial value: | — | — | — | ... | — | — | — | — |
| R/W: | — | — | — | ... | — | — | — | — |

SIRSR is a 16-bit register used to receive serial data. The data is fetched in MSB first from the SRxD pin in synchronization with the fall of the serial receive clock (SRCK), and is shifted into SIRSR. The data length is set by the transmit/receive data length select bit (DL) in the corresponding serial control register (SICTR). When data transfer to SIRSR is completed, the data contents are automatically transferred to the receive data register (SIRDR), and the receive data register full flag (RDRF) is set in the serial status register (SISTR).

If the next data word input operation ends before the RDRF flag is cleared, an overrun error occurs, the receive overrun error flag (RERR) is set in SISTR, and an overrun error signal is sent to the interrupt controller (INTC). The data in SIRSR overwrites the data in SIRDR.

### 17.2.2 Receive Data Register (SIRDR)

| Bit: | 15 | 14 | 13 | ... | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ... | | | | |
| Initial value: | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | ... | R | R | R | R |

SIRDR is a 16-bit register that stores serial receive data. When data is transferred from SIRSR to SIRDR, the receive data register full flag (RDRF) is set in the serial status register (SISTR). If the receive interrupt enable flag (RIE) is set in SICTR, a receive-data-full interrupt (RDFI) request is sent to the interrupt controller (INTC). If the transmit interrupt enable flag (TIE) in SICTR is cleared, this interrupt request signal is not generated. SIRDR is initialized to H'0000 by a reset.

### 17.2.3 Transmit Shift Register (SITSR)

| Bit: | 15 | 14 | 13 | ... | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ... | | | | |
| Initial value: | — | — | — | ... | — | — | — | — |
| R/W: | — | — | — | ... | — | — | — | — |

SITSR is a 16-bit register used to transmit serial data. The contents of this register are shifted in MSB-first order in synchronization with the rising edge of the serial transmit clock (STCK), and output from the STxD pin. The transfer data length is set by the transmit/receive data length select bit (DL) in the serial control register (SICTR). When the DL bit is cleared to 0 (8-bit data length), the lower 8 bits of the transmit data register (SITDR) are output. When the serial transmission synchronization signal (STS) goes high, or the last data transmission ends without the synchronization enable (SE) bit being set in SICTR, the contents of SITDR are transferred to SITSR, and if the transmit data empty flag (TDRE) in SISTR is 0, TDRE is then set. If output of the next data begins before TDRE is cleared, an overrun error occurs, the transmit overrun error flag (TERR) is set in SISTR, and a transmit overrun error interrupt request is sent to the INTC.

### 17.2.4 Transmit Data Register (SITDR)

| Bit: | 15 | 14 | 13 | ... | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ... | | | | |
| Initial value: | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | ... | R/W | R/W | R/W | R/W |

SITDR is a 16-bit register that stores serial transmit data. Data should be written to SITDR when the transmit data register empty flag (TDRE) is set to 1 in SISTR. If data is written to SITDR when TDRE is 0, the previous data will be overwritten. When STS goes high or data output from transmit shift register SITSR ends with the SE bit cleared to 0 in SICTR, the data in SITDR is automatically transferred to SITSR, and if TDRE is 0, TDRE is then set. If the transmit interrupt enable flag (TIE) is set, a transmit-data-empty interrupt (TDEI) request is sent to the INTC. When TIE is cleared, this interrupt request is not generated. The TDRE flag is set only by hardware. SITDR is initialized to H'0000 by a reset.

### 17.2.5 Serial Control Register (SICTR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | TM | SE | DL | TIE | RIE | TE | RE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

SICTR is a 16-bit register used to set parameters for serial port control. SICTR is initialized to H'0000 by a reset.

When modifying bit 4, 5, or 6 (TM, SE, or DL), TE and RE should be cleared to 0 beforehand.

Bits 15 to 7—Reserved: These bits always read 0. The write value should always be 0.

Bit 6—Transfer Mode Control (TM): Specifies whether the transmission synchronization signal is to be input from an external source or generated internally by the chip. When this flag is cleared, the transmission synchronization signal is STS pin input. When this flag is set, the transmission synchronization signal is generated by the chip, and is output to an external device from the STS pin. This bit does not affect reception.

| Bit 6: TM | Description |
|---|---|
| 0 | External signal input from STS pin is used as transmission start indication (Initial value) |
| 1 | Internal signal output from STS pin is used as transmission start indication |

Bit 5—Synchronization Signal Enable (SE): Specifies whether the synchronization signals are to be used for all serial data transfers, or only for the first transfer.

When this bit is cleared to 0, the synchronization signals (SRS and STS) are necessary only for the first data transfer, and are not required for subsequent transfers. When this bit is set to 1, the synchronization signals are necessary for all data transfers.

| Bit 5: SE | Description |
|---|---|
| 0 | Continuous mode: SRS and STS are used only for the first data transfer (Initial value) |
| 1 | Interval mode: SRS and STS are used for all data transfers |

Bit 4—Transmit/Receive Data Length Select (DL): Specifies the serial I/O module's transfer data length. The initial value of this bit is 0, indicating an 8-bit data length. When an 8-bit data length is specified, the lower 8 bits of each I/O register are used.

| Bit 4: DL | Description | |
|---|---|---|
| 0 | 8-bit transfer data length | (Initial value) |
| 1 | 16-bit transfer data length | |

Bit 3—Transmit Interrupt Enable (TIE): Enables the transmit-data-empty interrupt. The initial value of this bit is 0.

| Bit 3: TIE | Description | |
|---|---|---|
| 0 | Transmit interrupt disabled | (Initial value) |
| 1 | Transmit interrupt enabled | |

Bit 2—Receive Interrupt Enable (RIE): Enables the receive-data-full interrupt. The initial value of this bit is 0.

| Bit 2: RIE | Description | |
|---|---|---|
| 0 | Receive interrupt disabled | (Initial value) |
| 1 | Receive interrupt enabled | |

Bit 1—Transmit Enable (TE): Enables data transmission. When this flag is cleared, the STxD, STCK, and STS pins go to the high-impedance state.

| Bit 1: TE | Description | |
|---|---|---|
| 0 | Transmission disabled: STxD, STCK, and STS pins go to high-impedance state | (Initial value) |
| 1 | Transmission enabled | |

Bit 0—Receive Enable (RE): Enables data reception. When this flag is cleared, the SRxD, SRCK, and SRS pins go to the high-impedance state.

| Bit 0: RE | Description | |
|---|---|---|
| 0 | Reception disabled: SRxD, SRCK, and SRS pins go to high-impedance state | (Initial value) |
| 1 | Reception enabled | |

### 17.2.6 Serial Status Register (SISTR)

| Bit: | 15 | 14 | ... | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | ... | — | TERR | RERR | TDRE | RDRF |
| Initial value: | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | ... | R | R/(W)* | R/(W)* | R/(W)* | R/(W)* |

Note: * Only 0 should be written, to clear the flag.

SISTR is a 16-bit register that indicates the status of the serial I/O module. SISTR is initialized to H'0002 by a reset.

Bits 15 to 4—Reserved: These bits always read 0. The write value should always be 0.

Bit 3—Transmit Overrun Error (TERR): Flag that indicates the occurrence of a transmit overrun.

| Bit 3: TERR | Description | |
|---|---|---|
| 0 | Transmission is in progress, or has ended normally | (Initial value) |
| | TERR is cleared to 0 in the following cases: | |
| | • When 0 is written to the TERR bit after reading TERR = 1 | |
| | • When the SH7612 enters the reset state | |
| 1 | A transmit overrun error has occurred | |
| | TERR is set to 1 if data transmission is started while TDRE = 1 | |

Bit 2—Receive Overrun Error (RERR): Flag that indicates the occurrence of a receive overrun.

| Bit 2: RERR | Description | |
|---|---|---|
| 0 | Reception is in progress, or has ended normally | (Initial value) |
| | RERR is cleared to 0 in the following cases: | |
| | • When 0 is written to the RERR bit after reading RERR = 1 | |
| | • When the SH7612 enters the reset state | |
| 1 | A receive overrun error has occurred | |
| | RERR is set to 1 if data reception ends while RDRE = 1 | |

Bit 1—Transmit Data Register Empty (TDRE): Flag that indicates that the SITDR register is empty and the next data can be written.

| Bit 1: TDRE | Description |
|---|---|
| 0 | SITDR transmit data is valid<br>TDRE is cleared to 0 in the following cases:<br>• When 0 is written to the TDRE bit after reading TDRE = 1 |
| 1 | SITDR transmit data is invalid (Initial value)<br>TDRE is set to 1 in the following cases:<br>• When data is transferred from SITDR to SITSR<br>• When the TE bit is cleared to 0 in the serial control register (SICTR)<br>• When the SH7612 enters the reset state |

Bit 0—Receive Data Register Full (RDRF): Flag that indicates that SIRDR receive data is waiting.

| Bit 0: RDRF | Description |
|---|---|
| 0 | SIRDR receive data is invalid (Initial value)<br>RDRF is cleared to 0 in the following cases:<br>• When 1 is read from RDRF and 0 is written<br>• When the RE bit is cleared to 0 in the serial control register (SICTR)<br>• When the SH7612 enters the reset state |
| 1 | SIRDR receive data is valid<br>RDRF is set to 1 when serial data reception ends normally and the data is transferred from SIRSR to SIRDR |

## 17.3 Operation

### 17.3.1 Input

Figure 17.2 shows interval transfer mode (SE set to 1 in SICTR), and figure 17.3 shows continuous transfer mode (SE cleared to 0 in SICTR).

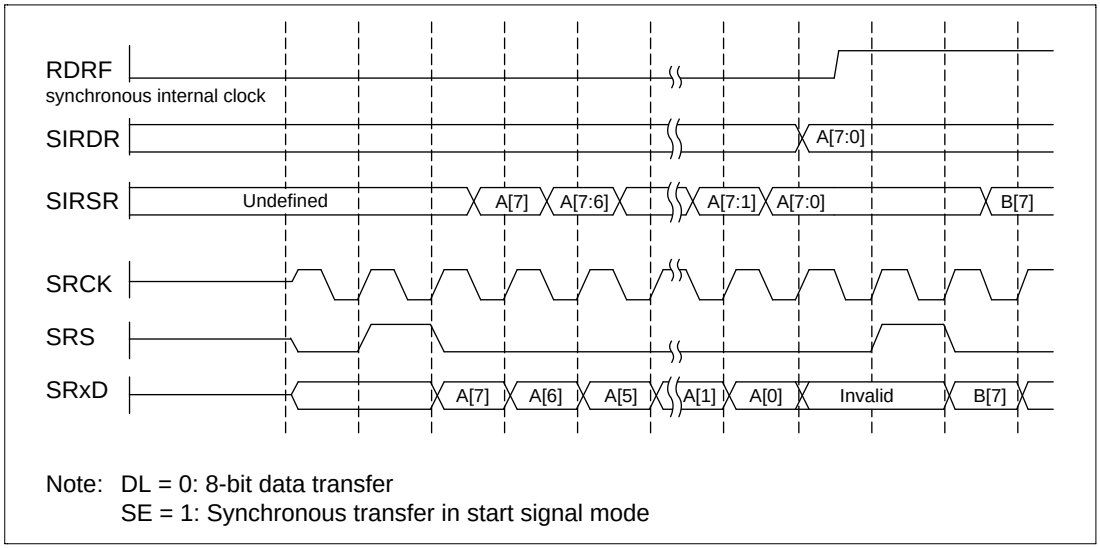



**Figure 17.2 Reception: Interval Transfer Mode**

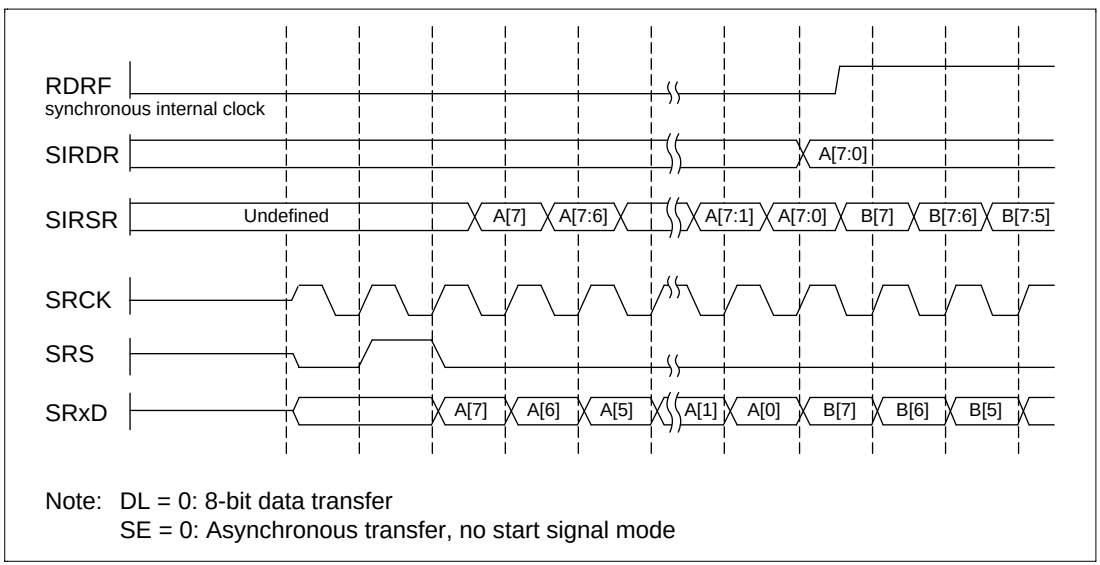



**Figure 17.3 Reception: Continuous Transfer Mode**

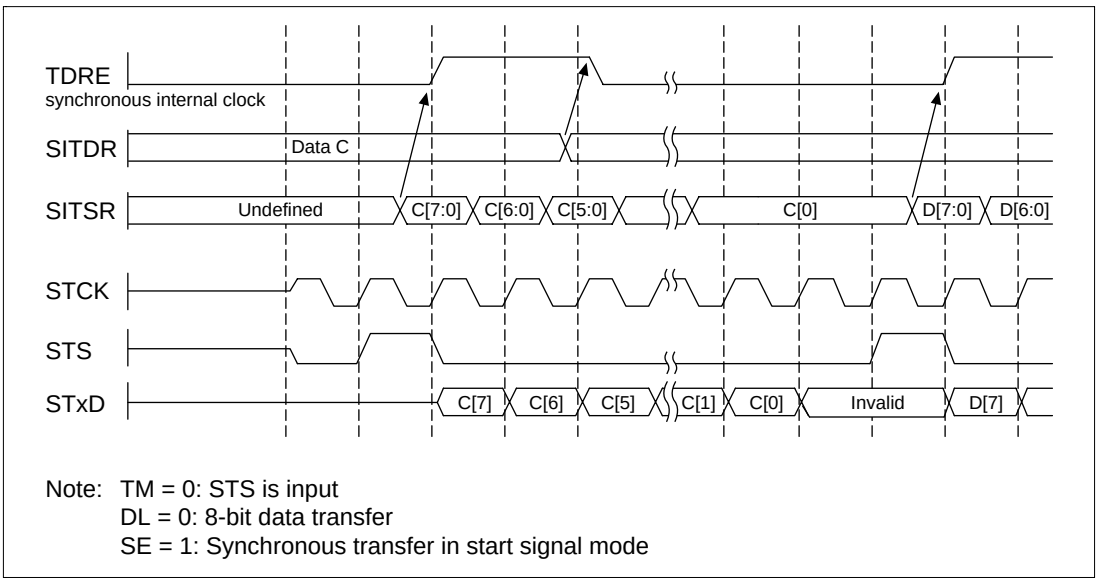
### 17.3.2 Output

Figure 17.4 shows interval transfer mode (SE set to 1 in SICTR) when TM is cleared to 0 in SICTR.

Figure 17.5 shows continuous transfer mode (SE cleared to 0 in SICTR) when TM is cleared to 0 in SICTR.

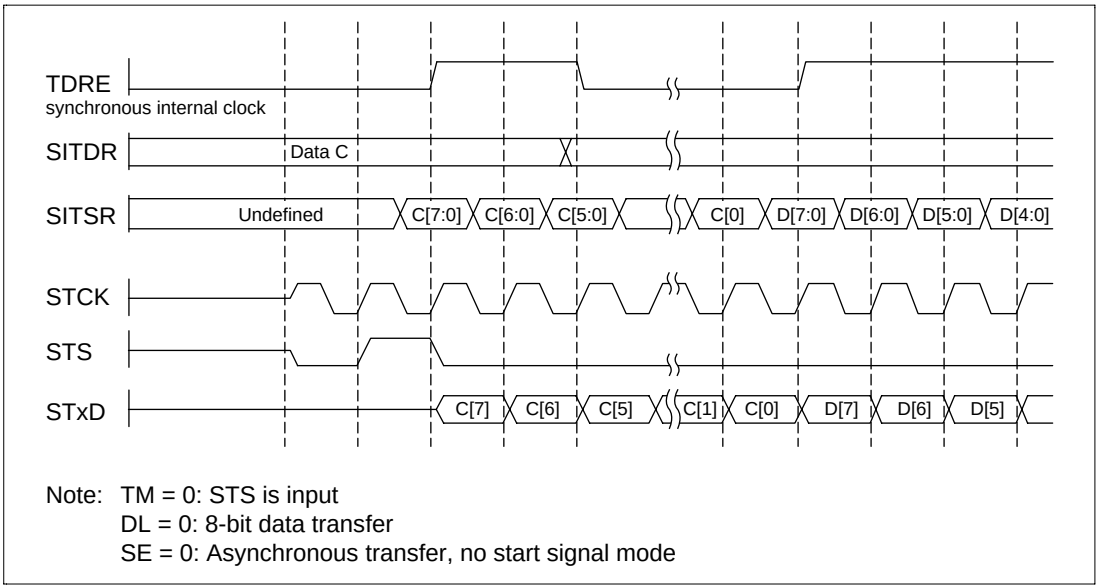
Figure 17.6 shows interval transfer mode (SE set to 1 in SICTR) when TM is set to 1 in SICTR.

Figure 17.7 shows continuous transfer mode (SE cleared to 0 in SICTR) when TM is set to 1 in SICTR.



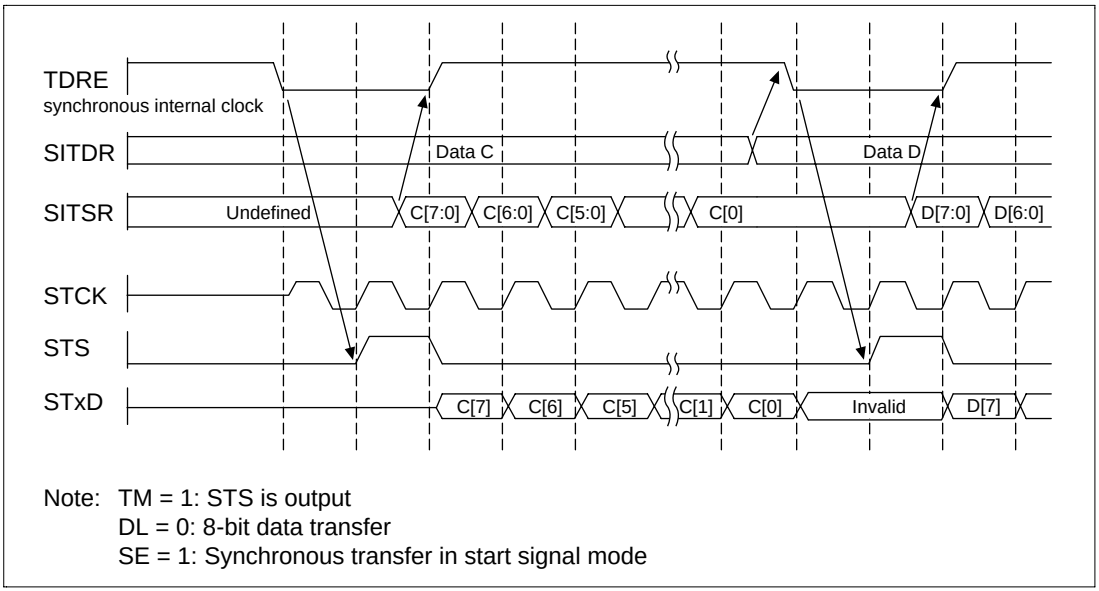

Note: TM = 0: STS is input
DL = 0: 8-bit data transfer
SE = 1: Synchronous transfer in start signal mode

**Figure 17.4 Transmission: Interval Transfer Mode (TM = 0 Mode)**



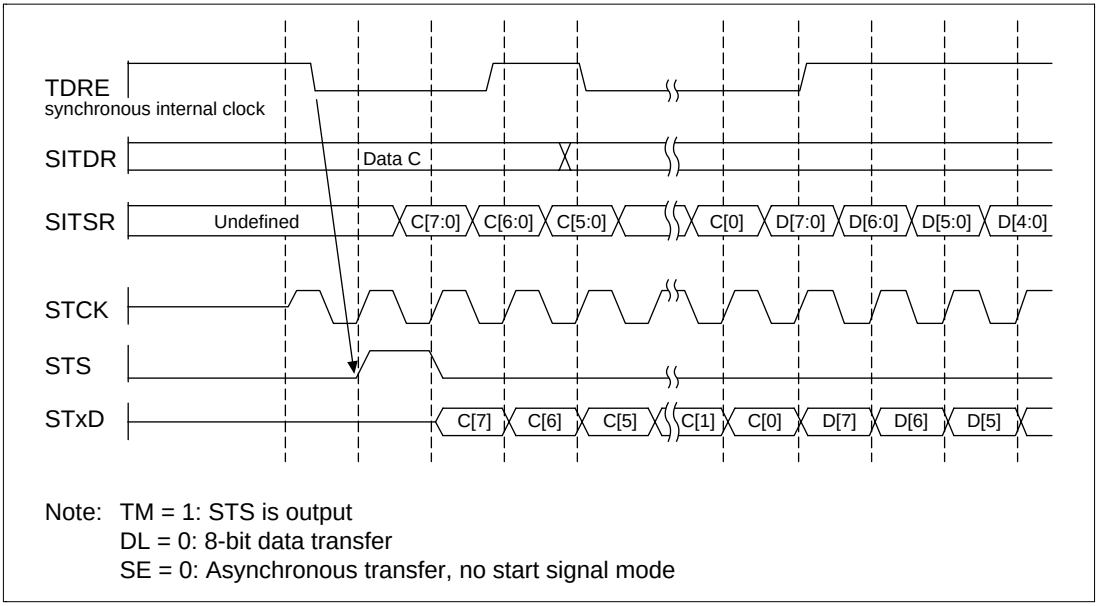

Note: TM = 0: STS is input
DL = 0: 8-bit data transfer
SE = 0: Asynchronous transfer, no start signal mode

**Figure 17.5 Transmission: Continuous Transfer Mode (TM = 0 Mode)**




Note: TM = 1: STS is output
DL = 0: 8-bit data transfer
SE = 1: Synchronous transfer in start signal mode

**Figure 17.6 Transmission: Interval Transfer Mode (TM = 1 Mode)**




Note: TM = 1: STS is output
DL = 0: 8-bit data transfer
SE = 0: Asynchronous transfer, no start signal mode

**Figure 17.7 Transmission: Continuous Transfer Mode (TM = 1 Mode)**

## 17.4 SIO Interrupt Sources and DMAC

Each SIO channel has four interrupt sources: the receive-overrun-error interrupt (RERI) request, transmit-overrun-error interrupt (TERI) request, receive-data-full interrupt (RDFI) request, and transmit-data-empty interrupt (TDEI) request. Table 17.3 shows the interrupt sources and their relative priorities. The RDFI and TDEI interrupts are enabled by the RIE and TIE bits, respectively, in SICTR. The RERI and TERI interrupts cannot be disabled.

An RDFI interrupt request is generated when the RDRF bit is set to 1 in SISTR.

A TDEI interrupt request is generated when the TDRE bit is set to 1 in SISTR.

When the RERR bit is set to 1 in SISTR, an RERI interrupt request is generated.

When the TERR bit is set to 1 in SISTR, a TERI interrupt request is generated.

Channel interrupt priority levels are set by means of the IRPE register, as described in section 5, Interrupt Controller (INTC).

**Table 17.3 SIO Interrupt Sources**

| Interrupt Source | Description | Priority |
|---|---|---|
| RERI | Receive overrun error (RERR) | High |
| TERI | Transmit overrun error (TERR) | ↑ |
| RDFI | Receive data register full (RDRF) | ↓ |
| TDEI | Transmit data register empty (TDRE) | Low |

# Section 18 16-Bit Timer Pulse Unit (TPU)

## 18.1 Overview

An on-chip 16-bit timer pulse unit (TPU) is provided that comprises three 16-bit timer channels.

### 18.1.1 Features

The TPU has the following features:

- Maximum 8-pulse input/output
- A total of eight timer general registers (TGRs) are provided (four for channel 0 and two each for channels 1, and 2).
  - — Each register can be set independently as an output compare/input capture register.
  - — TGRC and TGRD for channel 0 can be used as buffer registers
- Choice of seven or eight counter input clocks for each channel
- The following operations can be set for each channel:
  - — Waveform output by compare match: Selection of 0, 1, or toggle output
  - — Input capture function: Choice of rising edge, falling edge, or both edge detection
  - — Counter clear operation: Counter clearing possible by compare match or input capture
  - — Synchronous operation: Multiple timer counters (TCNT) can be written to simultaneously
    Simultaneous clearing by compare match and input capture possible
    Register simultaneous input/output possible by counter synchronous operation
  - — PWM mode: Any PWM output duty can be set
  - — Maximum of 7-phase PWM output possible by combination with synchronous operation
- Buffer operation settable for channel 0
  - — Input capture register double-buffering possible
  - — Automatic rewriting of output compare register possible
- Phase counting mode settable independently for each of channels 1, and 2
  - — Two-phase encoder pulse up/down-count possible
- Fast access via internal 16-bit bus
  - — Fast access is possible via a 16-bit bus interface
- 13 interrupt sources
  - — For channel 0 four compare match/input capture dual-function interrupts and one overflow interrupt can be requested independently
  - — For channels 1, and 2, two compare match/input capture dual-function interrupts, one overflow interrupt, and one underflow interrupt can be requested independently

Table 18.1 lists the functions of the TPU.

**Table 18.1 TPU Functions**

| Item | | Channel 0 | Channel 1 | Channel 2 |
|---|---|---|---|---|
| Count clock | | Pϕ/1<br>Pϕ/4<br>Pϕ/16<br>Pϕ/64<br>TCLKA<br>TCLKB<br>TCLKC<br>TCLKD | Pϕ/1<br>Pϕ/4<br>Pϕ/16<br>Pϕ/64<br>Pϕ/256<br>TCLKA<br>TCLKB | Pϕ/1<br>Pϕ/4<br>Pϕ/16<br>Pϕ/64<br>Pϕ/1024<br>TCLKA<br>TCLKB<br>TCLKC |
| General registers | | TGR0A<br>TGR0B | TGR1A<br>TGR1B | TGR2A<br>TGR2B |
| General registers/ buffer registers | | TGR0C<br>TGR0D | — | — |
| I/O pins | | TIOCA0<br>TIOCB0<br>TIOCC0<br>TIOCD0 | TIOCA1<br>TIOCB1 | TIOCA2<br>TIOCB2 |
| Counter clear function | | TGR compare match or input capture | TGR compare match or input capture | TGR compare match or input capture |
| Compare match output | 0 output | O | O | O |
| | 1 output | O | O | O |
| | Toggle output | O | O | O |
| Input capture function | | O | O | O |
| Synchronous operation | | O | O | O |
| PWM mode | | O | O | O |
| Phase counting mode | | — | O | O |
| Buffer operation | | O | — | — |

Notes: O : Possible
— : Not possible

**Table 18.1 TPU Functions (cont)**

| Item | Channel 0 | Channel 1 | Channel 2 |
|---|---|---|---|
| DMAC activation | TGR compare match or input capture | TGR compare match or input capture | TGR 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 |

### 18.1.2 Block Diagram

Figure 18.1 shows a block diagram of the TPU.

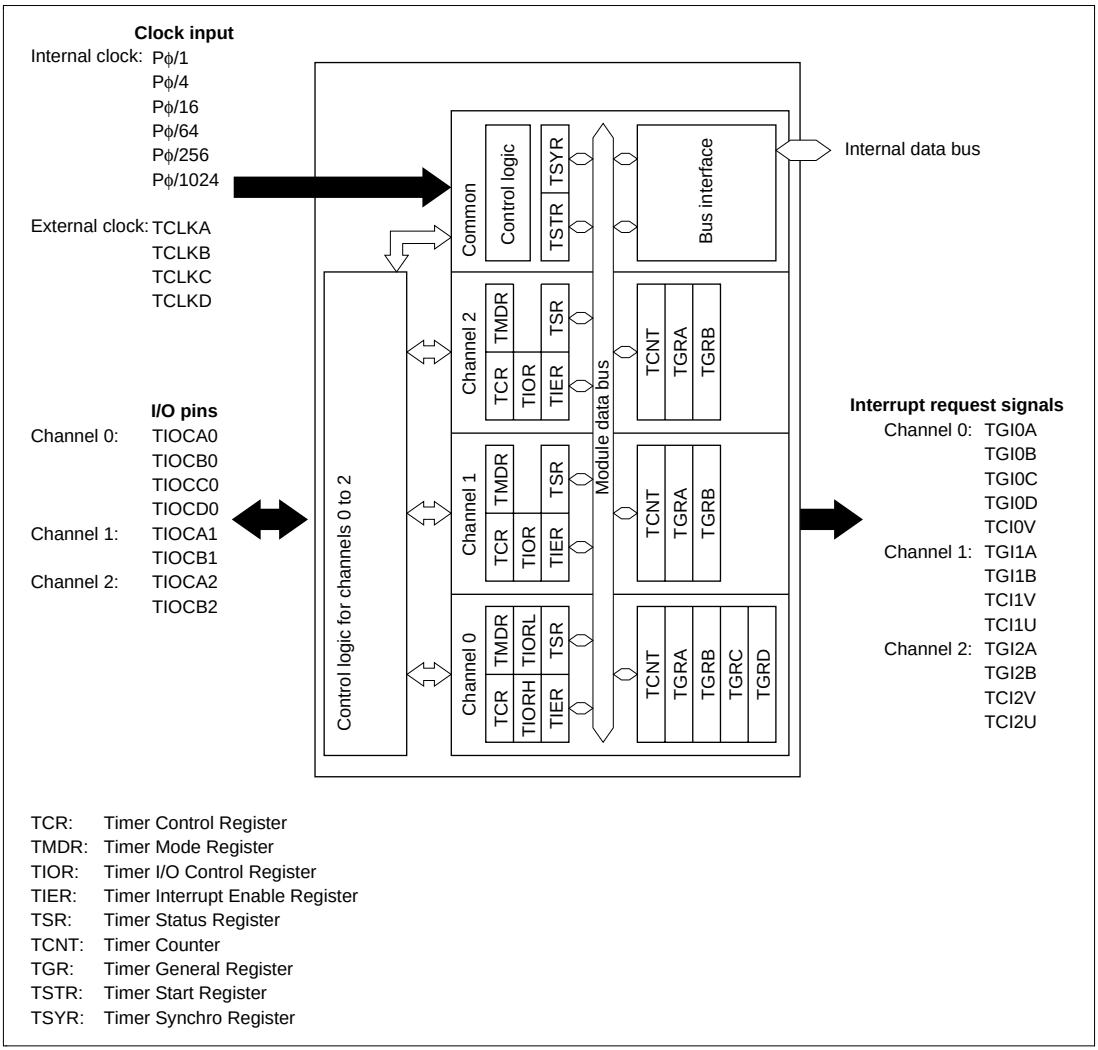



**Figure 18.1 TPU Block Diagram**

### 18.1.3 Pin Configuration

Table 18.2 shows the pin configuration of the TPU.

**Table 18.2 Pin Configuration**

| Channel | Name | Abbreviation | I/O | Function |
|---|---|---|---|---|
| All | Clock input A | TCLKA | Input | External clock A input pin (Channel 1 phase counting mode A phase input) |
| | Clock input B | TCLKB | Input | External clock B input pin (Channel 1 phase counting mode B phase input) |
| | Clock input C | TCLKC | Input | External clock C input pin (Channel 2 phase counting mode A phase input) |
| | Clock input D | TCLKD | Input | External clock D input pin (Channel 2 phase counting mode B phase input) |
| 0 | Input capture/output compare match A0 | TIOCA0 | I/O | TGR0A input capture input/output compare output/PWM output pin |
| | Input capture/output compare match B0 | TIOCB0 | I/O | TGR0B input capture input/output compare output/PWM output pin |
| | Input capture/output compare match C0 | TIOCC0 | I/O | TGR0C input capture input/output compare output/PWM output pin |
| | Input capture/output compare match D0 | TIOCD0 | I/O | TGR0D input capture input/output compare output/PWM output pin |
| 1 | Input capture/output compare match A1 | TIOCA1 | I/O | TGR1A input capture input/output compare output/PWM output pin |
| | Input capture/output compare match B1 | TIOCB1 | I/O | TGR1B input capture input/output compare output/PWM output pin |
| 2 | Input capture/output compare match A2 | TIOCA2 | I/O | TGR2A input capture input/output compare output/PWM output pin |
| | Input capture/output compare match B2 | TIOCB2 | I/O | TGR2B input capture input/output compare output/PWM output pin |

### 18.1.4 Register Configuration

Table 18.3 shows the register configuration of the TPU.

**Table 18.3 Register Configuration**

| Channel | Name | Abbreviation | R/W | Initial Value | Address | Access size (Bits) |
|---|---|---|---|---|---|---|
| 0 | Timer control register 0 | TCR0 | R/W | H'00 | H'FFFFFC50 | 8,16 |
| | Timer mode register 0 | TMDR0 | R/W | H'C0 | H'FFFFFC51 | 8,16 |
| | Timer I/O control register 0H | TIOR0H | R/W | H'00 | H'FFFFFC52 | 8,16 |
| | Timer I/O control register 0L | TIOR0L | R/W | H'00 | H'FFFFFC53 | 8,16 |
| | Timer interrupt enable register 0 | TIER0 | R/W | H'40 | H'FFFFFC54 | 8,16 |
| | Timer status register 0 | TSR0 | R/(W)* | H'C0 | H'FFFFFC55 | 8,16 |
| | Timer counter 0 | TCNT0 | R/W | H'0000 | H'FFFFFC56 | 16 |
| | Timer general register 0A | TGR0A | R/W | H'FFFF | H'FFFFFC58 | 16 |
| | Timer general register 0B | TGR0B | R/W | H'FFFF | H'FFFFFC5A | 16 |
| | Timer general register 0C | TGR0C | R/W | H'FFFF | H'FFFFFC5C | 16 |
| | Timer general register 0D | TGR0D | R/W | H'FFFF | H'FFFFFC5E | 16 |
| 1 | Timer control register 1 | TCR1 | R/W | H'00 | H'FFFFFC60 | 8,16 |
| | Timer mode register 1 | TMDR1 | R/W | H'C0 | H'FFFFFC61 | 8,16 |
| | Timer I/O control register 1 | TIOR1 | R/W | H'00 | H'FFFFFC62 | 8,16 |
| | Timer interrupt enable register 1 | TIER1 | R/W | H'40 | H'FFFFFC64 | 8,16 |
| | Timer status register 1 | TSR1 | R/(W)* | H'C0 | H'FFFFFC65 | 8,16 |
| | Timer counter 1 | TCNT1 | R/W | H'0000 | H'FFFFFC66 | 16 |
| | Timer general register 1A | TGR1A | R/W | H'FFFF | H'FFFFFC68 | 16 |
| | Timer general register 1B | TGR1B | R/W | H'FFFF | H'FFFFFC6A | 16 |

**Table 18.3 Register Configuration (cont)**

| Channel | Name | Abbreviation | R/W | Initial Value | Address | Access size (Bits) |
|---|---|---|---|---|---|---|
| 2 | Timer control register 2 | TCR2 | R/W | H'00 | H'FFFFFC70 | 8, 16 |
| | Timer mode register 2 | TMDR2 | R/W | H'C0 | H'FFFFFC71 | 8, 16 |
| | Timer I/O control register 2 | TIOR2 | R/W | H'00 | H'FFFFFC72 | 8, 16 |
| | Timer interrupt enable register 2 | TIER2 | R/W | H'40 | H'FFFFFC74 | 8, 16 |
| | Timer status register 2 | TSR2 | R/(W)* | H'C0 | H'FFFFFC75 | 8, 16 |
| | Timer counter 2 | TCNT2 | R/W | H'0000 | H'FFFFFC76 | 16 |
| | Timer general register 2A | TGR2A | R/W | H'FFFF | H'FFFFFC78 | 16 |
| | Timer general register 2B | TGR2B | R/W | H'FFFF | H'FFFFFC7A | 16 |

**Table 18.3 Register Configuration (cont)**

| Channel | Name | Abbreviation | R/W | nitial Value | Address | Access size (Bits) |
|---|---|---|---|---|---|---|
| All | Timer start register | TSTR | R/W | H'00 | H'FFFFFC40 | 8, 16 |
| | Timer synchro register | TSYR | R/W | H'00 | H'FFFFFC41 | 8, 16 |

Note: * Only 0 can be written, to clear the flags.

## 18.2 Register Descriptions

### 18.2.1 Timer Control Register (TCR)

**Channel 0: TCR0**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | CCLR2 | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Channel 1: TCR1**

**Channel 2: TCR2**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The TCR registers are 8-bit registers that control the TCNT channels. The TPU has three TCR registers, one for each of channels 0 to 2. The TCR registers are initialized to H'00 by a reset.

TCNT operation should be stopped when making TCR settings.

Bits 7, 6, 5—Counter Clear 2, 1, and 0 (CCLR2, CCLR1, CCLR0): These bits select the TCNT counter clearing source.

| Channel | Bit 7: CCLR2 | Bit 6: CCLR1 | Bit 5: CCLR0 | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | TCNT clearing disabled (Initial value) |
| | | | 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 |

| Channel | Bit 7: Reserved*[3] | Bit 6: CCLR1 | Bit 5: CCLR0 | Description |
|---|---|---|---|---|
| 1, 2 | 0 | 0 | 0 | TCNT clearing disabled (Initial value) |
| | | | 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 setting is performed 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.
3. Bit 7 is reserved in channels 1 and 2. It is always read as 0 and should only be written with 0.

Bits 4 and 3—Clock Edge 1 and 0 (CKEG1, CKEG0): These bits select the input clock edge. When a both-edges count is selected, a clock divided by two from the input clock can be selected. (e.g. Pϕ/4 both edges = Pϕ/2 rising edge). If phase counting mode is used on channels 1, and 2, this setting is ignored and the phase counting mode setting has priority.

| Bit 4: CKEG1 | Bit 3: CKEG0 | Description |
|---|---|---|
| 0 | 0 | Count at rising edge (Initial value) |
| | 1 | Count at falling edge |
| 1 | — | Count at both edges |

Note: Internal clock edge selection is valid when the input clock is Pϕ/4 or slower. If Pϕ/1 is selected for the input clock, this setting is ignored and a rising-edge count is selected.

Bits 2, 1, and 0—Time Prescaler 2, 1, and 0 (TPSC2 to TPSC0): These bits select the TCNT counter clock. The clock source can be selected independently for each channel. Table 18.4 shows the clock sources that can be set for each channel.

**Table 18.4 TPU Clock Sources**

| | Internal Clock | | | | | | External Clock | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Channel** | **Pϕ/1** | **Pϕ/4** | **Pϕ/16** | **Pϕ/64** | **Pϕ/256** | **Pϕ/1024** | **TCLKA** | **TCLKB** | **TCLKC** | **TCLKD** |
| 0 | O | O | O | O | | | O | O | O | O |
| 1 | O | O | O | O | O | | O | O | | |
| 2 | O | O | O | O | | O | O | O | O | |

Notes: O: Setting
Blank: No setting

| Channel | Bit 2: TPSC2 | Bit 1: TPSC1 | Bit 0: TPSC0 | Description | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Internal clock: counts on Pϕ/1 | (Initial value) |
| | | | 1 | Internal clock: counts on Pϕ/4 | |
| | | 1 | 0 | Internal clock: counts on Pϕ/16 | |
| | | | 1 | Internal clock: counts on Pϕ/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 | |

| Channel | Bit 2: TPSC2 | Bit 1: TPSC1 | Bit 0: TPSC0 | Description | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | Internal clock: counts on Pϕ/1 | (Initial value) |
| | | | 1 | Internal clock: counts on Pϕ/4 | |
| | | 1 | 0 | Internal clock: counts on Pϕ/16 | |
| | | | 1 | Internal clock: counts on Pϕ/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 Pϕ/256 | |
| | | | 1 | Setting prohibited | |

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

| Channel | Bit 2: TPSC2 | Bit 1: TPSC1 | Bit 0: TPSC0 | Description |
|---|---|---|---|---|
| 2 | 0 | 0 | 0 | Internal clock: counts on Pϕ/1 (Initial value) |
| | | | 1 | Internal clock: counts on Pϕ/4 |
| | | 1 | 0 | Internal clock: counts on Pϕ/16 |
| | | | 1 | Internal clock: counts on Pϕ/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 Pϕ/1024 |

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

### 18.2.2 Timer Mode Register (TMDR)

**Channel 0: TMDR0**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | BFB | BFA | MD3 | MD2 | MD1 | MD0 |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Channel 1: TMDR1**

**Channel 2: TMDR2**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | MD3 | MD2 | MD1 | MD0 |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W |

The TMDR registers are 8-bit readable/writable registers that are used to set the operating mode for each channel. The TPU has three TMDR registers, one for each channel. The TMDR registers are initialized to H'C0 by a reset.

TCNT operation should be stopped when making TMDR settings.

Bits 7 and 6—Reserved: These bits are always read as 1 and should only be written with 1.

Bit 5—Buffer Operation B (BFB): 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.

In channels 1, and 2, which have no TGRD, bit 5 is reserved. It is always read as 0 and should only be written with 0.

| Bit 5: BFB | Description | |
|---|---|---|
| 0 | TGRB operates normally | (Initial value) |
| 1 | TGRB and TGRD used together for buffer operation | |

Bit 4—Buffer Operation A (BFA): 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.

In channels 1, and 2, which have no TGRC, bit 4 is reserved. It is always read as 0 and should only be written with 0.

| Bit 4: BFA | Description | |
|---|---|---|
| 0 | TGRA operates normally | (Initial value) |
| 1 | TGRA and TGRC used together for buffer operation | |

Bits 3 to 0—Modes 3 to 0 (MD3 to MD0): These bits are used to set the timer operating mode.

| Bit 3: MD3*[1] | Bit 2: MD2*[2] | Bit 1: MD1 | Bit 0: MD0 | Description | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal operation | (Initial value) |
| | | | 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 | * | * | * | — | |

*: 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 channel 0. In this case, 0 should always be written to MD2.

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

**Channel 0: TIOR0H**

**Channel 1: TIOR1**

**Channel 2: TIOR2**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Channel 0: TIOR0L**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | IOD3 | IOD2 | IOD1 | IOD0 | IOC3 | IOC2 | IOC1 | IOC0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

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

The TIOR registers are 8-bit registers that control the TGR registers. The TPU has four TIOR registers, two for channel 0 and one each for channels 1, and 2. The TIOR registers are initialized to H'00 by a reset.

Note that TIOR is affected by the TMDR setting.

The initial output specified by TIOR becomes valid when the counter is halted (i.e. when the CST bit is cleared to 0 in TSTR). In PWM mode 2, the output at the point at which the counter is cleared to 0 is specified.

Bits 7 to 4— I/O Control B3 to B0 (IOB3 to IOB0)
I/O Control D3 to D0 (IOD3 to IOD0):
Bits IOB3 to IOB0 specify the function of TGRB.
Bits IOD3 to IOD0 specify the function of TGRD.

**TIOR0H**

| Channel | Bit 7: IOB3 | Bit 6: IOB2 | Bit 5: IOB1 | Bit 4: IOB0 | Description | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | TGR0B is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR0B is input capture register | Capture input source is TIOCB0 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don't care

**TIOR0L**

| Channel | Bit 7: IOD3 | Bit 6: IOD2 | Bit 5: IOD1 | Bit 4: IOD0 | Description | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | TGR0D is output compare register*[1] | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR0D is input capture register*[1] | Capture input source is TIOCD0 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don't care

Note: 1. When the BFB bit in TMDR0 is set to 1 and TGR0D is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**TIOR1**

| Channel | Bit 7: IOB3 | Bit 6: IOB2 | Bit 5: IOB1 | Bit 4: IOB0 | Description | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | TGR1B is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR1B is input capture register | Capture input source is TIOCB1 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don't care

**TIOR2**

| Channel | Bit 7: IOB3 | Bit 6: IOB2 | Bit 5: IOB1 | Bit 4: IOB0 | Description | | |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 0 | 0 | 0 | TGR2B is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | * | 0 | 0 | TGR2B is input capture register | Capture input source is TIOCB2 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |

*: Don't care

Bits 3 to 0— I/O Control A3 to A0 (IOA3 to IOA0)
I/O Control C3 to C0 (IOC3 to IOC0):
IOA3 to IOA0 specify the function of TGRA.
IOC3 to IOC0 specify the function of TGRC.

**TIOR0H**

| Channel | Bit 3: IOA3 | Bit 2: IOA2 | Bit 1: IOA1 | Bit 0: IOA0 | Description | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | TGR0A is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR0A is input capture register | Capture input source is TIOCA0 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don’t care

| Channel | Bit 3: IOC3 | Bit 2: IOC2 | Bit 1: IOC1 | Bit 0: IOC0 | Description | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | TGR0C is output compare register*[1] | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR0C is input capture register*[1] | Capture input source is TIOCC0 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don't care

Note: 1. When the BFA bit in TMDR0 is set to 1 and TGR0C is used as a buffer register, this setting is invalid and input capture/output compare is not generated.

**TIOR1**

| Channel | Bit 3: IOA3 | Bit 2: IOA2 | Bit 1: IOA1 | Bit 0: IOA0 | Description | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | TGR1A is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | 0 | 0 | 0 | TGR1A is input capture register | Capture input source is TIOCA1 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |
| | | 1 | * | * | | Setting prohibited | |

*: Don't care

**TIOR2**

| Channel | Bit 3: IOA3 | Bit 2: IOA2 | Bit 1: IOA1 | Bit 0: IOA0 | Description | | |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 0 | 0 | 0 | TGR2A is output compare register | Output disabled | (Initial value) |
| | | | | 1 | | Initial output is 0 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | | 1 | 0 | 0 | | Output disabled | |
| | | | | 1 | | Initial output is 1 output | 0 output at compare match |
| | | | 1 | 0 | | | 1 output at compare match |
| | | | | 1 | | | Toggle output at compare match |
| | 1 | * | 0 | 0 | TGR2A is input capture register | Capture input source is TIOCA2 pin | Input capture at rising edge |
| | | | | 1 | | | Input capture at falling edge |
| | | | 1 | * | | | Input capture at both edges |

*: Don't care

### 18.2.4 Timer Interrupt Enable Register (TIER)

**Channel 0: TIER0**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |
| Initial value: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Channel 1: TIER1**

**Channel 2: TIER2**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| Initial value: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R/W | R/W |

The TIER registers are 8-bit registers that control enabling or disabling of interrupt requests for each channel. The TPU has three TIER registers, one for each channel. The TIER registers are initialized to H'40 by a reset.

Bit 7—Reserved: This bit is always read as 0. The write value should always be 0.

Bit 6—Reserved: This bit is always read as 1 and should only be written with 1.

Bit 5—Underflow Interrupt Enable (TCIEU): Enables or disables interrupt requests (TCIU) by the TCFU flag when the TCFU flag in TSR is set to 1 in channels 1 and 2.

In channel 0, bit 5 is reserved. It is always read as 0 and should only be written with 0.

| Bit 5: TCIEU | Description | |
|---|---|---|
| 0 | Interrupt requests (TCIU) by TCFU disabled | (Initial value) |
| 1 | Interrupt requests (TCIU) by TCFU enabled | |

Bit 4—Overflow Interrupt Enable (TCIEV): Enables or disables interrupt requests (TCIV) by the TCFV flag when the TCFV flag in TSR is set to 1.

| Bit 4: TCIEV | Description | |
|---|---|---|
| 0 | Interrupt requests (TCIV) by TCFV disabled | (Initial value) |
| 1 | Interrupt requests (TCIV) by TCFV enabled | |

Bit 3—TGR Interrupt Enable D (TGIED): Enables or disables interrupt requests (TGID) by the TGFD bit when the TGFD bit in TSR is set to 1 in channel 0.

In channels 1, and 2, bit 3 is reserved. It is always read as 0 and should only be written with 0.

| Bit 3: TGIED | Description | |
|---|---|---|
| 0 | Interrupt requests (TGID) by TGFD bit disabled | (Initial value) |
| 1 | Interrupt requests (TGID) by TGFD bit enabled | |

Bit 2—TGR Interrupt Enable C (TGIEC): Enables or disables interrupt requests (TGIC) by the TGFC bit when the TGFC bit in TSR is set to 1 in channel 0.

In channels 1, and 2, bit 2 is reserved. It is always read as 0 and should only be written with 0.

| Bit 2: TGIEC | Description | |
|---|---|---|
| 0 | Interrupt requests (TGIC) by TGFC bit disabled | (Initial value) |
| 1 | Interrupt requests (TGIC) by TGFC bit enabled | |

Bit 1—TGR Interrupt Enable B (TGIEB): Enables or disables interrupt requests (TGIB) by the TGFB bit when the TGFB bit in TSR is set to 1.

| Bit 1: TGIEB | Description | |
|---|---|---|
| 0 | Interrupt requests (TGIB) by TGFB bit disabled | (Initial value) |
| 1 | Interrupt requests (TGIB) by TGFB bit enabled | |

Bit 0—TGR Interrupt Enable A (TGIEA): Enables or disables interrupt requests (TGIA) by the TGFA bit when the TGFA bit in TSR is set to 1.

| Bit 0: TGIEA | Description | |
|---|---|---|
| 0 | Interrupt requests (TGIA) by TGFA bit disabled | (Initial value) |
| 1 | Interrupt requests (TGIA) by TGFA bit enabled | |

### 18.2.5 Timer Status Register (TSR)

**Channel 0: TSR0**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | TCFV | TGFD | TGFC | TGFB | TGFA |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* |

Note: * Only 0 can be written, to clear the flags.

**Channel 1: TSR1**

**Channel 2: TSR2**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/(W)* | R/(W)* | R | R | R/(W)* | R/(W)* |

Note: * Only 0 can be written, to clear the flags.

The TSR registers are 8-bit registers that indicate the status of each channel. The TPU has three TSR registers, one for each channel. The TSR registers are initialized to H'C0 by a reset.

Bit 7—Count Direction Flag (TCFD): Status flag that shows the direction in which TCNT counts in channels 1, and 2.

In channel 0, bit 7 is reserved. It is always read as 1 and should only be written with 1.

| Bit 7: TCFD | Description | |
|---|---|---|
| 0 | TCNT counts down | |
| 1 | TCNT counts up | (Initial value) |

Bit 6—Reserved: This bit is always read as 1 and should only be written with 1.

Bit 5—Underflow Flag (TCFU): Status flag that indicates that TCNT underflow has occurred when channels 1 and 2 are set to phase counting mode.

In channel 0, bit 5 is reserved. It is always read as 0 and should only be written with 0.

| Bit 5: TCFU | Description |
|---|---|
| 0 | [Clearing condition] (Initial value)<br>When 0 is written to TCFU after reading TCFU = 1 |
| 1 | [Setting condition]<br>When the TCNT value underflows (changes from H'0000 to H'FFFF) |

Bit 4—Overflow Flag (TCFV): Status flag that indicates that TCNT overflow has occurred.

| Bit 4: TCFV | Description |
|---|---|
| 0 | [Clearing condition] (Initial value)<br>When 0 is written to TCFV after reading TCFV = 1 |
| 1 | [Setting condition]<br>When the TCNT value overflows (changes from H'FFFF to H'0000 ) |

Bit 3—Input Capture/Output Compare Flag D (TGFD): Status flag that indicates the occurrence of TGRD input capture or compare match in channel 0.

In channels 1, and 2, bit 3 is reserved. It is always read as 0 and should only be written with 0.

| Bit 3: TGFD | Description |
|---|---|
| 0 | [Clearing conditions] (Initial value)<br>• When DMAC is activated by TGID interrupt while DRCR setting in DMAC is TGI0D<br>• When 0 is written to TGFD after reading TGFD = 1 |
| 1 | [Setting conditions]<br>• When TCNT = TGRD while TGRD is functioning as output compare register<br>• When TCNT value is transferred to TGRD by input capture signal while TGRD is functioning as input capture register |

Bit 2—Input Capture/Output Compare Flag C (TGFC): Status flag that indicates the occurrence of TGRC input capture or compare match in channel 0.

In channels 1, and 2, bit 2 is reserved. It is always read as 0 and should only be written with 0.

| Bit 2: TGFC | Description |
|---|---|
| 0 | [Clearing conditions] (Initial value)<br>• When DMAC is activated by TGIC interrupt while DRCR setting in DMAC is TGI0C<br>• When 0 is written to TGFC after reading TGFC = 1 |
| 1 | [Setting conditions]<br>• When TCNT = TGRC while TGRC is functioning as output compare register<br>• When TCNT value is transferred to TGRC by input capture signal while TGRC is functioning as input capture register |

Bit 1—Input Capture/Output Compare Flag B (TGFB): Status flag that indicates the occurrence of TGRB input capture or compare match.

| Bit 1: TGFB | Description |
|---|---|
| 0 | [Clearing conditions] (Initial value)<br>• When DMAC is activated by TGIB interrupt while DRCR setting in DMAC is TGI0B<br>• When 0 is written to TGFB after reading TGFB = 1 |
| 1 | [Setting conditions]<br>• When TCNT = TGRB while TGRB is functioning as output compare register<br>• When TCNT value is transferred to TGRB by input capture signal while TGRB is functioning as input capture register |

Bit 0—Input Capture/Output Compare Flag A (TGFA): Status flag that indicates the occurrence of TGRA input capture or compare match.

| Bit 0: TGFA | Description |
|---|---|
| 0 | [Clearing conditions] (Initial value)<br>• When DMAC is activated by TGIA interrupt while DRCR setting in DMAC is TGI0A<br>• When 0 is written to TGFA after reading TGFA = 1 |
| 1 | [Setting conditions]<br>• When TCNT = TGRA while TGRA is functioning as output compare register<br>• When TCNT value is transferred to TGRA by input capture signal while TGRA is functioning as input capture register |

### 18.2.6 Timer Counter (TCNT)

**Channel 0: TCNT0 (up-counter)**

**Channel 1: TCNT1 (up/down-counter*)**

**Channel 2: TCNT2 (up/down-counter*)**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note: These counters can be used as up/down-counters only in phase counting mode. In other cases they function as up-counters.

The TCNT registers are 16-bit counters. The TPU has three TCNT counters, one for each channel.

The TCNT counters are initialized to H'0000 by a reset.

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

### 18.2.7 Timer General Register (TGR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The TGR registers are 16-bit registers with a dual function as output compare and input capture registers. The TPU has 8 TGR registers, four for channel 0 and two each for channels 1, and 2. TGRC and TGRD for channel 0 can also be designated for operation as buffer registers*. The TGR registers are initialized to H'FFFF by a reset.

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

Note: * TGR buffer register combinations are TGRA–TGRC and TGRB–TGRD.

### 18.2.8 Timer Start Register (TSTR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | CST2 | CST1 | CST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

TSTR is an 8-bit readable/writable register that selects operation/stoppage for channels 0 to 2. TSTR is initialized to H'00 by a reset.

TCNT counter operation should be stopped when setting the operating mode in TMDR or the TCNT count clock in TCR.

Bits 7 to 3—Reserved: These bits are always read as 0 and should only be written with 0.

Bits 2 to 0—Counter Start 2 to 0 (CST2 to CST0): These bits select operation or stoppage for TCNT.

| Bit n: CSTn | Description | |
|---|---|---|
| 0 | TCNTn count operation is stopped | (Initial value) |
| 1 | TCNTn performs count operation | |

Note: n = 2 to 0

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.

### 18.2.9 Timer Synchro Register (TSYR)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | — | — | — | SYNC2 | SYNC1 | SYNC0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

TSYR is an 8-bit readable/writable register that selects independent operation or synchronous operation for the channel 0 to 2 TCNT counters. A channel performs synchronous operation when the corresponding bit in TSYR is set to 1.

TSYR is initialized to H'00 by a reset.

Bits 7 to 3—Reserved: These bits are always read as 0 and should only be written with 0.

Bits 2 to 0—Timer Synchro 2 to 0 (SYNC2 to SYNC0): These bits select whether operation is independent of or synchronized with other channels.

When synchronous operation is selected, synchronous presetting of multiple channels*[1], and synchronous clearing through counter clearing on another channel*[2] are possible.

| Bit n: SYNCn | Description | |
|---|---|---|
| 0 | TCNTn operates independently<br>TCNT presetting/clearing is unrelated to other channels | (Initial value) |
| 1 | TCNTn performs synchronous operation<br>TCNT synchronous presetting/synchronous clearing is possible | |

Notes: n = 2 to 0

1. To set synchronous operation, the SYNC bits for two channels at least must be set to 1.
2. To set synchronous clearing, in addition to the SYNC bit , the TCNT clearing source must also be set by means of bits CCLR2 to CCLR0 in TCR.

## 18.3 Interface to Bus Master

### 18.3.1 16-Bit Registers

TCNT and TGR are 16-bit registers. As the data bus to the bus master is 16 bits wide, these registers can be read and written to in 16-bit units.

These registers cannot be read or written to in 8-bit units; 16-bit access must always be used.

An example of 16-bit register access operation is shown in figure 18.2.

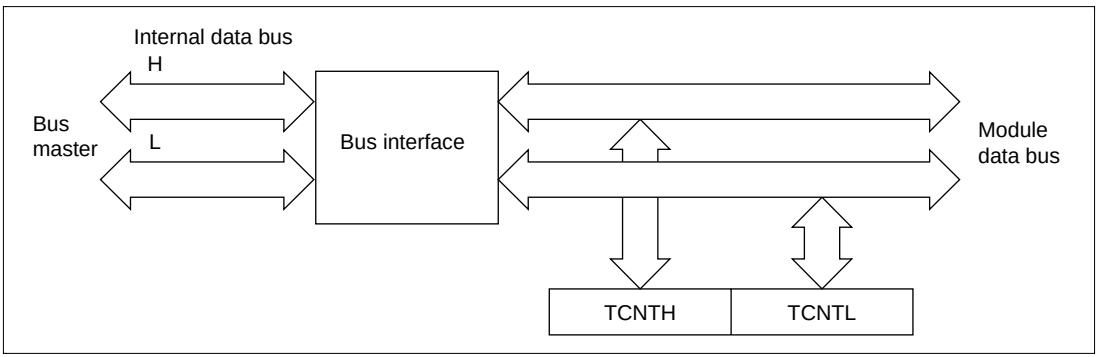



**Figure 18.2 16-Bit Register Access Operation [Bus Master ↔ TCNT (16 Bits)]**

### 18.3.2 8-Bit Registers

Registers other than TCNT and TGR are 8-bit. As the data bus to the CPU is 16 bits wide, these registers can be read and written to in 16-bit units. They can also be read and written to in 8-bit units.

Examples of 8-bit register access operation are shown in figures 18.3, 18.4, and 18.5.

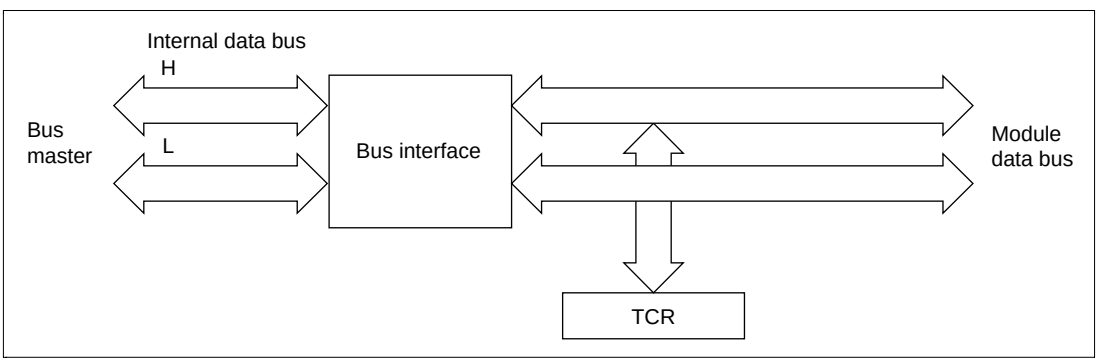



**Figure 18.3 8-Bit Register Access Operation [Bus Master ↔ TCR (Upper 8 Bits)]**

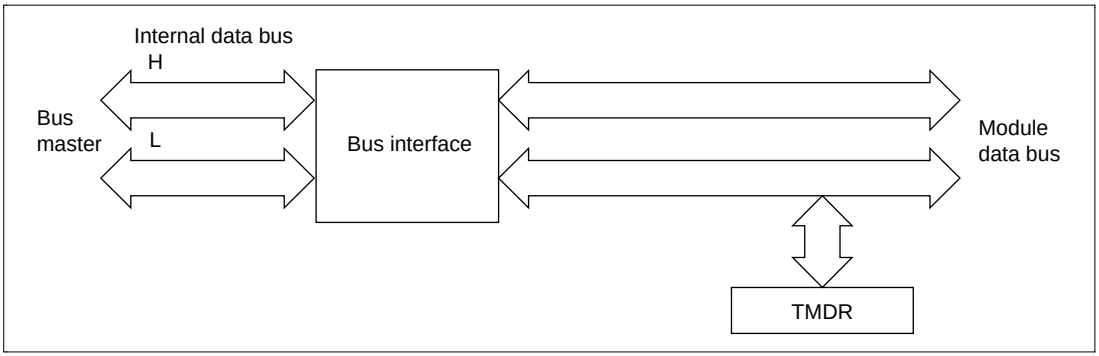



**Figure 18.4 8-Bit Register Access Operation [Bus Master ↔ TMDR (Lower 8 Bits)]**

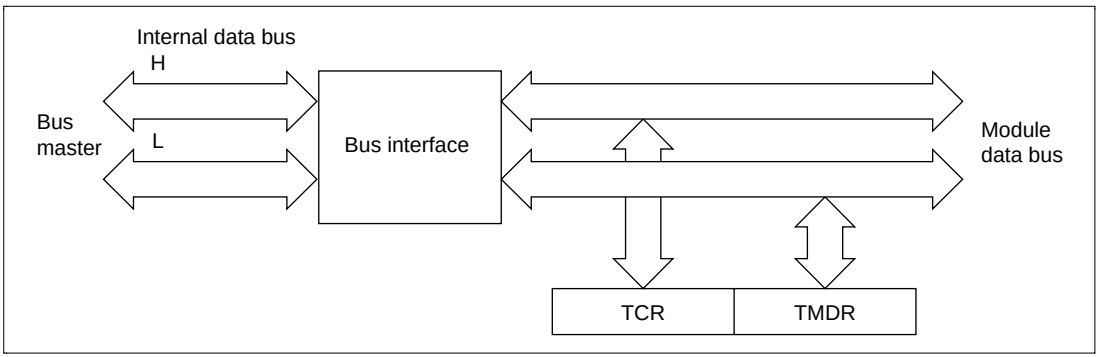



**Figure 18.5 8-Bit Register Access Operation [Bus Master ↔ TCR and TMDR (16 Bits)]**

## 18.4 Operation

### 18.4.1 Overview

Operation in each mode is outlined below.

**Normal Operation:** Each channel has a TCNT and TGR register. TCNT performs up-counting, and is also capable of free-running operation, synchronous counting, and external event counting.

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

**Synchronous Operation:** When synchronous operation is designated for a channel, TCNT for that channel performs synchronous presetting. That is, when TCNT for a channel designated for synchronous operation is rewritten, the TCNT counters for the other channels are also rewritten at the same time. Synchronous clearing of the TCNT counters is also possible by setting the timer synchronization bits in TSYR for channels designated for synchronous operation.

**Buffer Operation**

- When TGR is an output compare register

  When a compare match occurs, the value in the buffer register for the relevant channel is transferred to TGR.

- When TGR is an input capture register

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

**PWM Mode:** In this mode, a PWM waveform is output. The output level can be set by means of TIOR. A PWM waveform with a duty of between 0% and 100% can be output, according to the setting of each TGR register.

**Phase Counting Mode:** In this mode, TCNT is incremented or decremented by detecting the phases of two clocks input from the external clock input pins in channels 1, and 2. When phase counting mode is set, the corresponding TCLK pin functions as the clock input, and TCNT performs up- or down-counting.

This can be used for two-phase encoder pulse input.

### 18.4.2 Basic Functions

**Counter Operation:** When one of bits CST0 to CST2 is set to 1 in TSTR, the TCNT counter for the corresponding channel starts counting. TCNT can operate as a free-running counter, periodic counter, and so on.

- Example of count operation setting procedure

  Figure 18.6 shows an example of the count operation setting procedure.

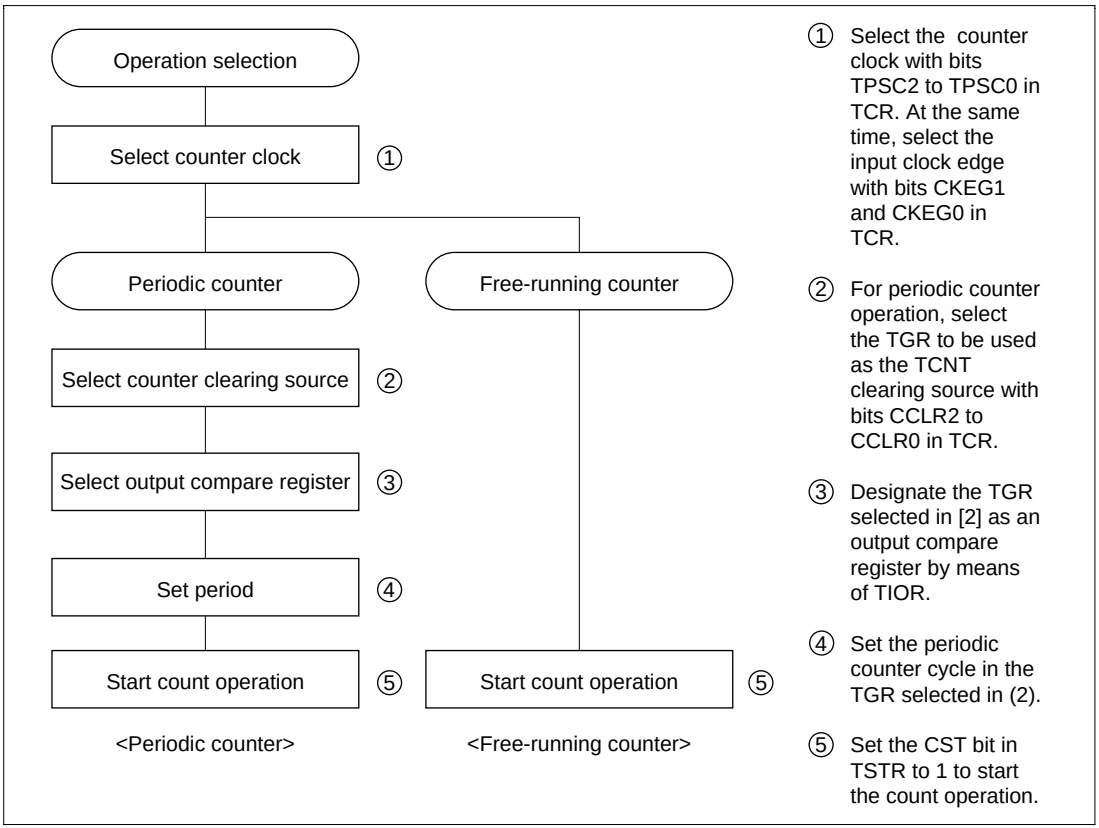



**Figure 18.6 Example of Counter Operation Setting Procedure**

- 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 18.7 illustrates free-running counter operation.

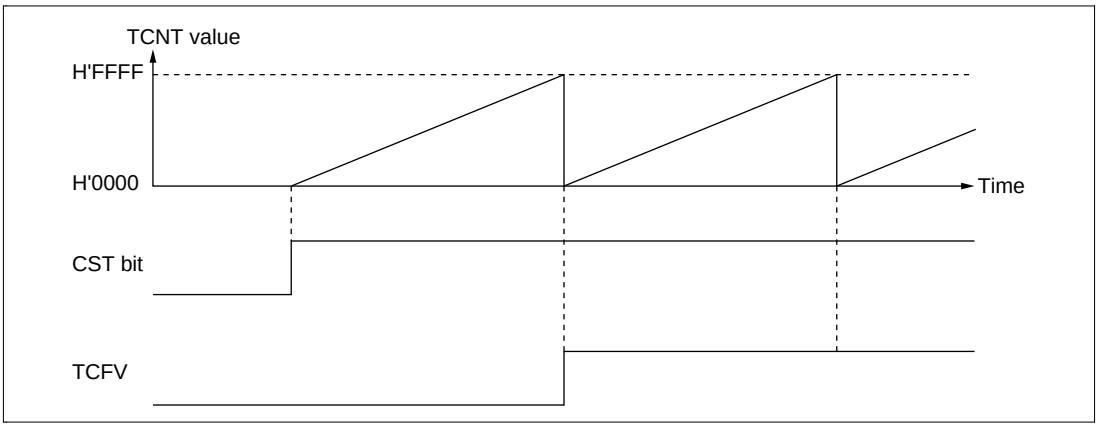



**Figure 18.7 Free-Running Counter Operation**

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 CCLR2 to CCLR0 in TCR. After the settings have been made, TCNT starts up-count operation as 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 18.8 illustrates periodic counter operation.

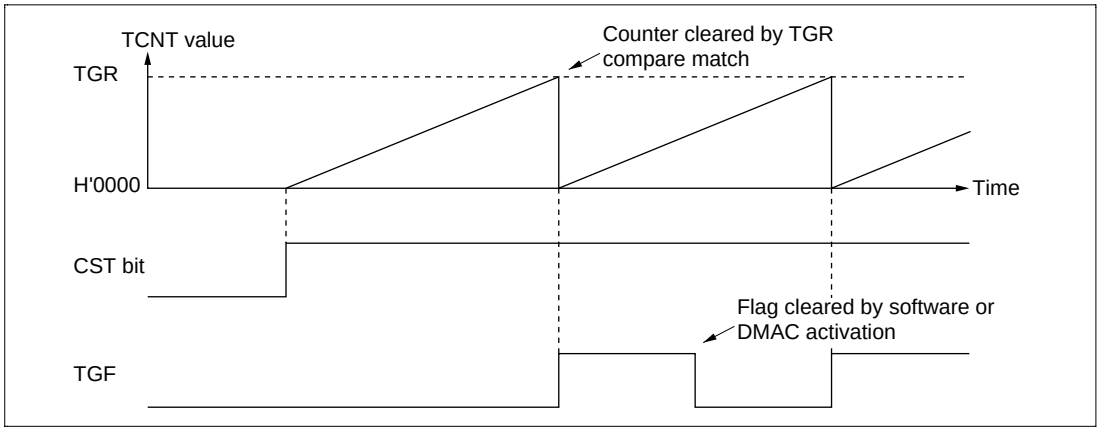



**Figure 18.8 Periodic Counter Operation**

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

- Example of setting procedure for waveform output by compare match

  Figure 18.9 shows an example of the setting procedure for waveform output by compare match

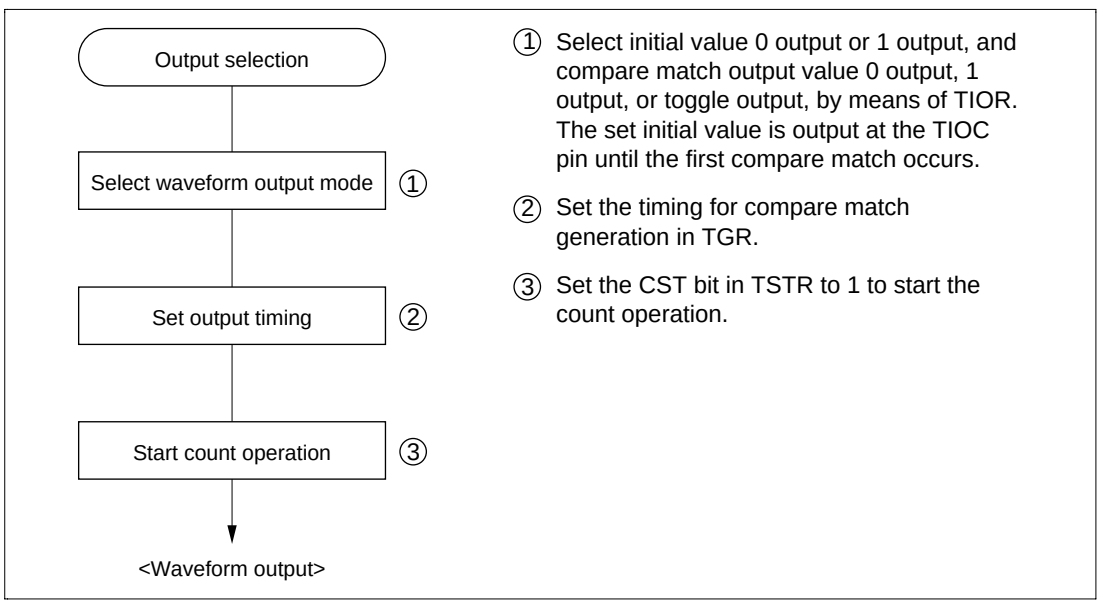



**Figure 18.9 Example of Setting Procedure for Waveform Output by Compare Match**

- Examples of waveform output operation

  Figure 18.10 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 so 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.

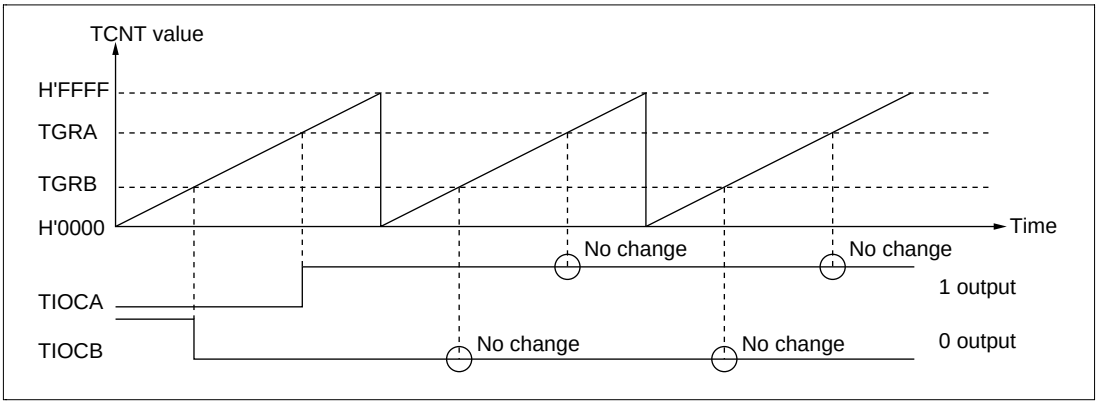



**Figure 18.10 Example of 0 Output/1 Output Operation**

Figure 18.11 shows an example of toggle output.

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

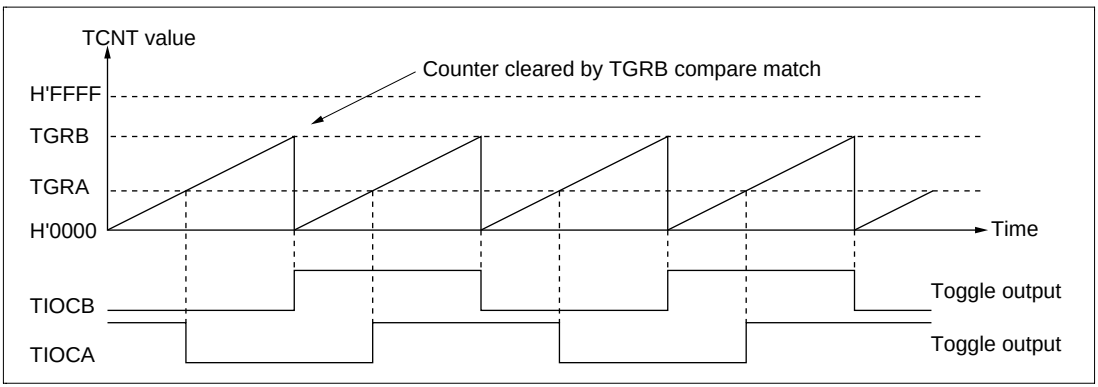



**Figure 18.11 Example of Toggle Output Operation**

**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.

- Example of input capture operation setting procedure

  Figure 18.12 shows an example of the input capture operation setting procedure.

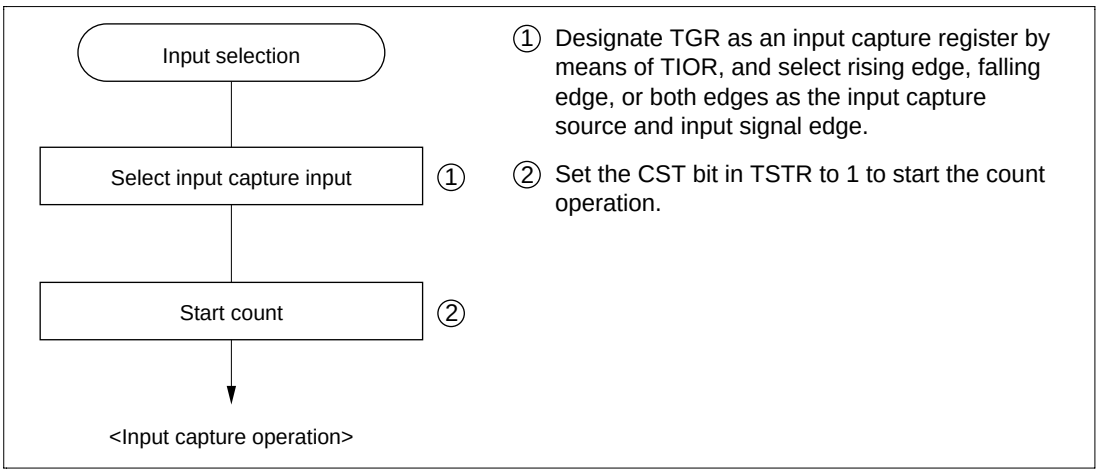



**Figure 18.12 Example of Input Capture Operation Setting Procedure**

- Example of input capture operation

  Figure 18.13 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, 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.

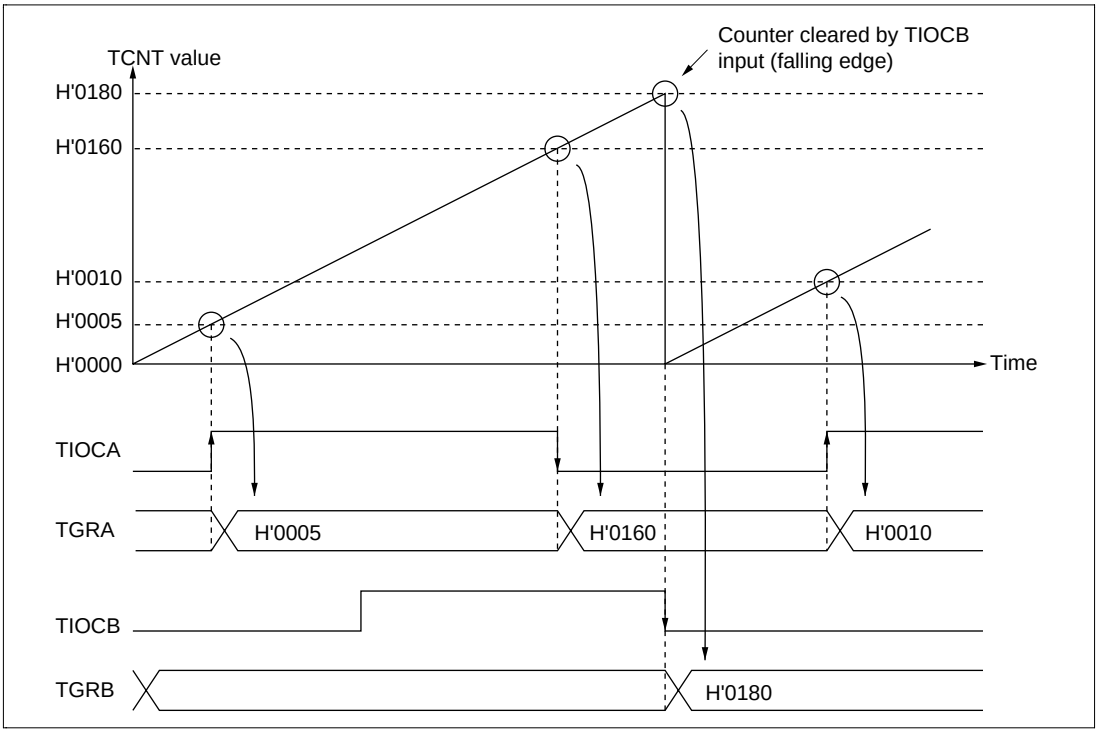



**Figure 18.13 Example of Input Capture Operation**

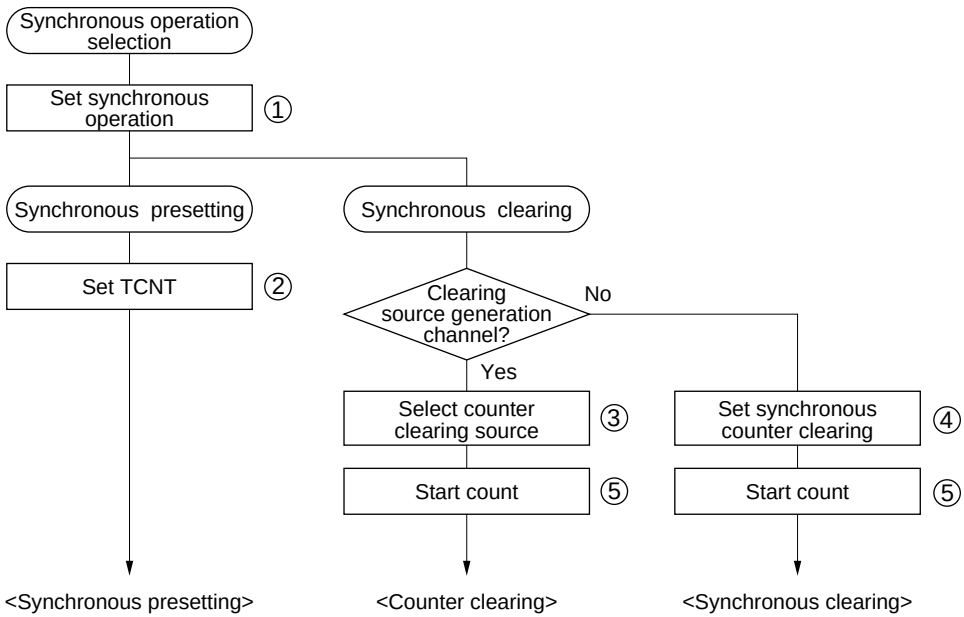
### 18.4.3 Synchronous Operation

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 2 can all be designated for synchronous operation.

**Example of Synchronous Operation Setting Procedure:** Figure 18.14 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 18.14 Example of Synchronous Operation Setting Procedure**

**Example of Synchronous Operation:** Figure 18.15 shows an example of synchronous operation.

In this example, synchronous operation and PWM mode 1 have been designated for channels 0 to 2, TGR0B 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 TIOC0A, TIOC1A, and TIOC2A. At this time, synchronous presetting, and synchronous clearing by TGR0B compare match, is performed for channel 0 to 2 TCNT counters, and the data set in TGR0B is used as the PWM cycle.

For details of PWM modes, see section 18.4.5, PWM Modes.

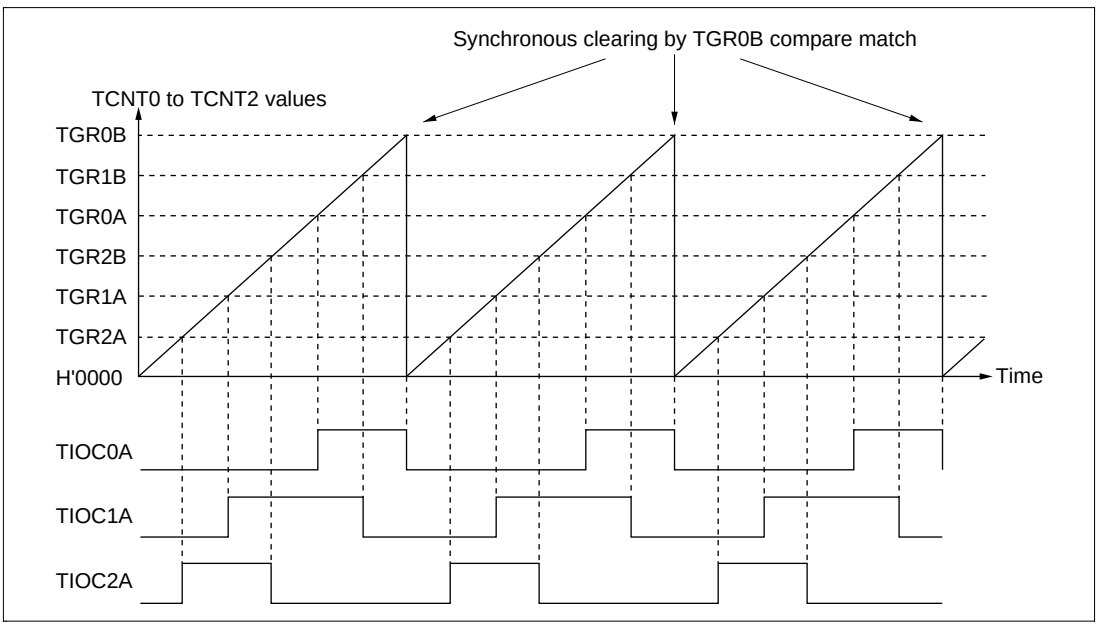



**Figure 18.15 Example of Synchronous Operation**

### 18.4.4 Buffer Operation

Buffer operation, provided for channel 0 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 18.5 shows the register combinations used in buffer operation.

**Table 18.5 Register Combinations in Buffer Operation**

| Channel | Timer General Register | Buffer Register |
|---|---|---|
| 0 | TGR0A | TGR0C |
| | TGR0B | TGR0D |

- 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 18.16.

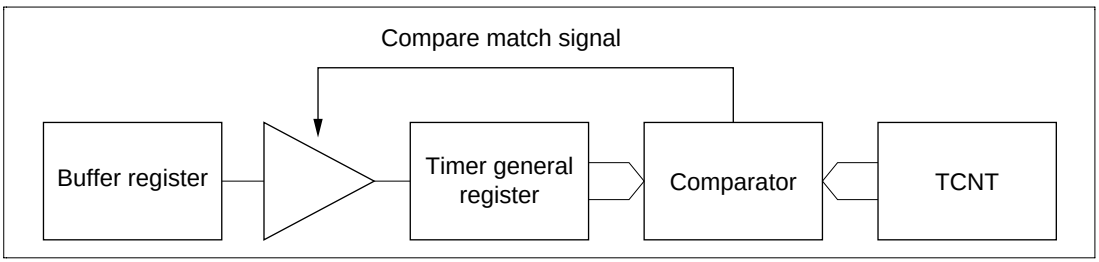



**Figure 18.16 Compare Match Buffer Operation**

- 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 18.17.

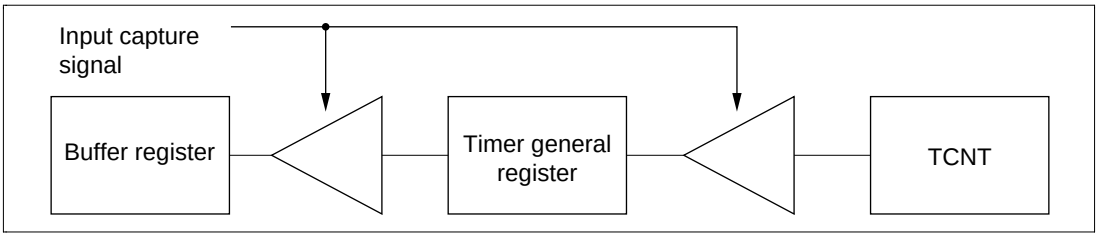



**Figure 18.17 Input Capture Buffer Operation**

**Example of Buffer Operation Setting Procedure:** Figure 18.18 shows an example of the buffer operation setting procedure.

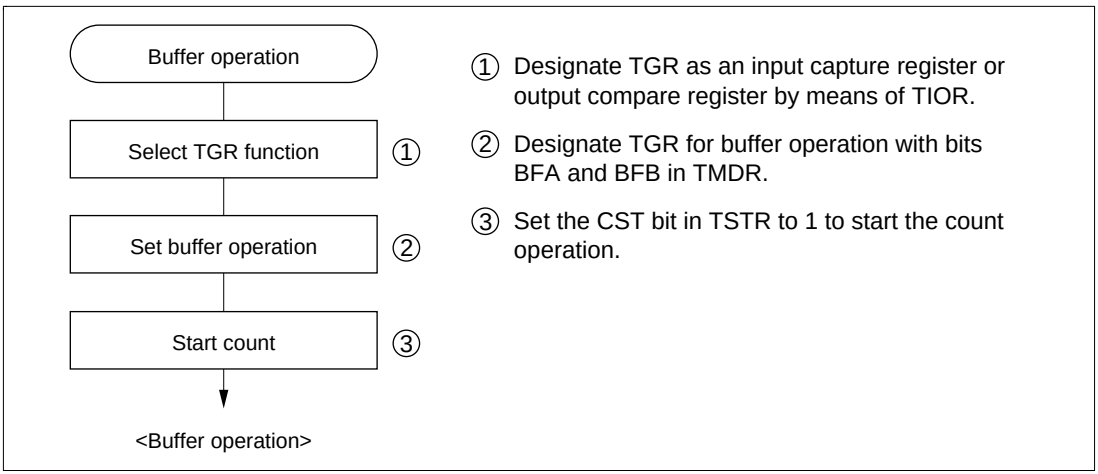



**Figure 18.18 Example of Buffer Operation Setting Procedure**

**Examples of Buffer Operation**

- When TGR is an output compare register

  Figure 18.19 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 compare match A occurs.

  For details of PWM modes, see section 18.4.5, PWM Modes.

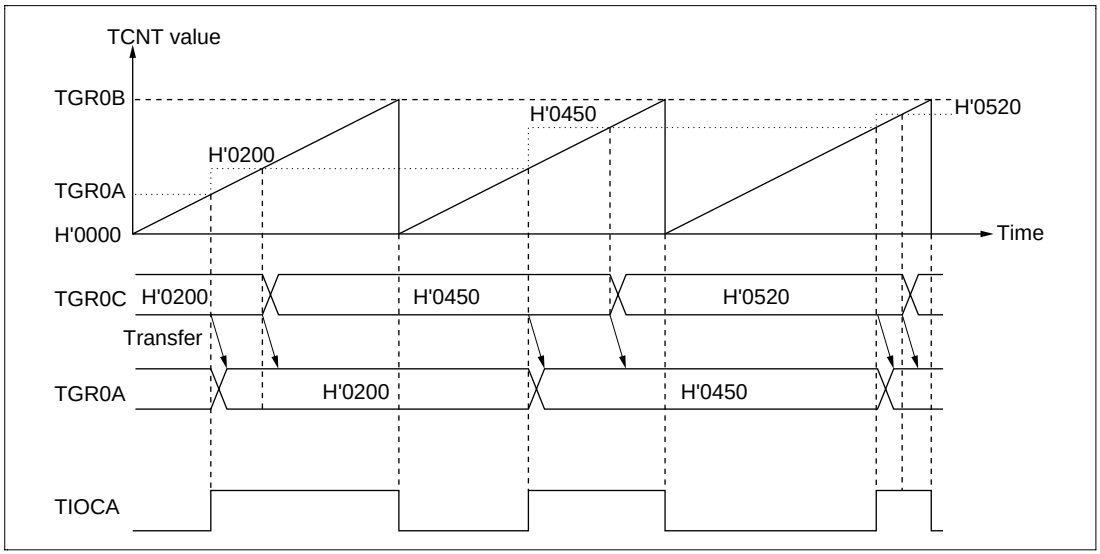



**Figure 18.19 Example of Buffer Operation (1)**

- When TGR is an input capture register

  Figure 18.20 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 occurrence of input capture A, the value previously stored in TGRA is simultaneously transferred to TGRC.

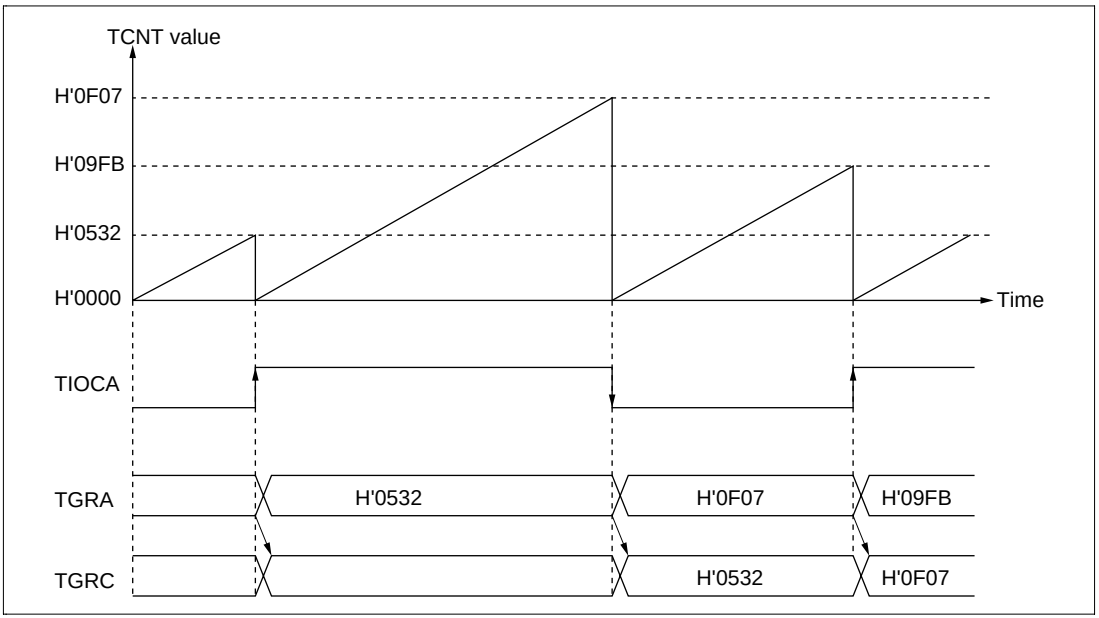



**Figure 18.20 Example of Buffer Operation (2)**

### 18.4.5 PWM Modes

In PWM mode, PWM waveforms are output from the output pins. 0, 1, or toggle output can be selected as the output level in response to compare match of each TGR.

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 by pairing TGRA with TGRB and TGRC with TGRD. The output specified by bits IOA3–IOA0 and IOC3–IOC0 in TIOR is performed in response to compare match A and C, and the output specified by bits IOB3–IOB0 and IOD3–IOD0 in TIOR in response to compare match B and D, from pins TIOCA and TIOCC. 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 4-phase PWM output is possible.

- PWM mode 2

  PWM output is generated using one TGR as the cycle register and the others as duty registers. The output specified by TIOR is performed in response to a compare match. Also, when the counter is cleared by a synchronization register compare match, pin output values are the initial values set in TIOR. If the set values of the period and duty registers are identical, the output value does not change when a compare match occurs.

  In PWM mode 2, a maximum 7-phase PWM output is possible by combined use with synchronous operation.

The correspondence between PWM output pins and registers is shown in table 18.6.

**Table 18.6 PWM Output Registers and Output Pins**

| | | Output Pins | |
|---|---|---|---|
| **Channel** | **Registers** | **PWM Mode 1** | **PWM Mode 2** |
| 0 | TGR0A | TIOCA0 | TIOCA0 |
| | TGR0B | | TIOCB0 |
| | TGR0C | TIOCC0 | TIOCC0 |
| | TGR0D | | TIOCD0 |
| 1 | TGR1A | TIOCA1 | TIOCA1 |
| | TGR1B | | TIOCB1 |
| 2 | TGR2A | TIOCA2 | TIOCA2 |
| | TGR2B | | TIOCB2 |

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 18.21 shows an example of the PWM mode setting procedure.

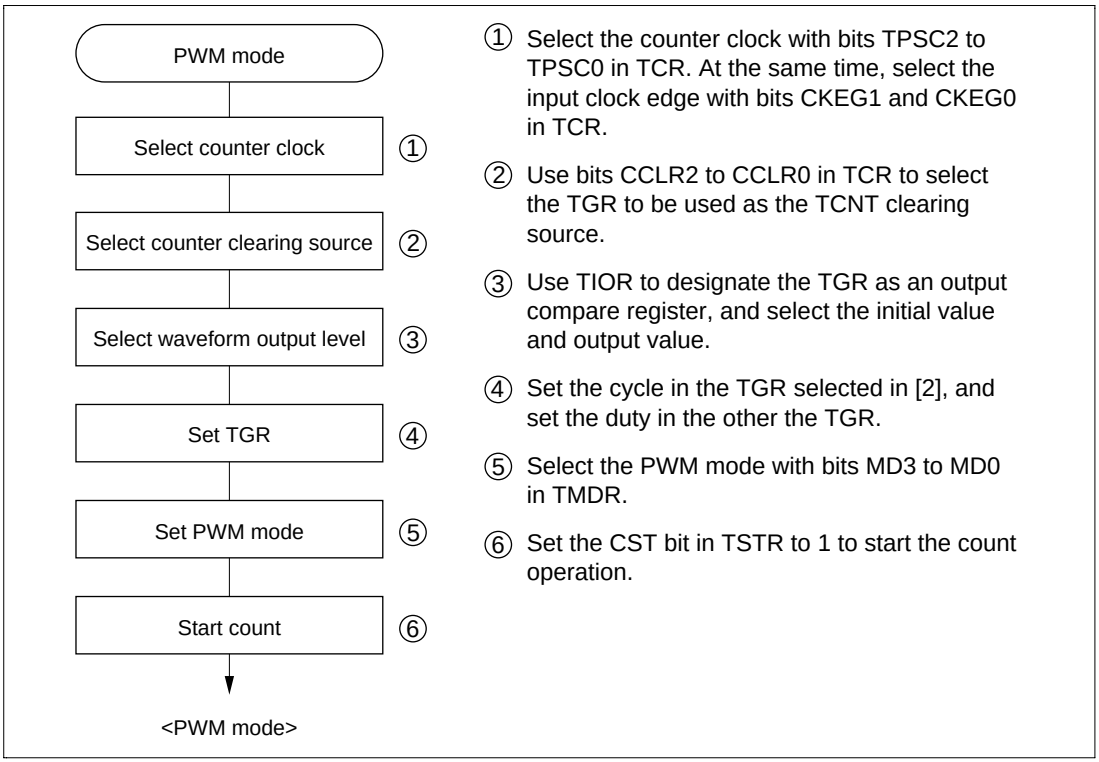



**Figure 18.21 Example of PWM Mode Setting Procedure**

**Examples of PWM Mode Operation:** Figure 18.22 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 output 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 TGRB registers as the duty.

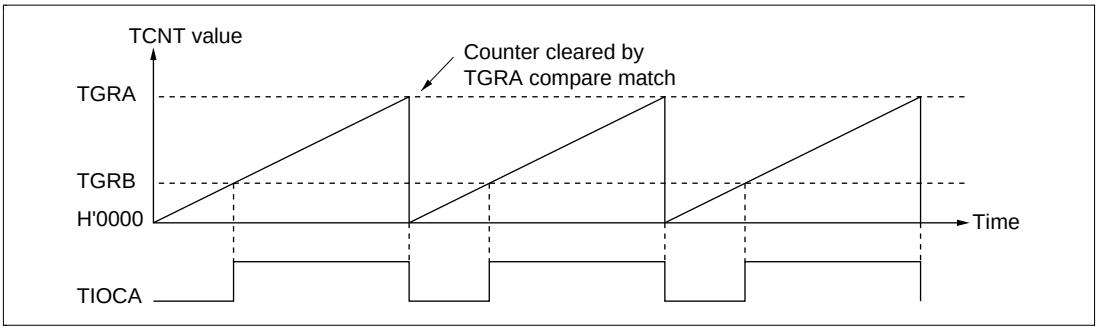



**Figure 18.22 Example of PWM Mode Operation (1)**

Figure 18.23 shows an example of PWM mode 2 operation.

In this example, synchronous operation is designated for channels 0 and 1, TGR1B 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, to output a 5-phase PWM waveform.

In this case, the value set in TGR1B is used as the cycle, and the values set in the other TGRs as the duty.

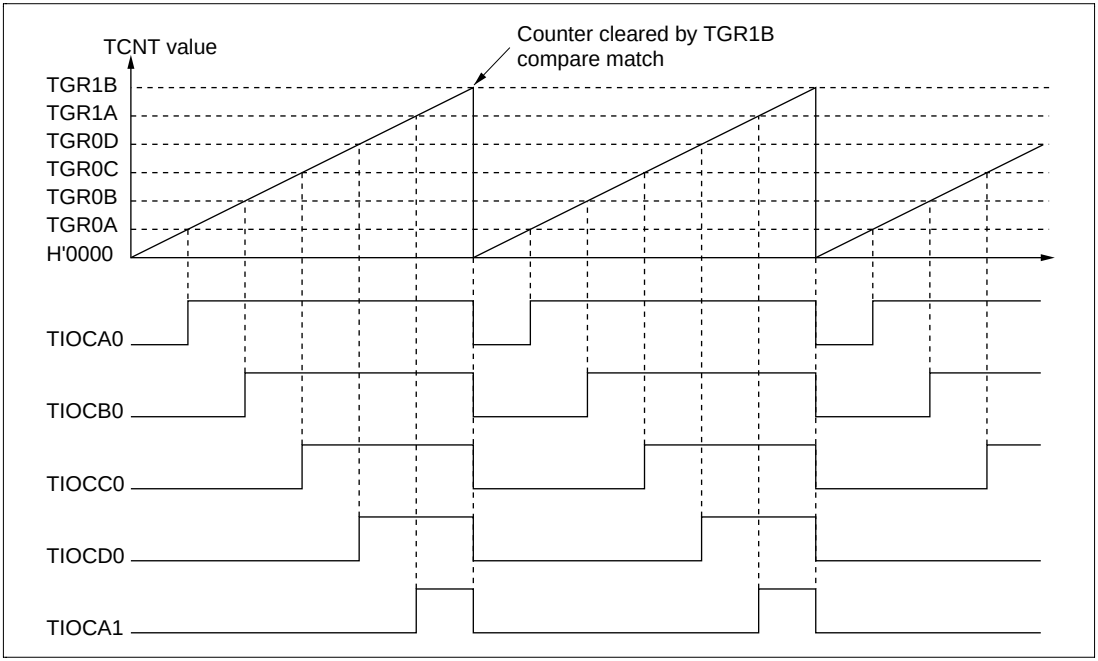



**Figure 18.23 Example of PWM Mode Operation (2)**

Figure 18.24 shows examples of PWM waveform output with 0% duty and 100% duty in PWM mode.

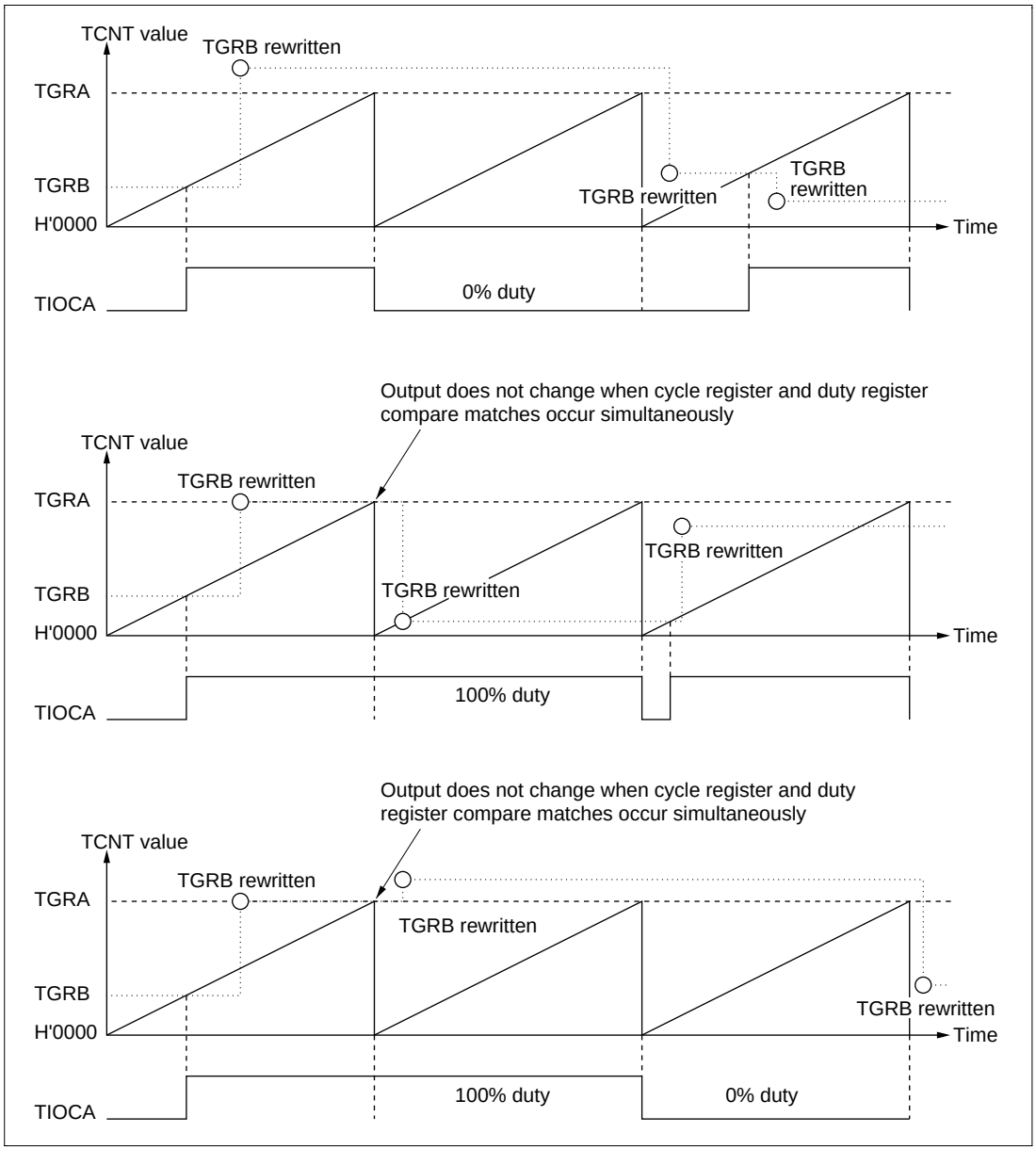



**Figure 18.24 Example of PWM Mode Operation (3)**

### 18.4.6 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, and 2.

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 TPSC2 to TPSC0 and bits CKEG1 and CKEG0 in TCR. However, the functions of bits CCLR1 and CCLR0 in TCR, and of TIOR, TIER, and TGR are valid, and input capture/compare match and interrupt functions can be used.

When overflow occurs while TCNT is counting up, the TCFV flag in TSR is set; when underflow occurs while TCNT is counting down, the TCFU flag is set.

The TCFD bit in TSR is the count direction flag. Reading the TCFD flag provides an indication of whether TCNT is counting up or down.

Table 18.7 shows the correspondence between external clock pins and channels.

**Table 18.7 Phase Counting Mode Clock Input Pins**

| | External Clock Pins | |
|---|---|---|
| **Channels** | **A-Phase** | **B-Phase** |
| When channel 1 is set to phase counting mode | TCLKA | TCLKB |
| When channel 2 is set to phase counting mode | TCLKC | TCLKD |

**Example of Phase Counting Mode Setting Procedure:** Figure 18.25 shows an example of the phase counting mode setting procedure.

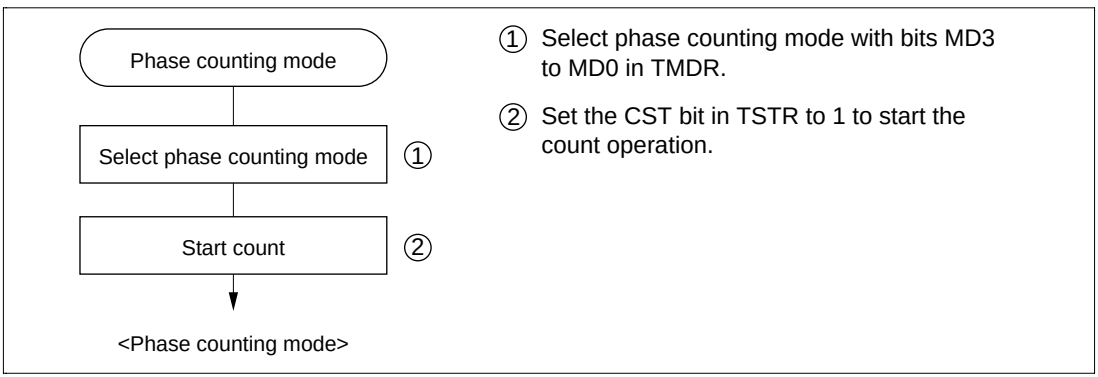



**Figure 18.25 Example of Phase Counting Mode Setting Procedure**

**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.

- Phase counting mode 1

  Figure 18.26 shows an example of phase counting mode 1 operation, and table 18.8 summarizes the TCNT up/down-count conditions.

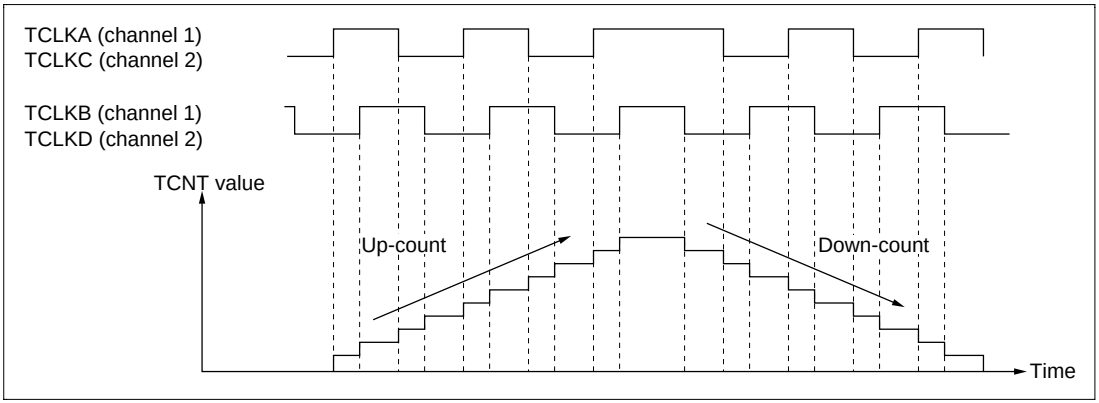



**Figure 18.26 Example of Phase Counting Mode 1 Operation**

**Table 18.8 Up/Down-Count Conditions in Phase Counting Mode 1**

| TCLKA (Channel 1)<br>TCLKC (Channel 2) | TCLKB (Channel 1)<br>TCLKD (Channel 2) | Operation |
|---|---|---|
| High level | ↑ | Up-count |
| Low level | ↓ | |
| ↑ | Low level | |
| ↓ | High level | |
| High level | ↓ | Down-count |
| Low level | ↑ | |
| ↑ | High level | |
| ↓ | Low level | |

Notes: ↑ : Rising edge
↓ : Falling edge

- Phase counting mode 2

  Figure 18.27 shows an example of phase counting mode 2 operation, and table 18.9 summarizes the TCNT up/down-count conditions.

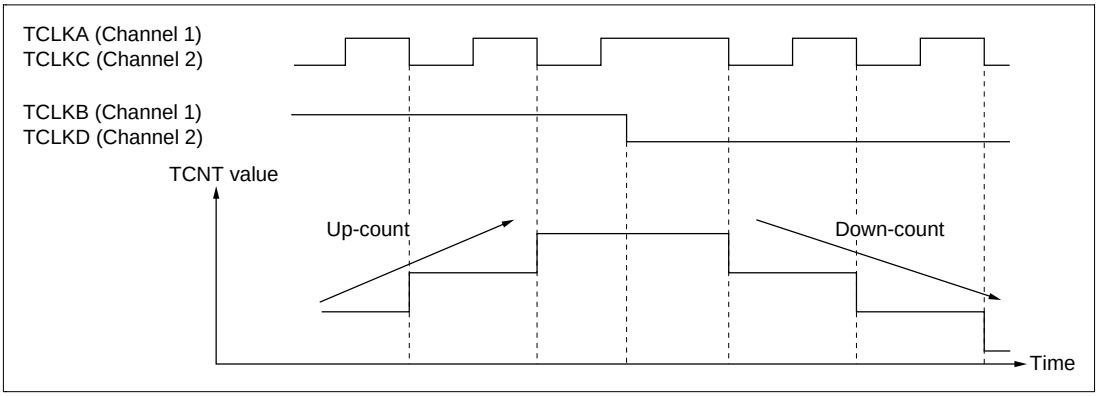



**Figure 18.27 Example of Phase Counting Mode 2 Operation**

**Table 18.9 Up/Down-Count Conditions in Phase Counting Mode 2**

| TCLKA (Channel 1)<br>TCLKC (Channel 2) | TCLKB (Channel 1)<br>TCLKD (Channel 2) | 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 |

Notes: ↑: Rising edge
↓: Falling edge

- Phase counting mode 3

  Figure 18.28 shows an example of phase counting mode 3 operation, and table 18.10 summarizes the TCNT up/down-count conditions.

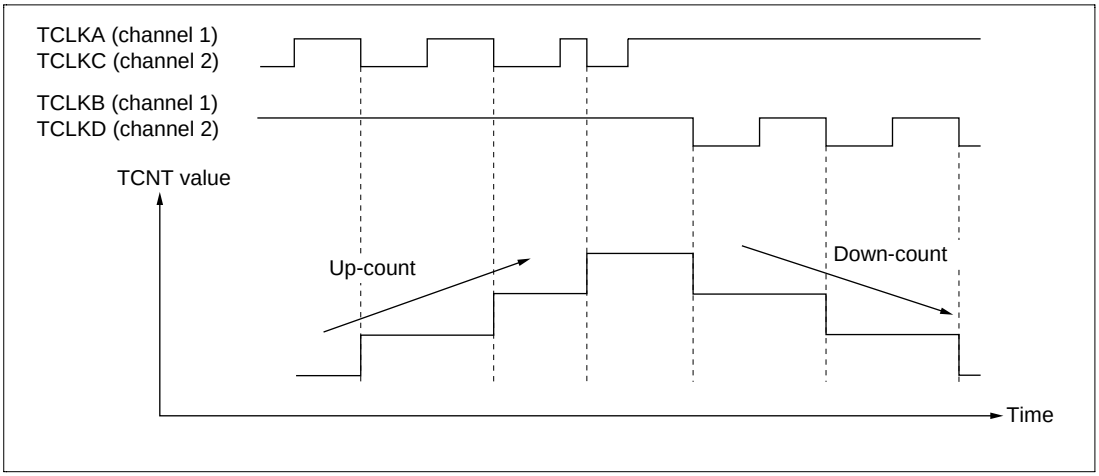



**Figure 18.28 Example of Phase Counting Mode 3 Operation**

**Table 18.10 Up/Down-Count Conditions in Phase Counting Mode 3**

| TCLKA (Channel 1)<br>TCLKC (Channel 2) | TCLKB (Channel 1)<br>TCLKD (Channel 2) | 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 |

Notes: ↑: Rising edge
↓: Falling edge

- Phase counting mode 4

  Figure 18.29 shows an example of phase counting mode 4 operation, and table 18.11 summarizes the TCNT up/down-count conditions.

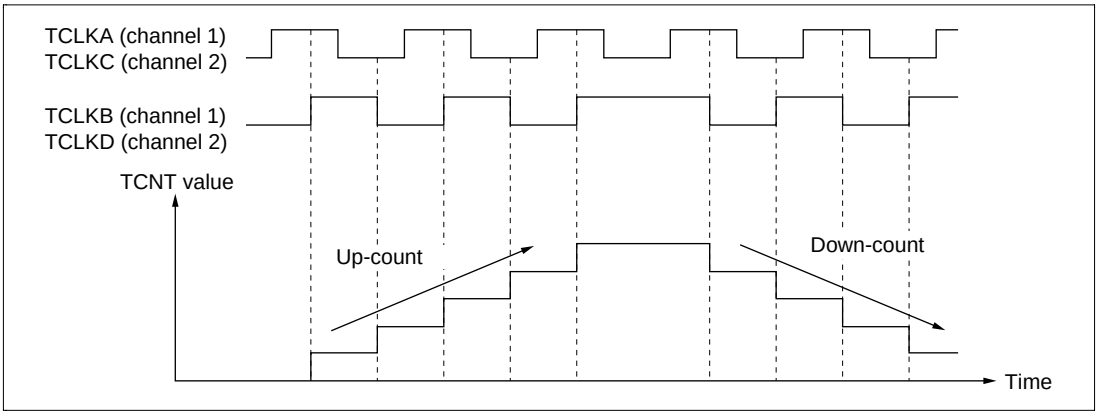



**Figure 18.29 Example of Phase Counting Mode 4 Operation**

**Table 18.11 Up/Down-Count Conditions in Phase Counting Mode 4**

| TCLKA (Channel 1)<br>TCLKC (Channel 2) | TCLKB (Channel 1)<br>TCLKD (Channel 2) | 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 | |

Notes: ↑: Rising edge
↓: Falling edge

## 18.5 Interrupts

### 18.5.1 Interrupt Sources and Priorities

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 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, but the priority order within a channel is fixed. For details, see section 5, Interrupt Controller (INTC).

Table 18.12 lists the TPU interrupt sources.

**Table 18.12 TPU Interrupts**

| Channel | Interrupt Source | Description | DMAC Activation | Priority |
|---|---|---|---|---|
| 0 | TGI0A | TGR0A input capture/compare match | Possible | High |
| | TGI0B | TGR0B input capture/compare match | Possible | ↑ |
| | TGI0C | TGR0C input capture/compare match | Possible | |
| | TGI0D | TGR0D input capture/compare match | Possible | |
| | TCI0V | TCNT0 overflow | Not possible | |
| 1 | TGI1A | TGR1A input capture/compare match | Not possible | |
| | TGI1B | TGR1B input capture/compare match | Not possible | |
| | TCI1V | TCNT1 overflow | Not possible | |
| | TCI1U | TCNT1 underflow | Not possible | |
| 2 | TGI2A | TGR2A input capture/compare match | Not possible | |
| | TGI2B | TGR2B input capture/compare match | Not possible | |
| | TCI2V | TCNT2 overflow | Not possible | ↓ |
| | TCI2U | TCNT2 underflow | Not possible | Low |

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 8 input capture/compare match interrupts, four for channel 0, and two each for channels 1, and 2.

**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 particular channel. The interrupt request is cleared by clearing the TCFV flag to 0. The TPU has three 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 channel. The interrupt request is cleared by clearing the TCFU flag to 0. The TPU has two underflow interrupts, one each for channels 1, and 2.

### 18.5.2 DMAC Activation

The DMAC can be activated by the TGR input capture/compare match interrupt for a channel. For details, see section 9, Direct Memory Access Controller (DMAC).

A total of four TPU input capture/compare match interrupts can be used as DMAC activation sources for channel 0.

## 18.6 Operation Timing

### 18.6.1 Input/Output Timing

**TCNT Count Timing:** Figure 18.30 shows TCNT count timing in internal clock operation, and figure 18.31 shows TCNT count timing in external clock operation.

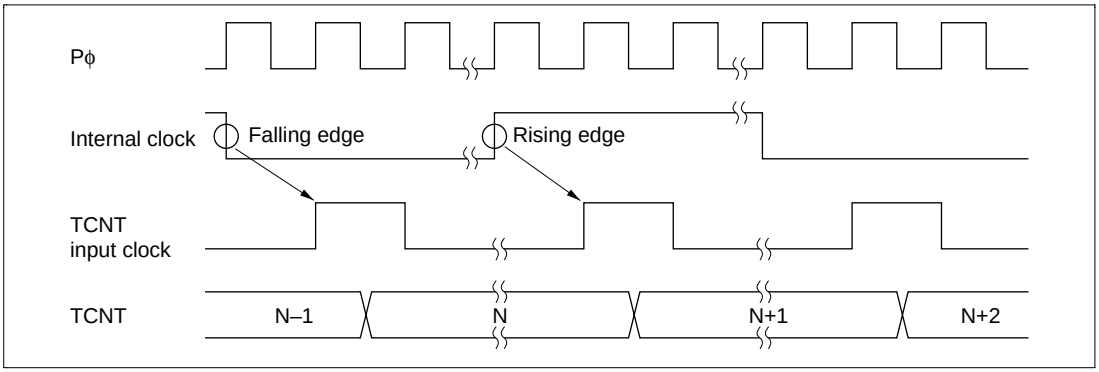



**Figure 18.30 Count Timing in Internal Clock Operation**

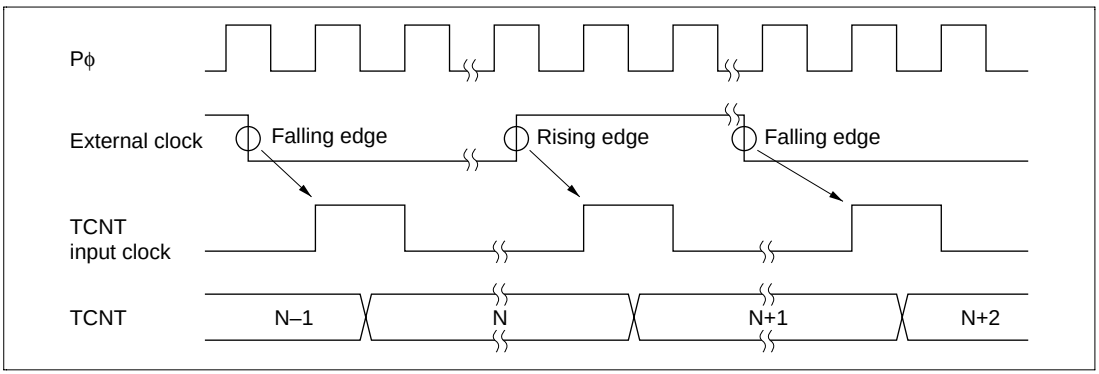



**Figure 18.31 Count Timing in External Clock Operation**

**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 (TIOC pin). After a match between TCNT and TGR, the compare match signal is not generated until the TCNT input clock is generated.

Figure 18.32 shows output compare output timing.

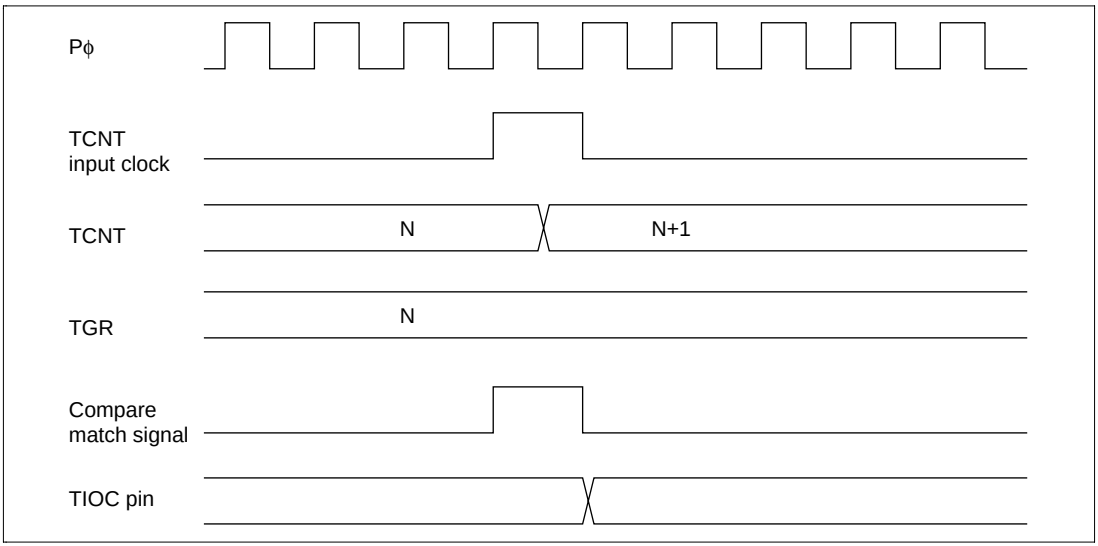



**Figure 18.32 Output Compare Output Timing**

**Input Capture Signal Timing:** Figure 18.33 shows input capture signal timing.

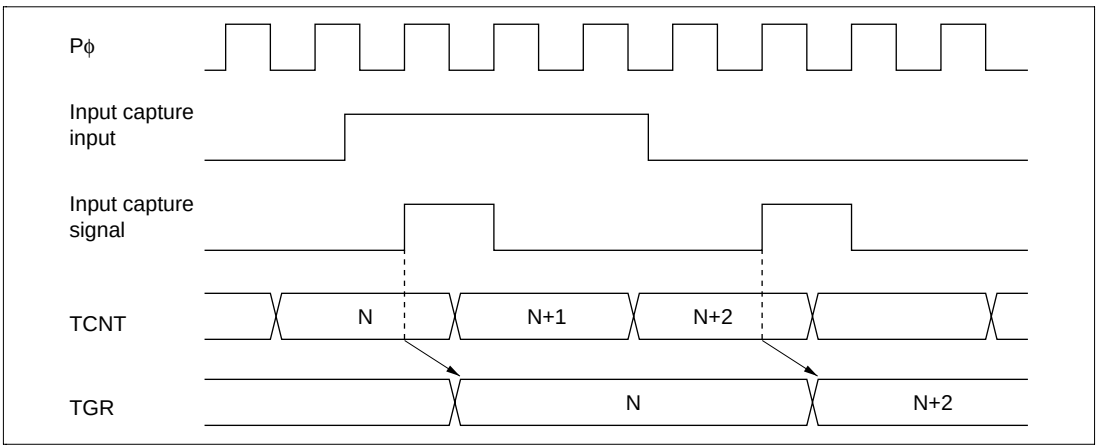



**Figure 18.33 Input Capture Input Signal Timing**

**Timing for Counter Clearing by Compare Match/Input Capture:** Figure 18.34 shows the timing when counter clearing by compare match occurrence is specified, and figure 18.35 shows the timing when counter clearing by input capture occurrence is specified.

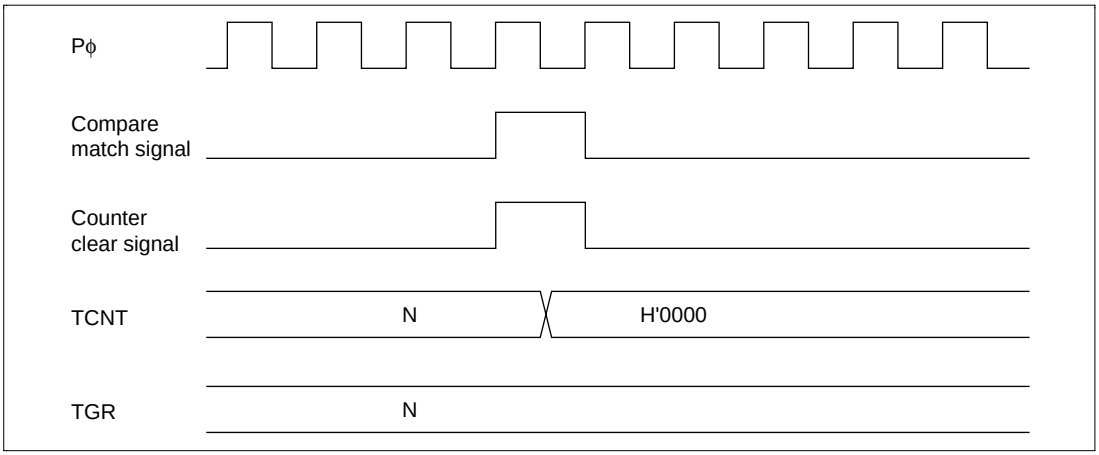



**Figure 18.34 Counter Clear Timing (Compare Match)**

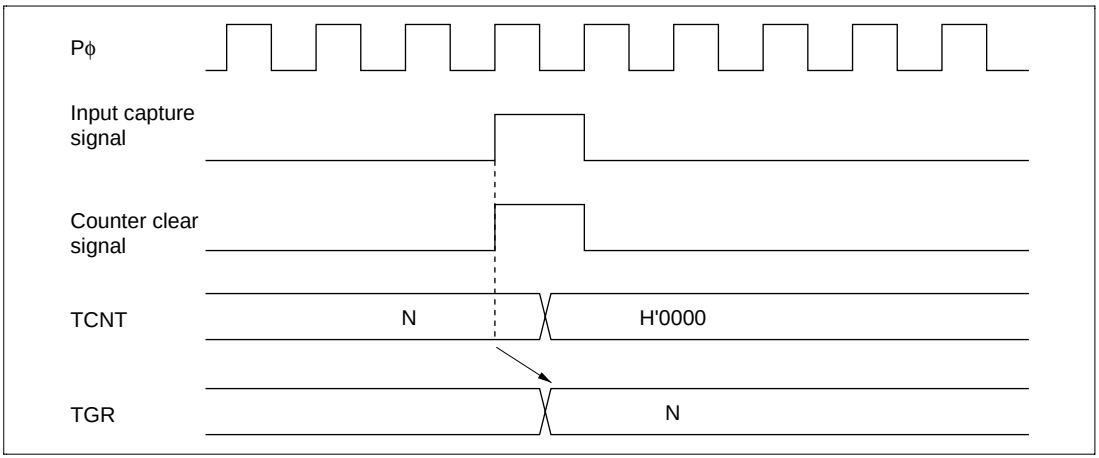



**Figure 18.35 Counter Clear Timing (Input Capture)**

**Buffer Operation Timing:** Figures 18.36 and 18.37 show the timing in buffer operation.

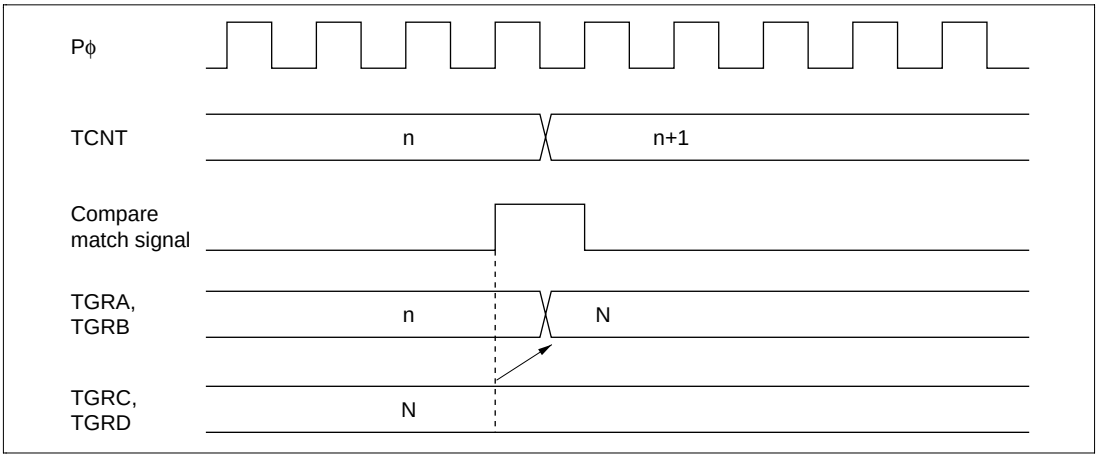



**Figure 18.36 Buffer Operation Timing (Compare Match)**

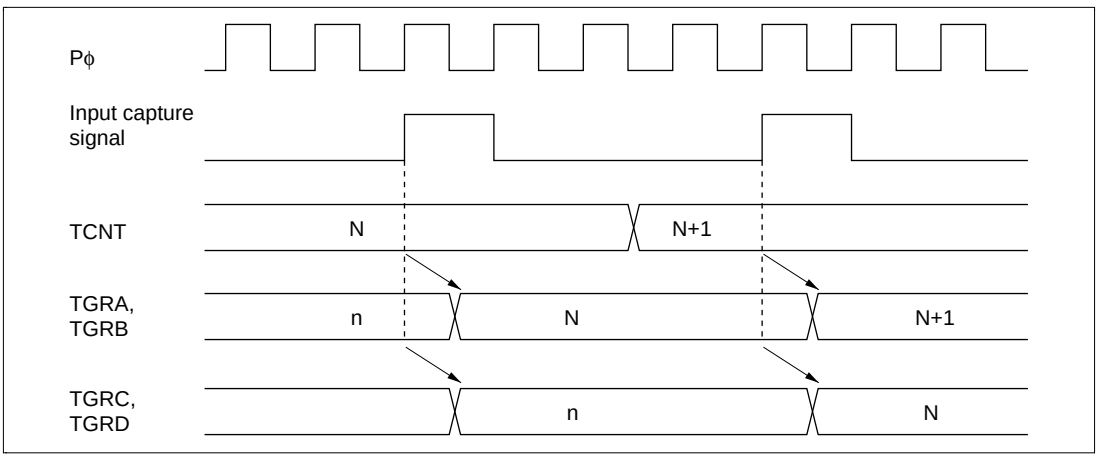



**Figure 18.37 Buffer Operation Timing (Input Capture)**

### 18.6.2 Interrupt Signal Timing

**TGF Flag Setting Timing in Case of Compare Match:** Figure 18.38 shows the timing for setting of the TGF flag in TSR by compare match occurrence, and TGI interrupt request signal timing.

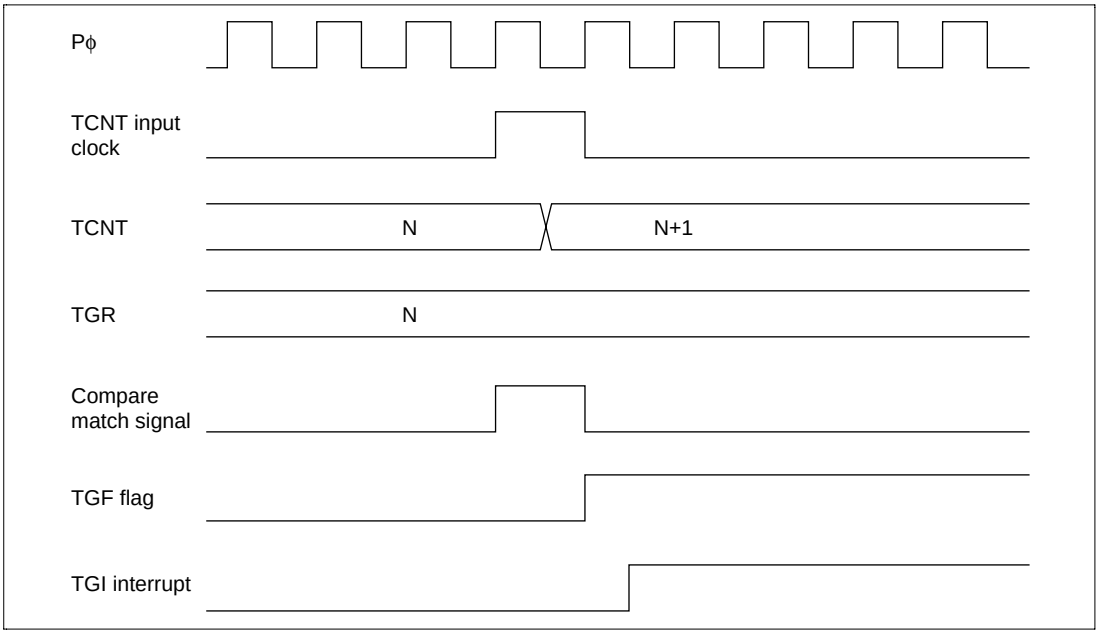



**Figure 18.38 TGI Interrupt Timing (Compare Match)**

**TGF Flag Setting Timing in Case of Input Capture:** Figure 18.39 shows the timing for setting of the TGF flag in TSR by input capture occurrence, and TGI interrupt request signal timing.

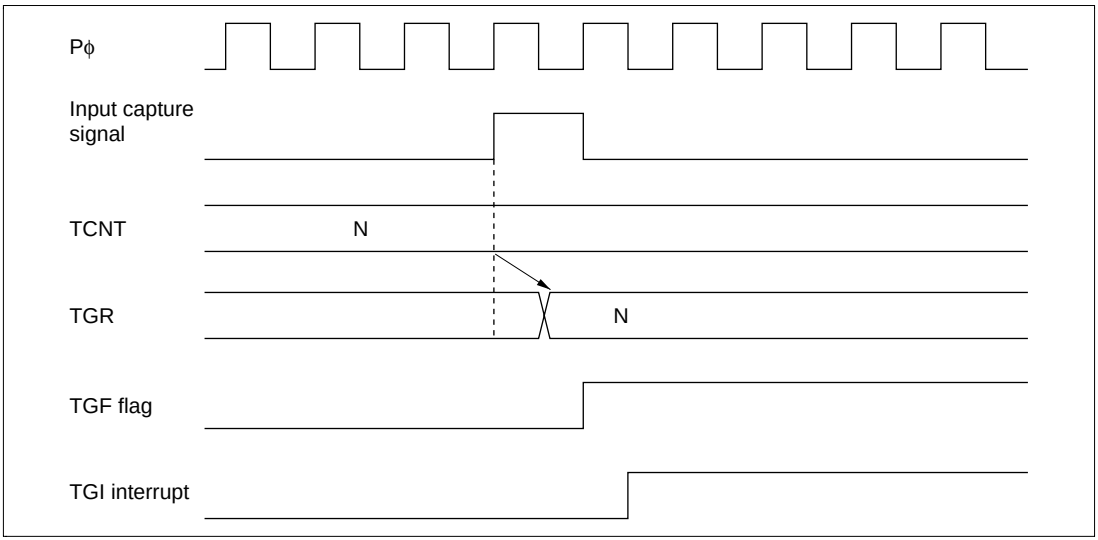



**Figure 18.39 TGI Interrupt Timing (Input Capture)**

**TCFV Flag/TCFU Flag Setting Timing:** Figure 18.40 shows the timing for setting of the TCFV flag in TSR by overflow occurrence, and TCIV interrupt request signal timing.

Figure 18.41 shows the timing for setting of the TCFU flag in TSR by underflow occurrence, and TCIU interrupt request signal timing.

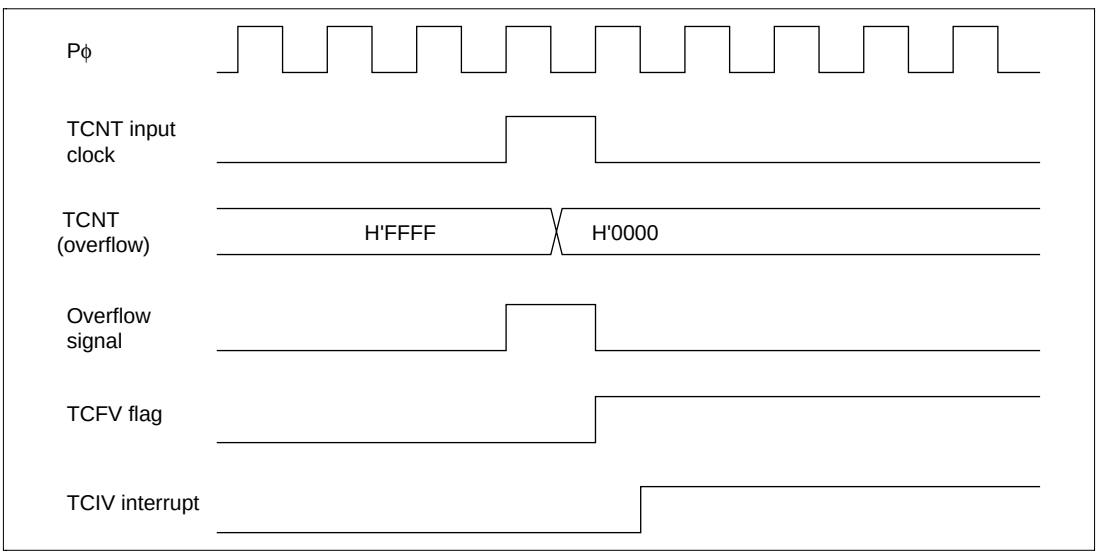



**Figure 18.40 TCIV Interrupt Setting Timing**

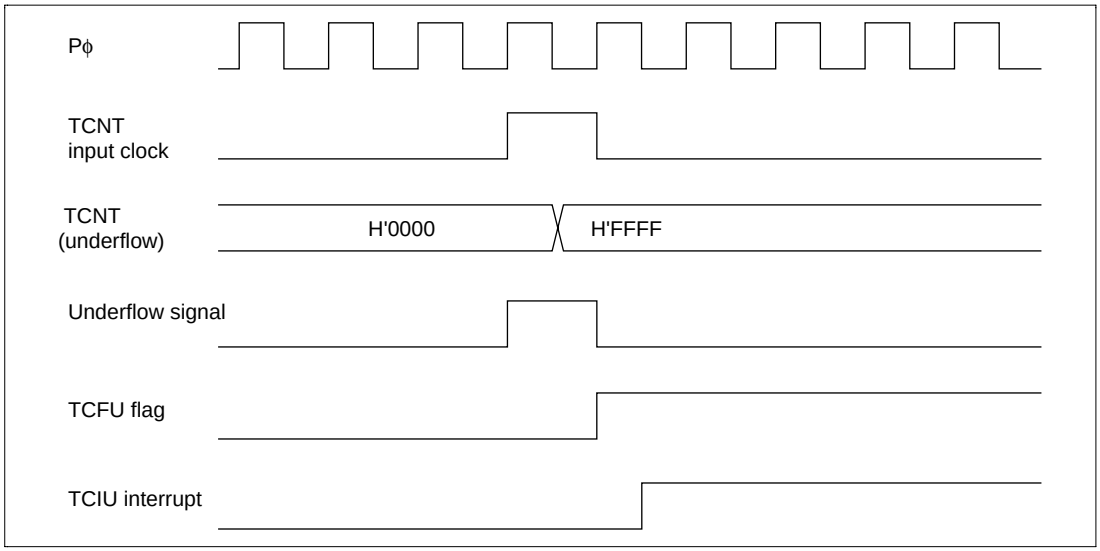



**Figure 18.41 TCIU Interrupt Setting Timing**

**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 DMAC is activated, the flag is cleared automatically. Figure 18.42 shows the timing for status flag clearing by the CPU, and figure 18.43 shows the timing for status flag clearing by the DMAC.

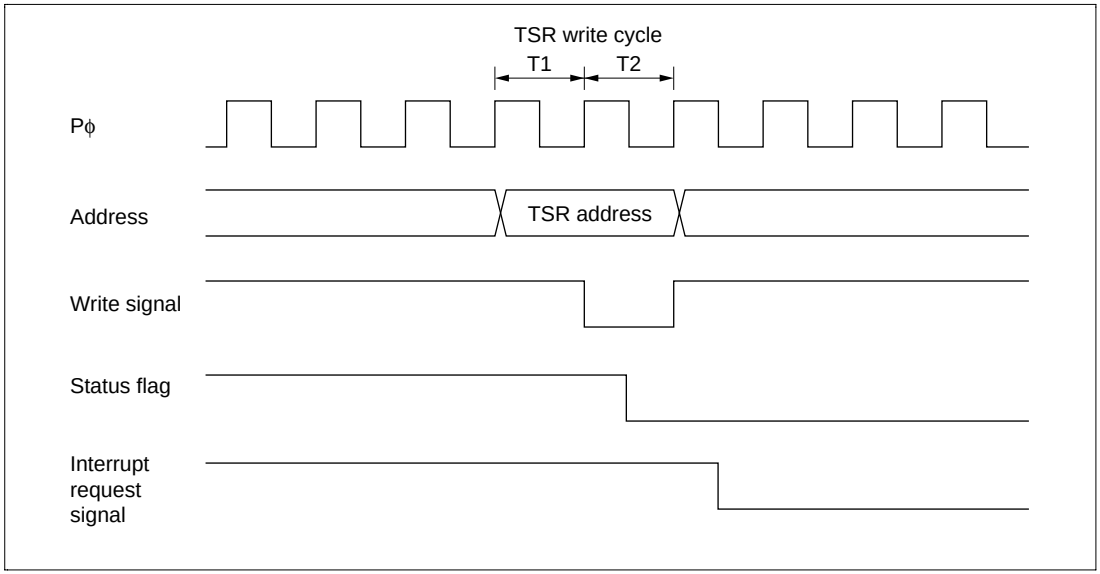



**Figure 18.42 Timing for Status Flag Clearing by CPU**

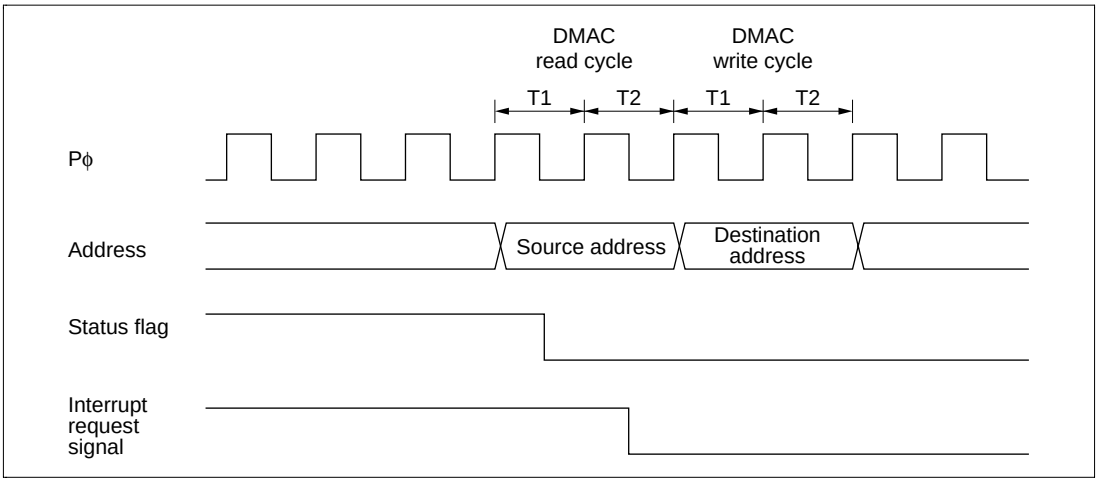



**Figure 18.43 Timing for Status Flag Clearing by DMAC Activation**

## 18.7 Usage Notes

Note that the kinds of operation and contention described below occur during TPU operation.

### 18.7.1 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 with a narrower pulse width.

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 18.44 shows the input clock conditions in phase counting mode.

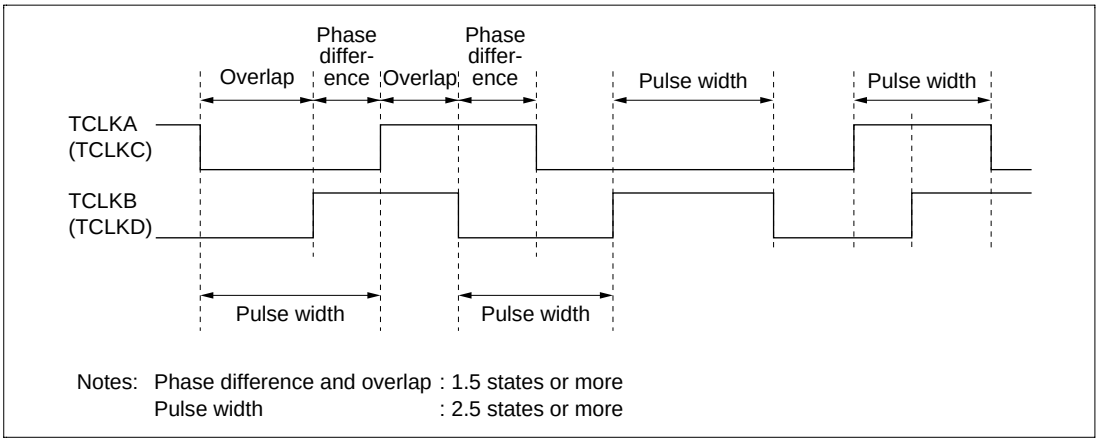



**Figure 18.44 Phase Difference, Overlap, and Pulse Width in Phase Counting Mode**

### 18.7.2 Caution on Period Setting

When counter clearing by 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:

$$f = \frac{P\phi}{(N + 1)}$$

Where f : Counter frequency
Pϕ : Peripheral module clock
N : TGR set value

### 18.7.3 Contention 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 18.45 shows the timing in this case.

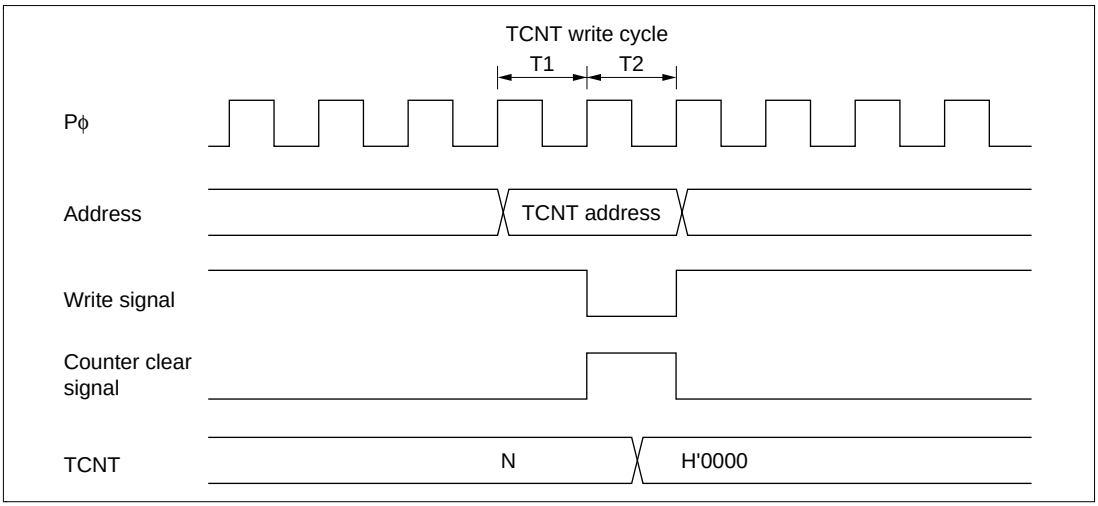



**Figure 18.45 Contention between TCNT Write and Clear Operations**

### 18.7.4 Contention 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 18.46 shows the timing in this case.

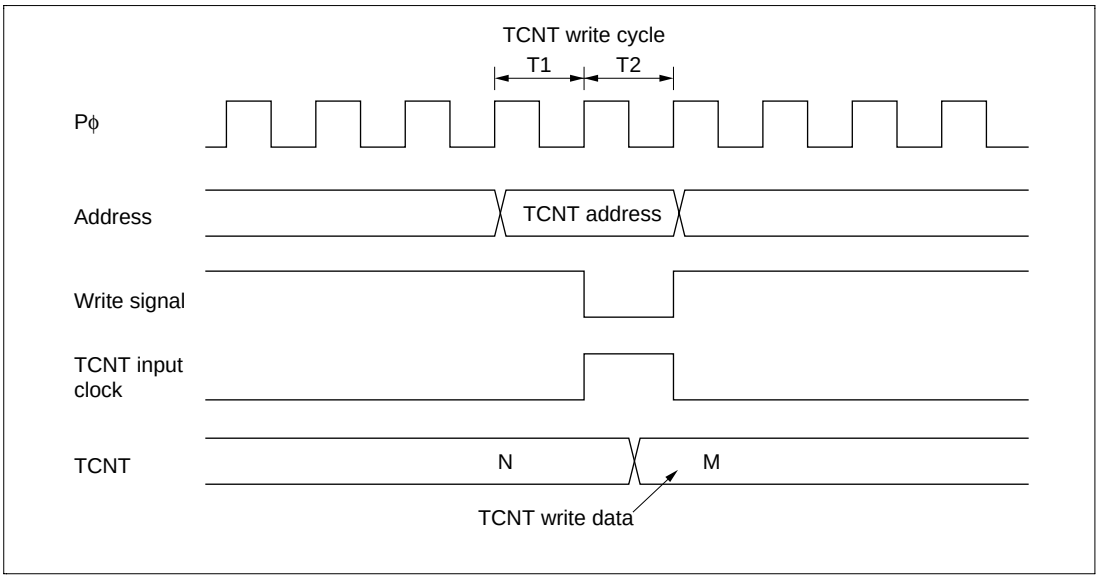



**Figure 18.46 Contention between TCNT Write and Increment Operations**

### 18.7.5 Contention 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 same value as before is written.

Figure 18.47 shows the timing in this case.

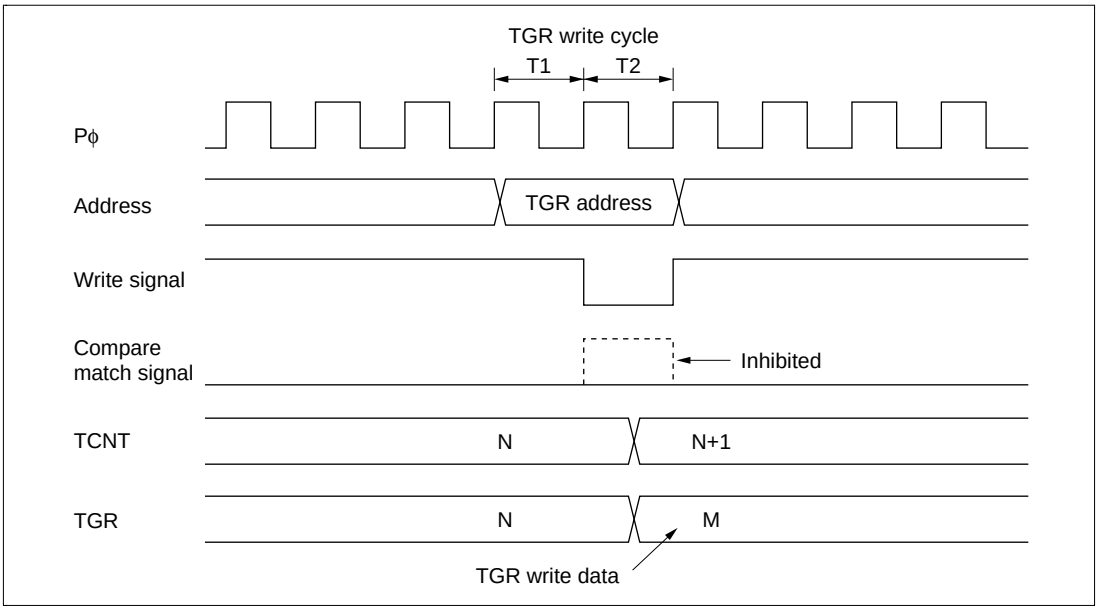



**Figure 18.47 Contention between TGR Write and Compare Match**

### 18.7.6 Contention between Buffer Register Write and Compare Match

If a compare match occurs in the T2 state of a TGR write cycle, the data transferred to TGR by the buffer operation will be the write data.

Figure 18.48 shows the timing in this case.

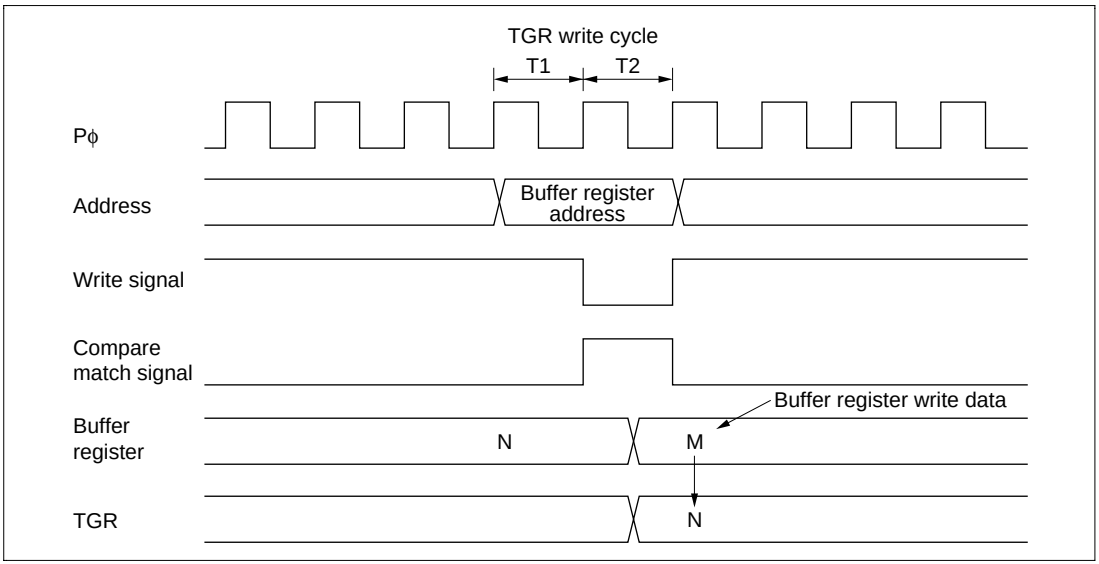



**Figure 18.48 Contention between Buffer Register Write and Compare Match**

### 18.7.7 Contention between TGR Read and Input Capture

If the input capture signal is generated in the T1 state of a TGR read cycle, the data that is read will be the data before input capture transfer.

Figure 18.49 shows the timing in this case.

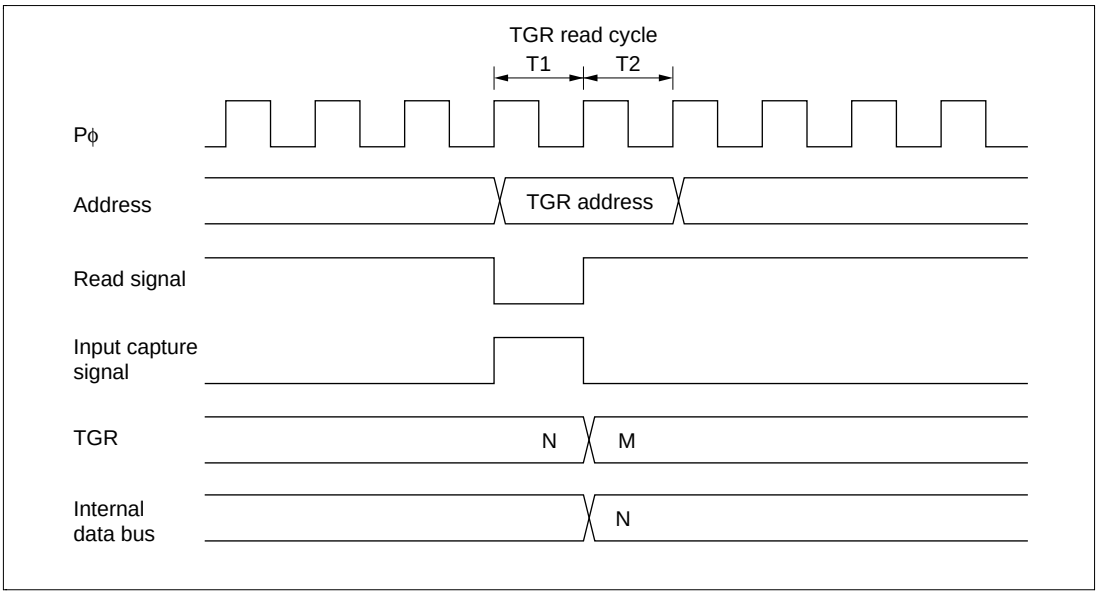



**Figure 18.49 Contention between TGR Read and Input Capture**

### 18.7.8 Contention between TGR Write and Input Capture

If the 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 18.50 shows the timing in this case.

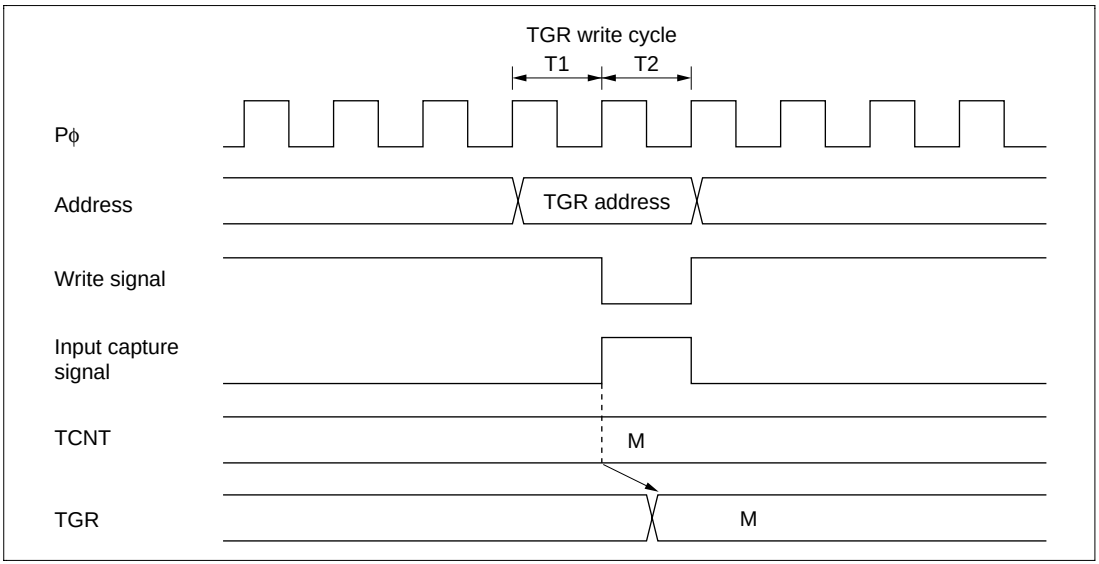



**Figure 18.50 Contention between TGR Write and Input Capture**

### 18.7.9 Contention between Buffer Register Write and Input Capture

If the input capture signal is generated in the T2 state of a buffer write cycle, the buffer operation takes precedence and the write to the buffer register is not performed.

Figure 18.51 shows the timing in this case.

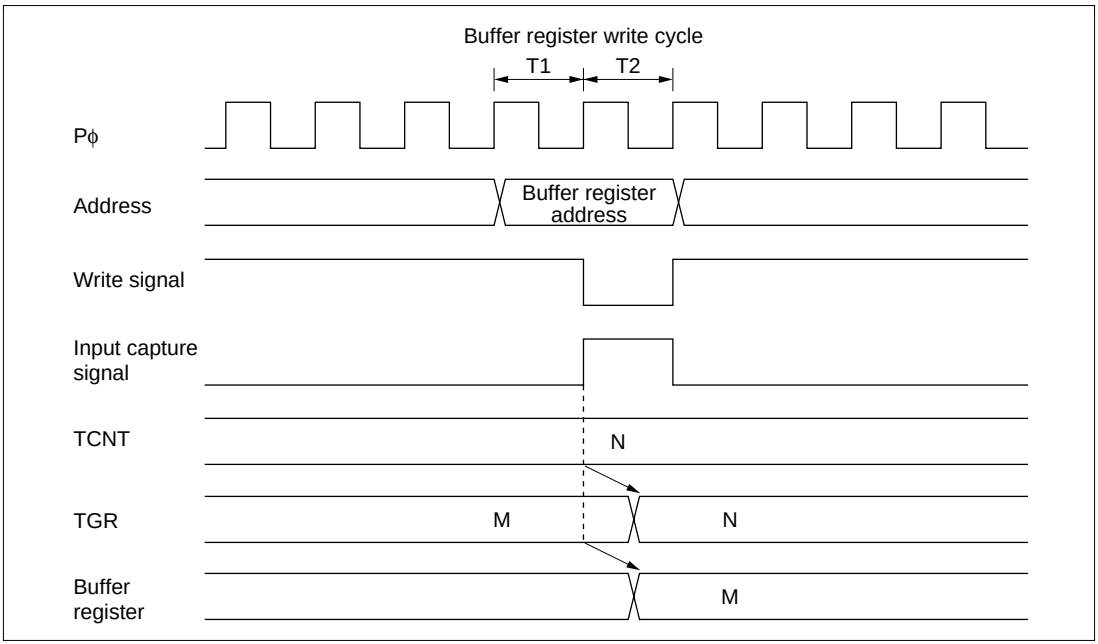



**Figure 18.51 Contention between Buffer Register Write and Input Capture**

### 18.7.10 Contention 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 18.52 shows the operation timing when a TGR compare match is specified as the clearing source, and H'FFFF is set in TGR.

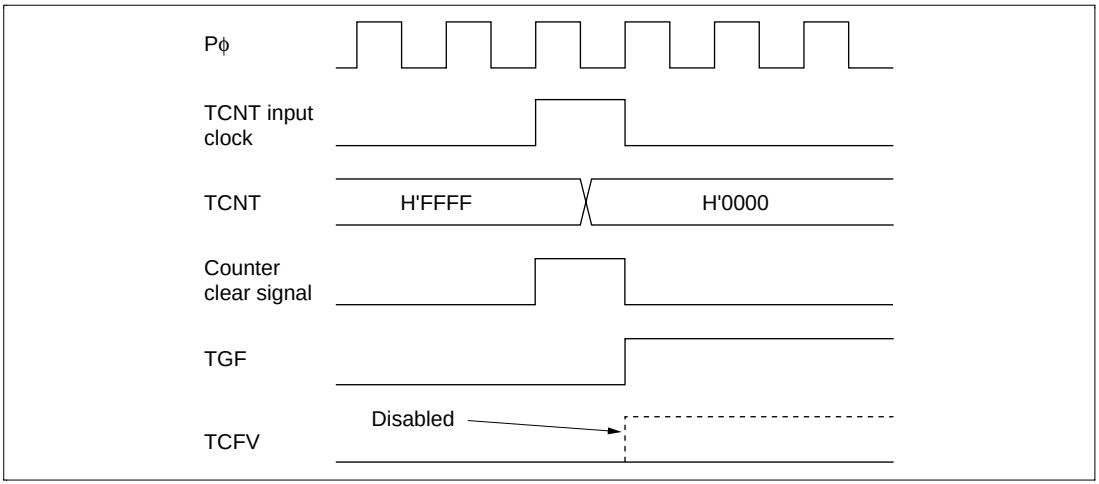



**Figure 18.52 Contention between Overflow and Counter Clearing**

### 18.7.11 Contention 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 18.53 shows the operation timing in the case of contention between a TCNT write and overflow.

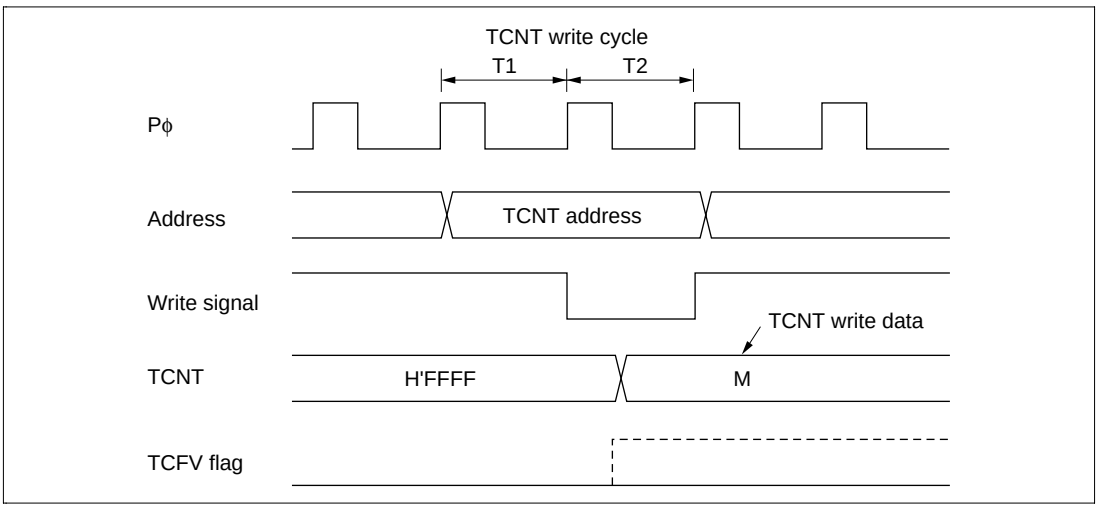



**Figure 18.53 Contention between TCNT Write and Overflow**

### 18.7.12 Multiplexing of I/O Pins

In the Chip, 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.

### 18.7.13 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 DMAC activation source. Interrupts should therefore be disabled before entering module stop mode.

# Section 19 Hitachi User Debug Interface (H-UDI)

## 19.1 Overview

The Hitachi user debug interface (H-UDI) provides data transfer and interrupt request functions. The H-UDI performs serial transfer by means of external signal control.

### 19.1.1 Features

The H-UDI has the following features conforming to the IEEE 1149.1 standard.

- Five test signals (TCK, TDI, TDO, TMS, and $\overline{\text{TRST}}$)
- TAP controller
- Instruction register
- Data register
- Bypass register

The H-UDI has two instructions.

- Bypass mode
  Test mode conforming to IEEE 1149.1
- H-UDI interrupt
  H-UDI interrupt request to INTC

This chip does not support test modes other than bypass mode.

### 19.1.2 H-UDI Block Diagram

Figure 19.1 shows a block diagram of the H-UDI.

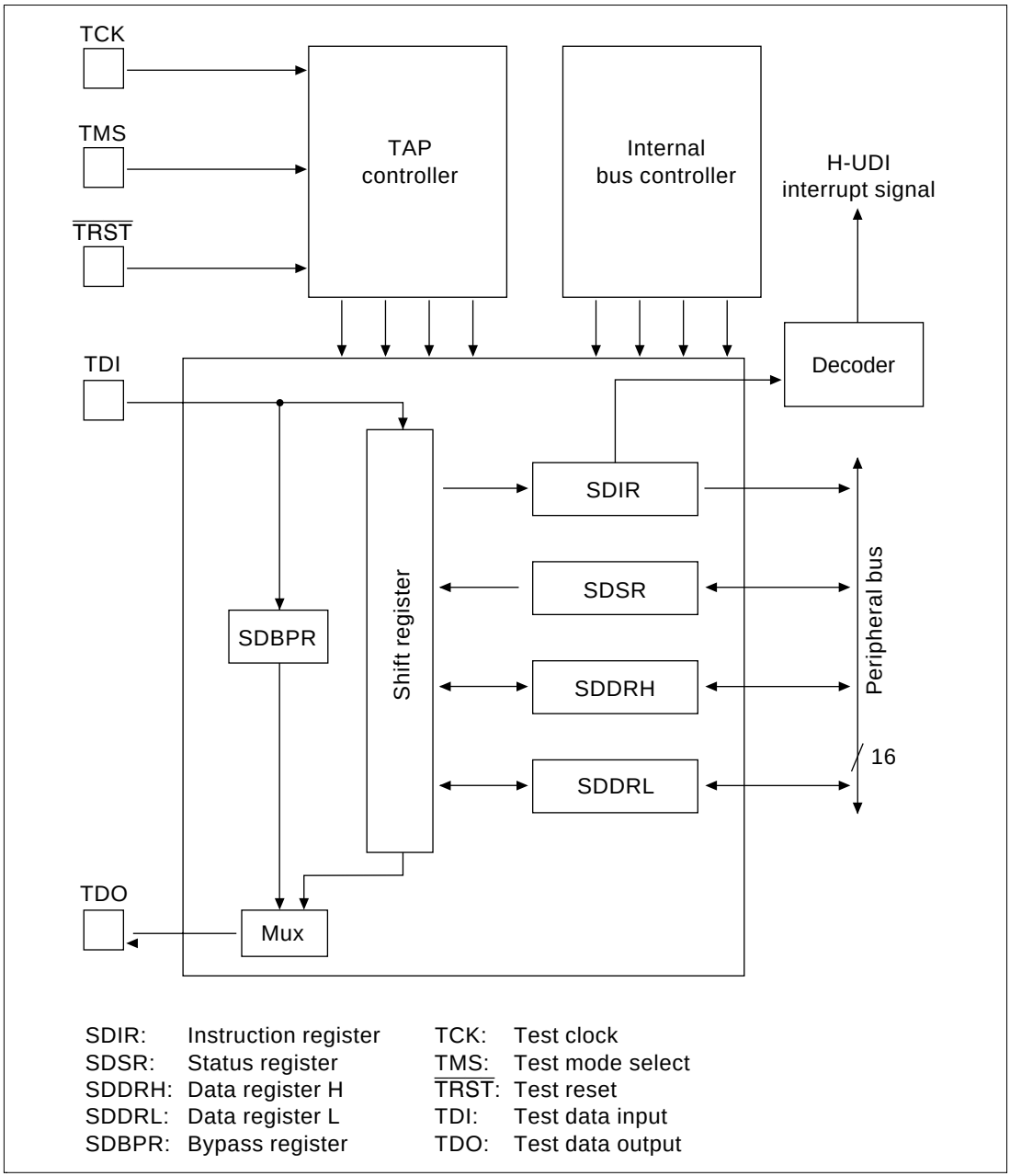



**Figure 19.1 H-UDI Block Diagram**

### 19.1.3 Pin Configuration

Table 19.1 shows the H-UDI pin configuration.

**Table 19.1 Pin Configuration**

| Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Test clock | TCK | Input | Test clock input |
| Test mode select | TMS | Input | Test mode select input signal |
| Test data input | TDI | Input | Serial data input |
| Test data output | TDO | Output | Serial data output |
| Test reset | $\overline{\text{TRST}}$ | Input | Test reset input signal |

### 19.1.4 Register Configuration

Table 19.2 shows the H-UDI registers.

**Table 19.2 Register Configuration**

| Register | Abbreviation | R/W*[1] | Initial Value*[2] | Address | Access Size (Bits) |
|---|---|---|---|---|---|
| Instruction register | SDIR | R | H'F000 | H'FFFFFCE0 | 8/16/32 |
| Status register | SDSR | R/W | H'0101 | H'FFFFFCE2 | 8/16/32 |
| Data register H | SDDRH | R/W | Undefined | H'FFFFFCE4 | 8/16/32 |
| Data register L | SDDRL | R/W | Undefined | H'FFFFFCE6 | 8/16/32 |
| Bypass register | SDBPR | — | — | — | — |

Notes: 1. Indicates whether the register can be read/written to by the CPU.
2. Initial value when the $\overline{\text{TRST}}$ signal is input. Registers are not initialized by a reset (power-on or manual) or in standby mode.

Instructions and data can be input to the instruction register (SDIR) and data register (SDDR) by serial transfer from the test data input pin (TDI). Data from the status register (SDSR) and SDDR can be output via the test data output pin (TDO). The bypass register (SDBPR) is a 1-bit register to which TDI and TDO are connected in bypass mode. All registers except SDBPR can be accessed by the CPU.

Table 19.3 shows the kinds of serial transfer possible with each register.

**Table 19.3 H-UDI Register Serial Transfer**

| Register | Serial Input | Serial Output |
|---|---|---|
| SDIR | Possible | Possible |
| SDSR | Not possible | Possible |
| SDDRH | Possible | Possible |
| SDDRL | Possible | Possible |
| SDBPR | Possible | Possible |

## 19.2 External Signals

### 19.2.1 Test Clock (TCK)

The test clock pin (TCK) provides an independent clock supply to the H-UDI. As the clock input to TCK is supplied directly to the H-UDI, a clock waveform with a duty cycle close to 50% should be input (for details, see section 23, Electrical Characteristics). If no clock is input, TCK is fixed at 1 by internal pull-up.

### 19.2.2 Test Mode Select (TMS)

The test mode select pin (TMS) is sampled on the rise of TCK. TMS controls the internal state of the TAP controller. If no signal is input, TMS is fixed at 1 by internal pull-up.

### 19.2.3 Test Data Input (TDI)

The test data input pin (TDI) performs serial input of instructions and data for H-UDI registers. TDI is sampled on the rise of TCK. If no signal is input, TDI is fixed at 1 by internal pull-up.

### 19.2.4 Test Data Output (TDO)

The test data output pin (TDO) performs serial output of instructions and data from H-UDI registers. Transfer is performed in synchronization with TCK. If there is no output, TDO goes to the high-impedance state.

### 19.2.5 Test Reset ($\overline{\text{TRST}}$)

The test reset pin ($\overline{\text{TRST}}$) initializes the H-UDI asynchronously. If no signal is input, $\overline{\text{TRST}}$ is fixed at 1 by internal pull-up.

## 19.3 Register Descriptions

### 19.3.1 Instruction Register (SDIR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TS3 | TS2 | TS1 | TS0 | — | — | — | — |
| Initial value: | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R |

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

The instruction register (SDIR) is a 16-bit register that can only be read by the CPU. H-UDI instructions can be transferred to SDIR by serial input from TDI. SDIR can be initialized by the $\overline{\text{TRST}}$ signal, but is not initialized by a reset or in standby mode.

Instructions transferred to SDIR must be 4 bits in length. If an instruction exceeding 4 bits is input, the last 4 bits of the serial data will be stored in SDIR.

Bits 15 to 12—Test Set Bits (TS3–TS0): Table 19.4 shows the instruction configuration.

**Table 19.4 Instruction Configuration**

| Bit 15: TS3 | Bit 14: TS2 | Bit 13: TS1 | Bit 12: TS0 | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Reserved |
| | | | 1 | Reserved |
| | | 1 | 0 | Reserved |
| | | | 1 | Reserved |
| | 1 | 0 | 0 | Reserved |
| | | | 1 | Reserved |
| | | 1 | 0 | Reserved |
| | | | 1 | Reserved |
| 1 | 0 | 0 | 0 | Reserved |
| | | | 1 | Reserved |
| | | 1 | 0 | H-UDI interrupt |
| | | | 1 | Reserved |
| | 1 | 0 | 0 | Reserved |
| | | | 1 | Reserved |
| | | 1 | 0 | Reserved |
| | | | 1 | Bypass mode (Initial value) |

Bits 11 to 0—Reserved: These bits always read 0. The write value should always be 0.

### 19.3.2 Status Register (SDSR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SDTRF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W |

The status register (SDSR) is a 16-bit register that can be read and written to by the CPU. Output from TDO is possible for SDSR, but serial data cannot be written to SDSR via TDI. The SDTRF bit is output by means of a 1-bit shift. In the case of a 2-bit shift, the SDTRF bit is first output, followed by a reserved bit.

SDSR is initialized by $\overline{\text{TRST}}$ signal input, but is not initialized by a reset or in standby mode.

Bits 15 to 1—Reserved: Bits 15 to 9 and 7 to 1 always read 0, and the write value should always be 0. Bit 8 always reads 1, and the write value should always be 1.

Bit 0—Serial Data Transfer Control Flag (SDTRF): Indicates whether H-UDI registers can be accessed by the CPU. The SDTRF bit is reset by the $\overline{\text{TRST}}$ signal , but is not initialized by a reset or in standby mode.

| Bit 0: SDTRF | Description |
|---|---|
| 0 | Serial transfer to SDDR has ended, and SDDR can be accessed |
| 1 | Serial transfer to SDDR in progress (Initial value) |

### 19.3.3 Data Register (SDDR)

The data register (SDDR) comprises data register H (SDDRH) and data register L (SDDRL), each of which has the following configuration.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

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

SDDRH and SDDRL are 16-bit registers that can be read and written to by the CPU. SDDR is connected to TDO and TDI for serial data transfer to and from an external device.

32-bit data is input and output in serial data transfer. If data exceeding 32 bits is input, only the last 32 bits will be stored in SDDR. Serial data is input starting from the MSB of SDDR (bit 15 of SDDRH), and output starting from the LSB (bit 0 of SDDRL).

This register is not initialized by a reset, in standby mode, or by the $\overline{\text{TRST}}$ signal.

### 19.3.4 Bypass Register (SDBPR)

The bypass register (SDBPR) is a one-bit shift register. In bypass mode, SDBPR is connected to TDI and TDO, and the chip is excluded from the board test when a boundary scan test is conducted. SDBPR cannot be read or written to by the CPU.

## 19.4 Operation

### 19.4.1 H-UDI Interrupt and Serial Transfer

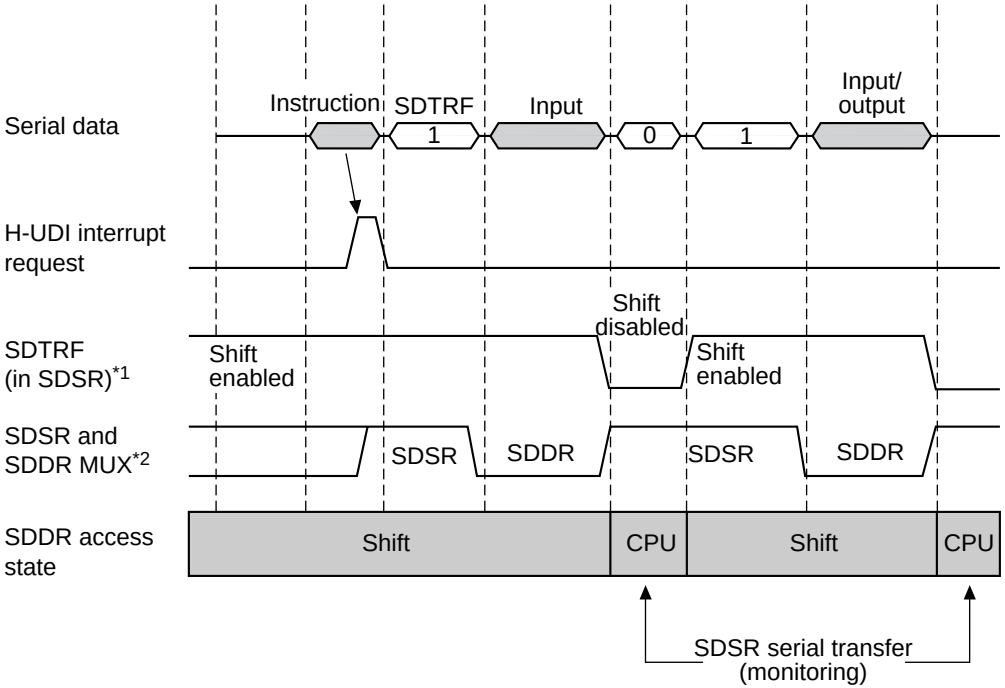
When an H-UDI interrupt instruction is transferred to SDIR via TDI, an interrupt is generated. Data transfer can be controlled by means of the H-UDI interrupt service routine. Transfer can be performed by means of SDDR.

Control of data input/output between an external device and the H-UDI is performed by monitoring the SDTRF bit in SDSR externally and internally. Internal SDTRF bit monitoring is carried out by having SDSR read by the CPU.

The H-UDI interrupt and serial transfer procedure is as follows.

1. An instruction is input to SDIR by serial transfer, and an H-UDI interrupt request is generated.
2. After the H-UDI interrupt request is issued, the SDTRF bit in SDSR is monitored externally. After output of SDTRF = 1 from TDO is observed, serial data is transferred to SDDR.
3. On completion of the serial transfer to SDDR, the SDTRF bit is cleared to 0, and SDDR can be accessed by the CPU. After SDDR has been accessed, SDDR serial transfer is enabled by setting the SDTRF bit to 1 in SDSR.
4. Serial data transfer between an external device and the H-UDI can be carried out by constantly monitoring the SDTRF bit in SDSR externally and internally.

Figures 19.2, 19.3, and 19.4 show the timing of data transfer between an external device and the H-UDI.




Notes: 1. SDTRF flag (in SDSR): Indicates whether SDDR access by the CPU or serial transfer data input/output to SDDR is possible.

| | |
|---|---|
| 1 | SDDR is shift-enabled. Do not access SDDR until SDTRF = 0. |
| 2 | SDDR is shift-disabled. SDDR access by the CPU is enabled. |

Conditions:
- SDTRF = 1
  — When $\overline{\text{TRST}}$ = 0
  — When the CPU writes 1
  — In bypass mode
- SDTRF = 0
  — End of SDDR shift access in serial transfer

2. SDSR/SDDR (Update-DR state) internal MUX switchover timing
   - Switchover from SDSR to SDDR: On completion of serial transfer in which SDTRF = 1 is output from TDO
   - Switchover from SDDR to SDSR: On completion of serial transfer to SDDR

**Figure 19.2 Data Input/Output Timing Chart (1)**

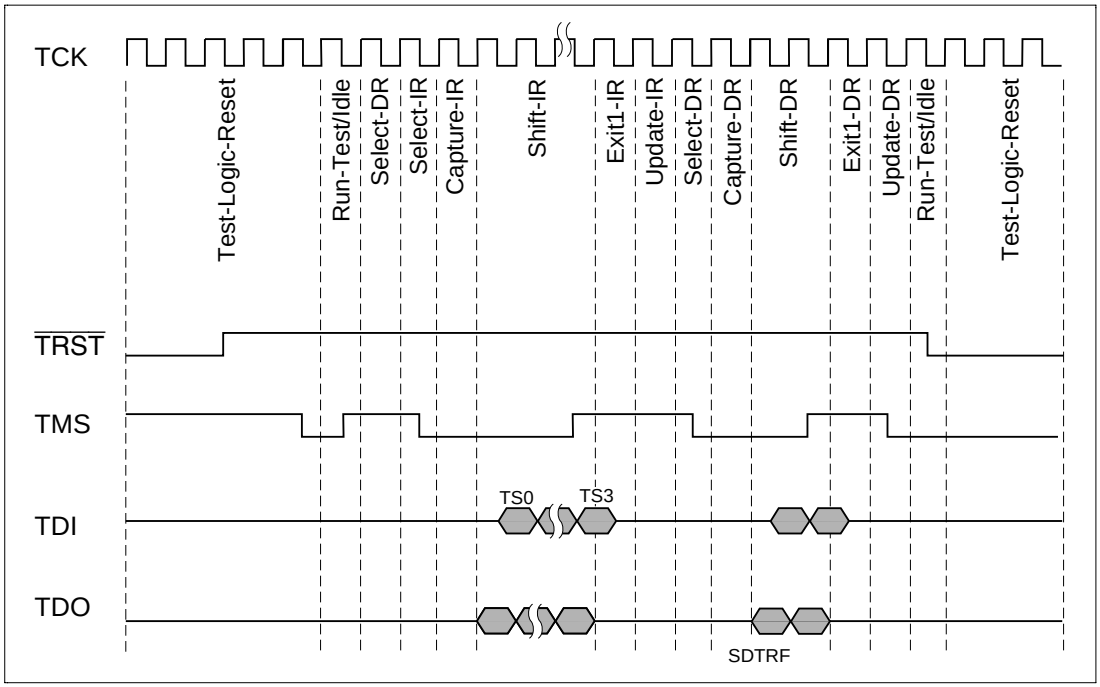



**Figure 19.3 Data Input/Output Timing Chart (2)**

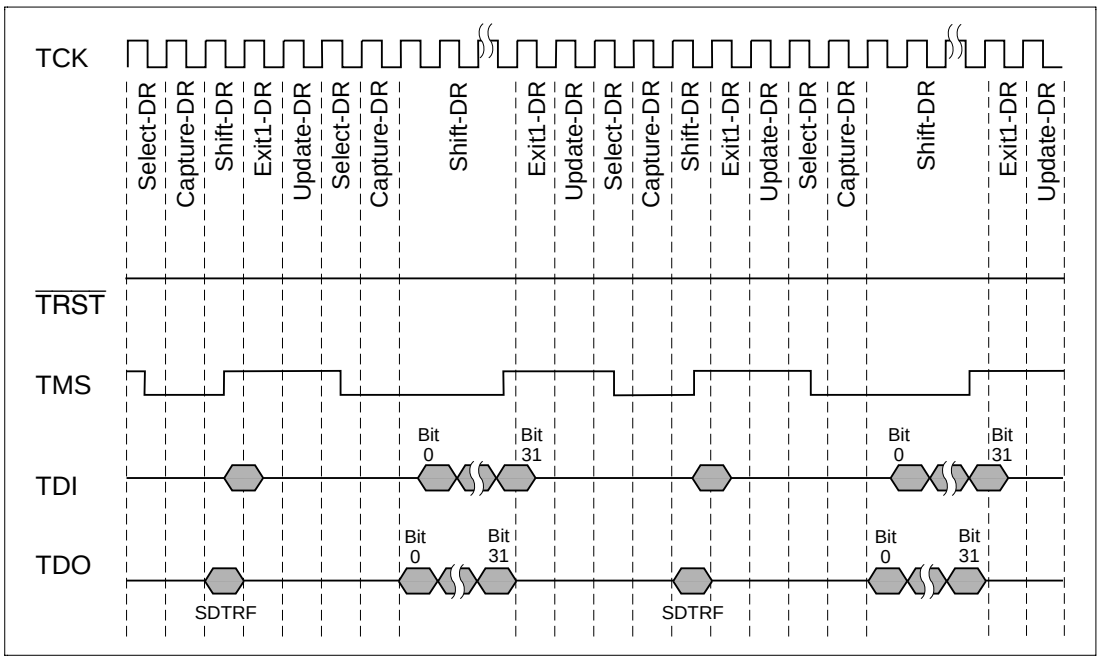



**Figure 19.4 Data Input/Output Timing Chart (3)**

### 19.4.2 Bypass Mode

Bypass mode can be used to exclude the chip from the test items in a boundary-scan test. Bypass mode is entered by transferring B'1111 to SDIR. In bypass mode, SDBPR is connected to TDI and TDO.

### 19.4.3 H-UDI Reset

The H-UDI can be reset in two ways.

- The H-UDI is reset when the $\overline{\text{TRST}}$ signal is held at 0.
- When $\overline{\text{TRST}}$ = 1, the H-UDI can be reset by inputting at least five TCK clock cycles while TMS = 1.

## 19.5 Usage Notes

- A reset must always be executed by driving the $\overline{\text{TRST}}$ signal to 0, regardless of whether or not the H-UDI is to be activated. $\overline{\text{TRST}}$ must be held low for 20 TCK clock cycles. For details, see section 23, Electrical Characteristics.
- The registers are not initialized in standby mode. If $\overline{\text{TRST}}$ is set to 0 in standby mode, bypass mode will be entered.
- The frequency of TCK must be lower than that of the peripheral module clock (Pϕ). For details, see section 23, Electrical Characteristics.
- In data transfer, data input/output starts with the LSB. Figure 19.5 shows serial data input/output.
- If more than 32 bits are serially transferred, serial data exceeding 32 bits output from TDO should be ignored.
- If the H-UDI serial transfer sequence is disrupted, a $\overline{\text{TRST}}$ reset must be executed. Transfer should then be retried, regardless of the transfer operation.
- The TDO output timing is from the rise of TCK.
- In the Shift-IR state, the lower 2 bits of the output data from TDO (the IR status word) may not be 01.
- When debugging is carried out with an E10 emulator using a debug chip, the H-UDI is used. Therefore, the five H-UDI-related pins should not be used as general I/O ports.
- When the emulator is used, the UBC and H-UDI are used by the emulator, and therefore operation relating to the UBC and H-UDI may differ between the emulator and the actual chip. If the UBC and H-UDI are not used by the user system, register settings should not be made.

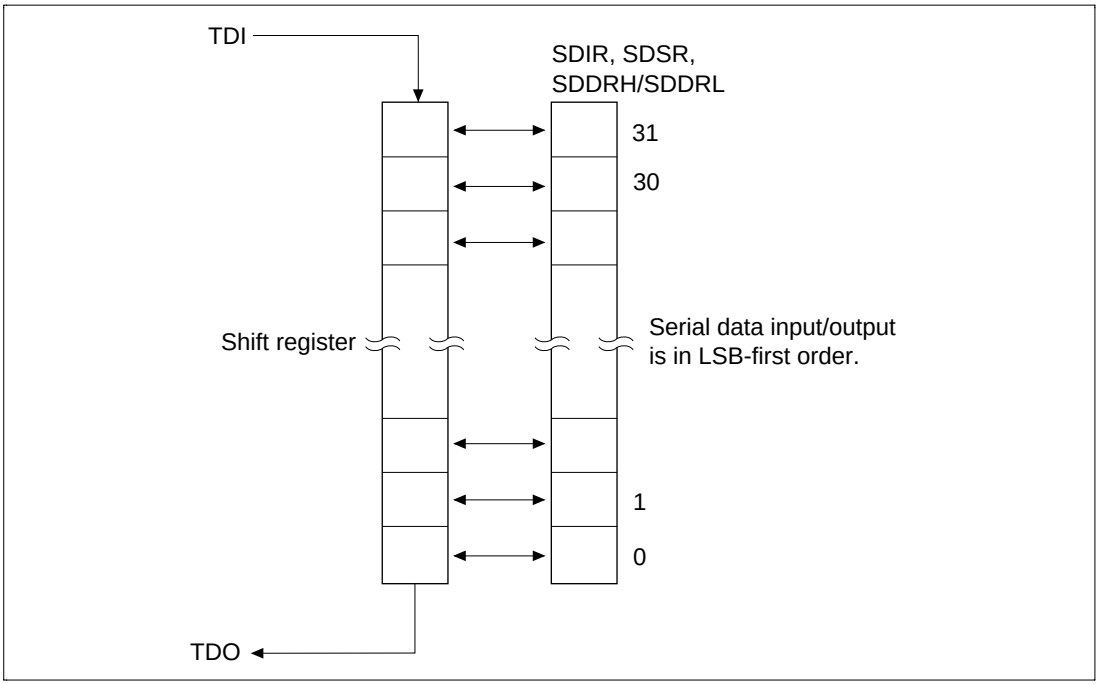



**Figure 19.5 Serial Data Input/Output**

# Section 20 Pin Function Controller (PFC)

## 20.1 Overview

The pin function controller (PFC) consists of registers to select multiplexed pin functions and input/output direction. The pin function and input/output direction can be selected for individual pins regardless of the operating mode of the chip. Table 20.1 shows the chip's multiplex pins.

**Table 20.1 Multiplex Pins**

| | Function 1 | | | Function 2 | | | Function 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Port | Signal Name | I/O | Related Module | Signal Name | I/O | Related Module | Signal Name | I/O | Related Module |
| A | TCK | I | (H-UDI) | PA13 | I/O | (Port) | — | — | — |
| A | TMS | I | (H-UDI) | PA12 | I/O | (Port) | — | — | — |
| A | TDI | I | (H-UDI) | PA11 | I/O | (Port) | — | — | — |
| A | TDO | O | (H-UDI) | PA10 | I/O | (Port) | — | — | — |
| A | PA9 | I/O | (Port) | STS2 | I/O | (SIO2) | — | — | — |
| A | PA8 | I/O | (Port) | STXD2 | O | (SIO2) | — | — | — |
| A | $\overline{\text{WDTOVF}}$ | O | (WDT) | PA7 | I/O | (Port) | — | — | — |
| A | FTCI | I | (FRT) | PA6 | I/O | (Port) | — | — | — |
| A | FTI | I | (FRT) | PA5 | I/O | (Port) | — | — | — |
| A | FTOA | O | (FRT) | PA4 | I/O | (Port) | — | — | — |
| A | FTOB | O | (FRT) | CKPO | O | (Port) | — | — | — |
| A | SCK | I/O | (SCI)*[1] | PA2 | I/O | (Clock) | — | — | — |
| A | RXD | I | (SCI) | PA1 | I/O | (Port) | — | — | — |
| A | TXD | O | (SCI) | PA0 | I/O | (Port) | — | — | — |
| B | PB15 | I/O | (Port) | SRCK0 | I | (SIO) | SCK2 | I/O | (SCIF) |
| B | PB14 | I/O | (Port) | SRS0 | I | (SIO) | RXD2 | I | (SCIF) |
| B | PB13 | I/O | (Port) | SRXD0 | I | (SIO) | TXD2 | O | (SCIF) |
| B | PB12 | I/O | (Port) | STCK0 | I | (SIO) | SCK1 | I/O | (SCIF) |
| B | PB11 | I/O | (Port) | STS0 | I/O | (SIO) | RXD1 | I | (SCIF) |
| B | PB10 | I/O | (Port) | STXD0 | O | (SIO) | TXD1 | O | (SCIF) |
| B | PB9 | I/O | (Port) | SRCK1 | I | (SIO) | $\overline{\text{RTS}}$ | O | (SCIF) |
| B | PB8 | I/O | (Port) | SRS1 | I | (SIO) | $\overline{\text{CTS}}$ | I | (SCIF) |
| B | PB7 | I/O | (Port) | SRXD1 | I | (SIO) | TIOCA2 | I/O | (TPU) |
| B | PB6 | I/O | (Port) | STCK1 | I | (SIO) | TIOCB2 | I/O | (TPU) |
| B | PB5 | I/O | (Port) | STS1 | I/O | (SIO) | TIOCA1 | I/O | (TPU) |
| B | PB4 | I/O | (Port) | STXD1 | O | (SIO) | TIOCB1 | I/O | (TPU) |
| B | PB3 | I/O | (Port) | SRCK2 | I | (SIO) | TIOCA0 | I/O | (TPU) |
| B | PB2 | I/O | (Port) | SRS2 | I | (SIO) | TIOCB0 | I/O | (TPU) |
| B | PB1 | I/O | (Port) | SRXD2 | I | (SIO) | TIOCC0 | I/O | (TPU) |
| B | PB0 | I/O | (Port) | STCK2 | I | (SIO) | TIOCD0 | I/O | (TPU) |

Notes: In the initial state, function 1 is selected.

1. The initial state is input.

## 20.2 Register Configuration

Table 20.2 shows the PFC registers.

**Table 20.2 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Port A control register | PACR | R/W | H'0000 | H'FFFFFC80 | 8, 16 |
| Port A I/O register | PAIOR | R/W | H'0000 | H'FFFFFC82 | 8, 16 |
| Port B control register | PBCR | R/W | H'0000 | H'FFFFFC88 | 8, 16 |
| Port B I/O register | PBIOR | R/W | H'0000 | H'FFFFFC8A | 8, 16 |
| Port B control register 2 | PBCR2 | R/W | H'0000 | H'FFFFFC8E | 8, 16 |

## 20.3 Register Descriptions

### 20.3.1 Port A Control Register (PACR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | PA13MD | PA12MD | PA11MD | PA10MD | PA9MD | PA8MD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PA7MD | PA6MD | PA5MD | PA4MD | PA3MD | PA2MD | PA1MD | PA0MD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The port A control register (PACR) is a 16-bit read/write register that selects the functions of the 14 multiplex pins in port A.

PACR is initialized to H'0000 by a power-on reset. It is not initialized by a manual reset or in standby mode or sleep mode.

Bits 15 and 14—Reserved. These bits always read 0. The write value should always be 0.

Bit 13—PA13 Mode Bit (PA13MD): Selects the function of pin TCK/PA13.

| Bit 13: PA13MD | Description | |
|---|---|---|
| 0 | H-UDI clock input (TCK) | (Initial value) |
| 1 | General input/output (PA13) | |

Bit 12—PA12 Mode Bit (PA12MD): Selects the function of pin TMS/PA12.

| Bit 12: PA12MD | Description | |
|---|---|---|
| 0 | H-UDI mode signal input (TMS) | (Initial value) |
| 1 | General input/output (PA12) | |

Bit 11—PA11 Mode Bit (PA11MD): Selects the function of pin TDI/PA11.

| Bit 11: PA11MD | Description | |
|---|---|---|
| 0 | H-UDI data input (TDI) | (Initial value) |
| 1 | General input/output (PA11) | |

Bit 10—PA10 Mode Bit (PA10MD): Selects the function of pin TDO/PA10.

| Bit 10: PA10MD | Description | |
|---|---|---|
| 0 | H-UDI data output (TDO) | (Initial value) |
| 1 | General input/output (PA10) | |

Bit 9—PA9 Mode Bit (PA9MD): Selects the function of pin PA9/STS2.

| Bit 9: PA9MD | Description | |
|---|---|---|
| 0 | General input/output (PA9) | (Initial value) |
| 1 | SIO2 transmit start signal input/output (STS2) | |

Bit 8—PA8 Mode Bit (PA8MD): Selects the function of pin PA8/STXD2.

| Bit 8: PA8MD | Description | |
|---|---|---|
| 0 | General input/output (PA8) | (Initial value) |
| 1 | SIO2 transmit data output (STXD2) | |

Bit 7—PA7 Mode Bit (PA7MD): Selects the function of pin $\overline{\text{WDTOVF}}$/PA7.

| Bit 7: PA7MD | Description | |
|---|---|---|
| 0 | WDT overflow signal output ($\overline{\text{WDTOVF}}$) | (Initial value) |
| 1 | General input/output (PA7) | |

Bit 6—PA6 Mode Bit (PA6MD): Selects the function of pin FTCI/PA6.

| Bit 6: PA6MD | Description | |
|---|---|---|
| 0 | FRT clock input (FTCI) | (Initial value) |
| 1 | General input/output (PA6) | |

Bit 5—PA5 Mode Bit (PA5MD): Selects the function of pin FTI/PA5.

| Bit 5: PA5MD | Description | |
|---|---|---|
| 0 | FRT input capture input (FTI) | (Initial value) |
| 1 | General input/output (PA5) | |

Bit 4—PA4 Mode Bit (PA4MD): Selects the function of pin FTO4/PA4.

| Bit 4: PA4MD | Description | |
|---|---|---|
| 0 | FRT output compare output (FTOA) | (Initial value) |
| 1 | General input/output (PA4) | |

Bit 3—PA3 Mode Bit (PA3MD): Selects the function of pin FTOB/CKPO.

| Bit 3: PA3MD | Description | |
|---|---|---|
| 0 | FRT output compare output (FTOB) | (Initial value) |
| 1 | Peripheral module clock output (CKPO) | |

Bit 2—PA2 Mode Bit (PA2MD): Selects the function of pin SCK/PA2.

| Bit 2: PA2MD | Description | |
|---|---|---|
| 0 | SCI transmit/receive clock input/output (SCK) | (Initial value) |
| 1 | General input/output (PA2) | |

Bit 1—PA1 Mode Bit (PA1MD): Selects the function of pin RXD/PA1.

| Bit 1: PA1MD | Description | |
|---|---|---|
| 0 | SCI receive data input (RXD) | (Initial value) |
| 1 | General input/output (PA1) | |

Bit 0—PA0 Mode Bit (PA0MD): Selects the function of pin TXD/PA0.

| Bit 0: PA0MD | Description | |
|---|---|---|
| 0 | SCI transmit data output (TXD) | (Initial value) |
| 1 | General input/output (PA0) | |

### 20.3.2 Port A I/O Register (PAIOR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | PA13IOR | PA12IOR | PA11IOR | PA10IOR | PA9IOR | PA8IOR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PA7IOR | PA6IOR | PA5IOR | PA4IOR | — | PA2IOR | PA1IOR | PA0IOR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

The port A I/O register (PAIOR) is a 16-bit read/write register that selects the input/output direction of the 13 multiplex pins in port A. Bits PA13IOR to PA4IOR and PA2IOR to PA0IOR correspond to individual pins in port A. PAIOR is enabled when port A pins function as general input pins (PA13 to PA4 and PA2 to PA0), and disabled otherwise. When port A pins function as PA13 to PA0, a pin becomes an output when the corresponding bit in PAIOR is set to 1, and an input when the bit is cleared to 0.

PAIOR is initialized to H'0000 by a power-on reset. It is not initialized by a manual reset or in standby mode or sleep mode.

### 20.3.3 Port B Control Registers (PBCR, PBCR2)

The port B control registers (PBCR and PBCR2) are 16-bit read/write registers that select the functions of the 16 multiplex pins in port B. PBCR selects the functions of the pins for the upper 8 bits in port B, and PBCR2 selects the functions of the pins for the lower 8 bits in port B.

PBCR and PBCR2 are initialized to H'0000 by a power-on reset. They are not initialized by a manual reset or in standby mode or sleep mode.

**Port B Control Register (PBCR)**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB15 MD1 | PB15 MD0 | PB14 MD1 | PB14 MD0 | PB13 MD1 | PB13 MD0 | PB12 MD1 | PB12 MD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB11 MD1 | PB11 MD0 | PB10 MD1 | PB10 MD0 | PB9 MD1 | PB9 MD0 | PB8 MD1 | PB8 MD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 14—PB15 Mode Bits 1 and 0 (PB15MD1, PB15MD0): These bits select the function of pin PB15/SRCK0/SCK2.

| Bit 15: PB15MD1 | Bit 14: PB15MD0 | Description |
|---|---|---|
| 0 | 0 | General input/output (PB15) (Initial value) |
| | 1 | Reserved |
| 1 | 0 | SIO0 receive clock input (SRCK0) |
| | 1 | SCIF2 transmit/receive clock input/output (SCK2) |

Bits 13 and 12—PB14 Mode Bits 1 and 0 (PB14MD1, PB14MD0): These bits select the function of pin PB14/SRS0/RXD2.

| Bit 13: PB14MD1 | Bit 12: PB14MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB14) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO0 receive start signal input (SRS0) | |
| | 1 | SCIF2 receive data input (RXD2) | |

Bits 11 and 10—PB13 Mode Bits 1 and 0 (PB13MD1, PB13MD0): These bits select the function of pin PB13/SRXD0/TXD2.

| Bit 11: PB13MD1 | Bit 10: PB13MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB13) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO0 receive data input (SRXD0) | |
| | 1 | SCIF2 transmit data output (TXD2) | |

Bits 9 and 8—PB12 Mode Bits 1 and 0 (PB12MD1, PB12MD0): These bits select the function of pin PB12/STCK0/SCK1.

| Bit 9: PB12MD1 | Bit 8: PB12MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB12) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO0 transmit clock input (STCK0) | |
| | 1 | SCIF1 transmit/receive clock input/output (SCK1) | |

Bits 7 and 6—PB11 Mode Bits 1 and 0 (PB11MD1, PB11MD0): These bits select the function of pin PB11/STS0/RXD1.

| Bit 7: PB11MD1 | Bit 6: PB11MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB11) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO0 transmit start signal input/output (STS0) | |
| | 1 | SCIF1 receive data input (RXD1) | |

Bits 5 and 4—PB10 Mode Bits 1 and 0 (PB10MD1, PB10MD0): These bits select the function of pin PB10/STXD0/TXD1.

| Bit 5: PB10MD1 | Bit 4: PB10MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB10) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO0 transmit data output (STXD0) | |
| | 1 | SCIF1 transmit data output (TXD1) | |

Bits 3 and 2—PB9 Mode Bits 1 and 0 (PB9MD1, PB9MD0): These bits select the function of pin PB9/SRCK1/$\overline{\text{RTS}}$.

| Bit 3: PB9MD1 | Bit 2: PB9MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB9) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 receive clock input (SRCK1) | |
| | 1 | SCIF1 transmit request signal output ($\overline{\text{RTS}}$) | |

Bits 1 and 0—PB8 Mode Bits 1 and 0 (PB8MD1, PB8MD0): These bits select the function of pin PB8/SRS1/$\overline{\text{CTS}}$.

| Bit 1: PB8MD1 | Bit 0: PB8MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB8) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 receive start signal input (SRS1) | |
| | 1 | SCIF1 transmit clear signal input ($\overline{\text{CTS}}$) | |

**Port B Control Register 2 (PBCR2)**

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB7 MD1 | PB7 MD0 | PB6 MD1 | PB6 MD0 | PB5 MD1 | PB5 MD0 | PB4 MD1 | PB4 MD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB3 MD1 | PB3 MD0 | PB2 MD1 | PB2 MD0 | PB1 MD1 | PB1 MD0 | PB0 MD1 | PB0 MD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bits 15 and 14—PB7 Mode Bits 1 and 0 (PB7MD1, PB7MD0): These bits select the function of pin PB7/SRXD1/TIOCA2.

| Bit 15: PB7MD1 | Bit 14: PB7MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB7) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 receive data input (SRXD1) | |
| | 1 | TPU input capture input/output compare output (TIOCA2) | |

Bits 13 and 12—PB6 Mode Bits 1 and 0 (PB6MD1, PB6MD0): These bits select the function of pin PB6/STCK1/TIOCB2.

| Bit 13: PB6MD1 | Bit 12: PB6MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB6) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 transmit clock input (STCK1) | |
| | 1 | TPU input capture input/output compare output (TIOCB2) | |

Bits 11 and 10—PB5 Mode Bits 1 and 0 (PB5MD1, PB5MD0): These bits select the function of pin PB5/STS1/TIOCA1.

| Bit 11: PB5MD1 | Bit 10: PB5MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB5) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 transmit start signal input/output (STS1) | |
| | 1 | TPU input capture input/output compare output (TIOCA1) | |

Bits 9 and 8—PB4 Mode Bits 1 and 0 (PB4MD1, PB4MD0): These bits select the function of pin PB4/STXD1/TIOCB1.

| Bit 9: PB4MD1 | Bit 8: PB4MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB4) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO1 transmit data output (STXD1) | |
| | 1 | TPU input capture input/output compare output (TIOCB1) | |

Bits 7 and 6—PB3 Mode Bits 1 and 0 (PB3MD1, PB3MD0): These bits select the function of pin PB3/SRCK2/TIOCA0.

| Bit 7: PB3MD1 | Bit 6: PB3MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB3) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO2 receive clock input (SRCK2) | |
| | 1 | TPU input capture input/output compare output (TIOCA0) | |

Bits 5 and 4—PB2 Mode Bits 1 and 0 (PB2MD1, PB2MD0): These bits select the function of pin PB2/SRS2/TIOCB0.

| Bit 5: PB2MD1 | Bit 4: PB2MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB2) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO2 receive start signal input (SRS2) | |
| | 1 | TPU input capture input/output compare output (TIOCB0) | |

Bits 3 and 2—PB1 Mode Bits 1 and 0 (PB1MD1, PB1MD0): These bits select the function of pin PB1/SRXD2/TIOCC0.

| Bit 3: PB1MD1 | Bit 2: PB1MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB1) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO2 receive data input (SRXD2) | |
| | 1 | TPU input capture input/output compare output (TIOCC0) | |

Bits 1 and 0—PB0 Mode Bits 1 and 0 (PB0MD1, PB0MD0): These bits select the function of pin PB0/STCK2/TIOCD0.

| Bit 1: PB0MD1 | Bit 0: PB0MD0 | Description | |
|---|---|---|---|
| 0 | 0 | General input/output (PB0) | (Initial value) |
| | 1 | Reserved | |
| 1 | 0 | SIO2 transmit clock input (STCK2) | |
| | 1 | TPU input capture input/output compare output (TIOCD0) | |

### 20.3.4 Port B I/O Register (PBIOR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB15 IOR | PB14 IOR | PB13 IOR | PB12 IOR | PB11 IOR | PB10 IOR | PB9 IOR | PB8 IOR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB7 IOR | PB6 IOR | PB5 IOR | PB4 IOR | PB3 IOR | PB2 IOR | PB1 IOR | PB0 IOR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The port B I/O register (PBIOR) is a 16-bit read/write register that selects the input/output direction of the 16 multiplex pins in port B. Bits PB15IOR to PB0IOR correspond to individual pins in port B. PBIOR is enabled when port B pins function as general input pins (PB15 to PB0), and disabled otherwise. When port B pins function as PB15 to PB0, a pin becomes an output when the corresponding bit in PBIOR is set to 1, and an input when the bit is cleared to 0.

PBIOR is initialized to H'0000 by a power-on reset. It is not initialized by a manual reset or in standby mode or sleep mode.

## 21.1 Overview

This chip has two ports, designated A and B. Port A is a 13-bit input/output port, and port B is a 16-bit input/output port. The port pins are multiplexed as general input/output and other functions. (The function of multiplexed multiplex pins is selected by means of the pin function controller (PFC).) Ports A and B are each provided with a data register for storing pin data.

## 21.2 Port A

Port A is an input/output port with the 14 pins shown in figure 21.1. Of the 14 pins, the FTOB pin has no port data register bit, and is multiplexed as an internal clock pin.

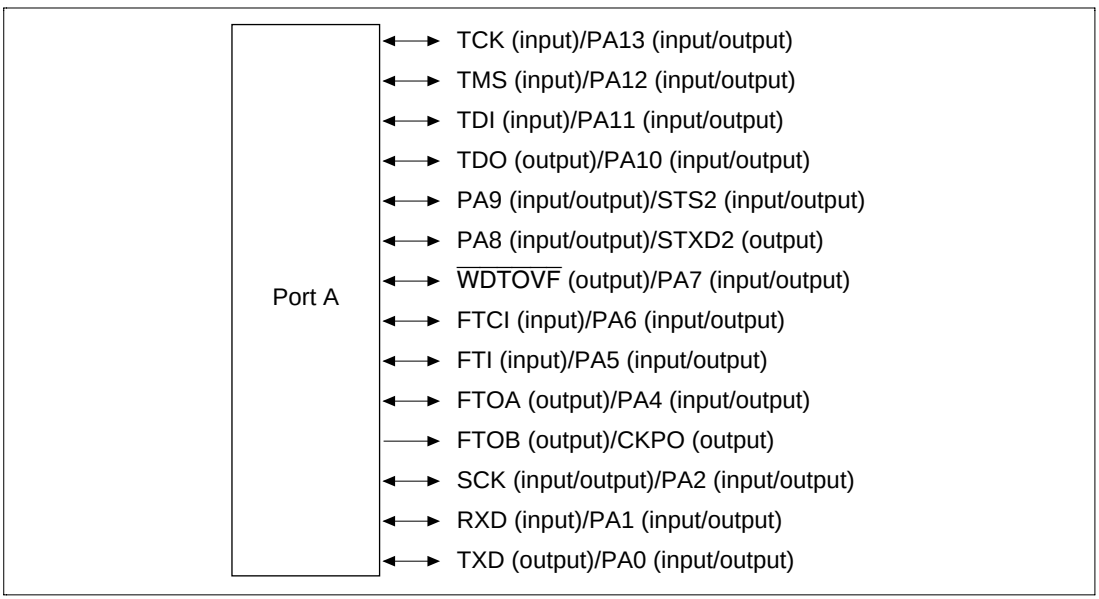



**Figure 21.1 Port A**

### 21.2.1 Register Configuration

The port A register is shown in table 21.1.

**Table 21.1 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Port A data register | PADR | R/W | H'0000 | H'FFFFFC84 | 8, 16 |

### 21.2.2 Port A Data Register (PADR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | PA13DR | PA12DR | PA11DR | PA10DR | PA9DR | PA8DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PA7DR | PA6DR | PA5DR | PA4DR | — | PA2DR | PA1DR | PA0DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

The port A data register (PADR) is a 16-bit read/write register that stores port A data. Bits 15, 14, and 3 are reserved: they always read 0, and the write value should always be 0. Bits PA13DR to PA4DR and PA2DR to PA0DR correspond to pins PA13 to PA4 and PA2 to PA0. When a pin functions as a general output, if a value is written to PADR, that value is output directly from the pin, and if PADR is read, the register value is returned directly regardless of the pin state. When a pin functions as a general input, if PADR is read the pin state, not the register value, is returned directly. If a value is written to PADR, although that value is written into PADR it does not affect the pin state. Table 21.2 summarizes port A data register read/write operations.

PADR is initialized to H'0000 by a power-on reset. It is not initialized by a manual reset or in standby mode or sleep mode.

**Table 21.2 Port A Data Register (PADR) Read/Write Operations**

| PAIOR | Pin Function | Read | Write |
|---|---|---|---|
| 0 | General input | Pin state | Value is written to PADR, but does not affect pin state |
| | Other than general input | Pin state | Value is written to PADR, but does not affect pin state |
| 1 | General output | PADR value | Write value is output from pin |
| | Other than general output | PADR value | Value is written to PADR, but does not affect pin state |

## 21.3 Port B

Port B is an input/output port with the 16 pins shown in figure 21.2.

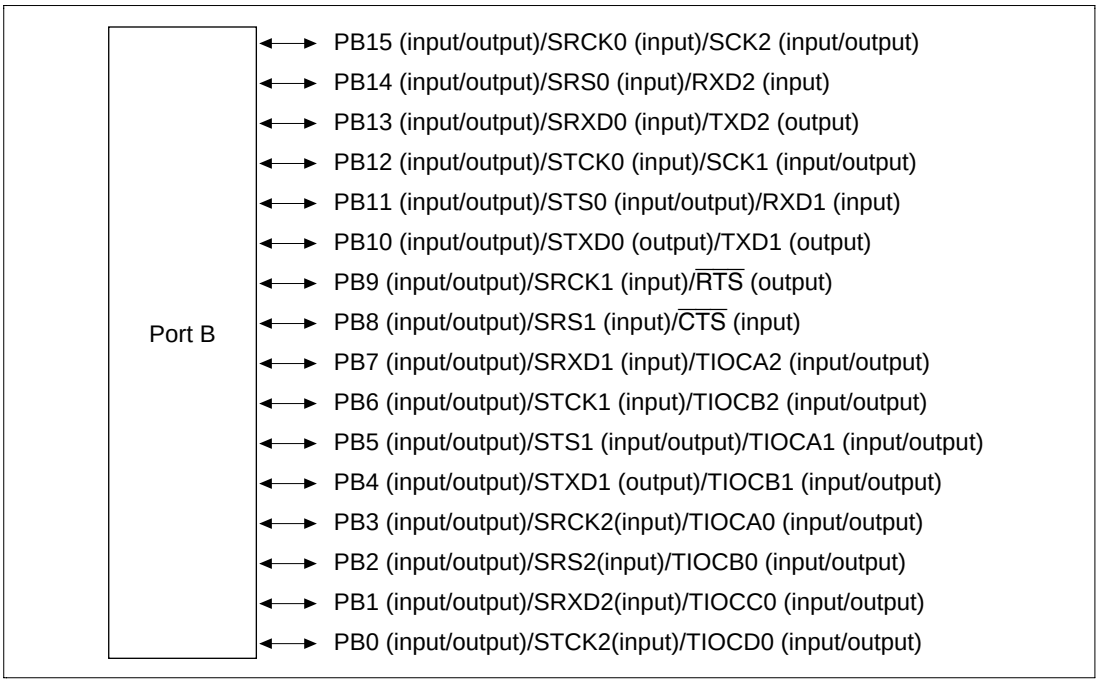



**Figure 21.2 Port B**

### 21.3.1 Register Configuration

Table 21.3 shows the port B register.

**Table 21.3 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Port B data register | PBDR | R/W | H'0000 | H'FFFFFC8C | 8, 16 |

### 21.3.2 Port B Data Register (PBDR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB15DR | PB14DR | PB13DR | PB12DR | PB11DR | PB10DR | PB9DR | PB8DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | PB7DR | PB6DR | PB5DR | PB4DR | PB3DR | PB2DR | PB1DR | PB0DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The port B data register (PBDR) is a 16-bit read/write register that stores port B data. Bits PB15DR to PB0DR correspond to pins PB15 to PB0. When a pin functions as a general output, if a value is written to PBDR, that value is output directly from the pin, and if PBDR is read, the register value is returned directly regardless of the pin state. When a pin functions as a general input, if PBDR is read the pin state, not the register value, is returned directly. If a value is written to PBDR, although that value is written into PBDR it does not affect the pin state. Table 21.4 shows port B data register read/write operations.

PBDR is initialized to H'0000 by a power-on reset. It is not initialized by a manual reset or in standby mode or sleep mode.

**Table 21.4 Port B Data Register (PBDR) Read/Write Operations**

| PBIOR | Pin Function | Read | Write |
|---|---|---|---|
| 0 | General input | Pin state | Value is written to PBDR, but does not affect pin state |
| | Other than general input | Pin state | Value is written to PBDR, but does not affect pin state |
| 1 | General output | PBDR value | Write value is output from pin |
| | Other than general output | PBDR value | Value is written to PBDR, but does not affect pin state |

# Section 22 Power-Down Modes

## 22.1 Overview

This chip has a module standby function (which reduces power consumption by selectively halting operation of unnecessary modules among the on-chip peripheral modules and the DSP unit), a sleep mode (which halts CPU functions), and a standby mode (which halts all functions).

### 22.1.1 Power-Down Modes

The following modes and function are provided as power-down modes:

1. Sleep mode
2. Standby mode
3. Module standby function
   (UBC, DMAC, DSP, DIVU, FRT, SCIF1–2, TPU, SIO0–2, SCI)

Table 22.1 shows the transition conditions for entering the modes from the program execution state, as well as the CPU and peripheral module states in each mode and the procedures for canceling each mode.

**Table 22.1 Power-Down Modes**

| Mode | Transition Condition | State | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | On-Chip Oscillation Circuit | CPU, Cache | DSP | BSC | UBC, DMAC, DIVU, FRT, SCIF1–2, TPU, SIO2–0, SCI | Pins | Canceling Procedure |
| Sleep mode | SLEEP instruction executed with SBY bit set to 0 in SBYCR1 | Runs | Halted | Halted | Runs | Runs | Runs | 1. Interrupt<br>2. DMA address error<br>3. Power-on reset<br>4. Manual reset |
| Standby mode | SLEEP instruction executed with SBY bit set to 1 in SBYCR1 | Halted | Halted | Halted | Halted, and register values held | UBC: Halted, and register values held<br>Other than UBC: Halted | Held or high impedance | 1. NMI interrupt<br>2. Power-on reset<br>3. Manual reset |
| Module standby function | MSTP bit for relevant module is set to 1 | Runs | Runs | When MSTP is 1, the clock supply is halted | Runs | When an MSTP bit is 1, the clock supply to the relevant module is halted | FRT, SCI, and SCIF1, 2 pins are initialized, and others operate | 1. Clear MSTP bit to 0<br>2. Power-on reset<br>3. Manual reset |

### 22.1.2 Register

Table 22.2 shows the register configuration.

**Table 22.2 Register Configuration**

| Name | Abbreviation | R/W | Initial Value | Address | Access Size |
|---|---|---|---|---|---|
| Standby control register 1 | SBYCR1 | R/W | H'00 | H'FFFFFE91 | 8 |
| Standby control register 2 | SBYCR2 | R/W | H'00 | H'FFFFFE93 | 8 |

## 22.2 Register Descriptions

### 22.2.1 Standby Control Register 1 (SBYCR1)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | SBY | HIZ | MSTP5<br>UBC | MSTP4<br>DMAC | MSTP3<br>DSP | MSTP2<br>DIVU | MSTP1<br>FRT | MSTP0<br>SCI |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Standby control register 1 (SBYCR1) is an 8-bit read/write register that sets the power-down mode. SBYCR is initialized to H'00 by a reset.

When a write is performed to SBYCR1, the entire chip stops for a maximum of 5 peripheral module clock (Pø) cycles.

Bit 7—Standby (SBY): Specifies transition to standby mode. To enter the standby mode, halt the WDT (set the TME bit in WTCSR to 0) and set the SBY bit.

| Bit 7: SBY | Description | |
|---|---|---|
| 0 | Executing a SLEEP instruction puts the chip into sleep mode | (Initial value) |
| 1 | Executing a SLEEP instruction puts the chip into standby mode | |

Note: Do not use the standby mode when the on-chip crystal oscillation circuit is used.

Bit 6—Port High Impedance (HIZ): Selects whether output pins are set to high impedance or retain the output state in standby mode. When HIZ = 0 (initial state), the specified pin retains its output state. When HIZ = 1, the pin goes to the high-impedance state. See Appendix A.1, Pin States during Resets, Power-Down States and Bus Release State, for which pins are controlled.

| Bit 6: HIZ | Description | |
|---|---|---|
| 0 | Pin state retained in standby mode | (Initial value) |
| 1 | Pin goes to high impedance in standby mode | |

Bit 5—Module Stop 5 (MSTP5): Specifies halting the clock supply to the user break controller (UBC). When the MSTP5 bit is set to 1, the supply of the clock to the UBC is halted. When the clock halts, the UBC registers retain their pre-halt state. Do not set this bit while the UBC is running.

| Bit 5: MSTP5 | Description | |
|---|---|---|
| 0 | UBC running | (Initial value) |
| 1 | Clock supply to UBC halted | |

Bit 4—Module Stop 4 (MSTP4): Specifies halting the clock supply to the DMAC. When MSTP4 bit is set to 1, the supply of the clock to the DMAC is halted. When the clock halts, the DMAC retains its pre-halt state. When MSTP4 is cleared to 0 and the DMAC begins running again, its starts operating from its pre-halt state. Set this bit while the DMAC is halted; this bit cannot be set while the DMAC is operating (transferring data).

| Bit 4: MSTP4 | Description | |
|---|---|---|
| 0 | DMAC running | (Initial value) |
| 1 | Clock supply to DMAC halted | |

Bit 3—Module Stop 3 (MSTP3): Specifies halting the clock supply to the DSP unit. When the MSTP3 bit is set to 1, the supply of the clock to the DSP unit is halted. When the clock halts, the operation result prior to the halt is retained. This bit should be set when the DSP unit is halted. When the DSP unit is halted, no instructions with a DSP register, MACH, or MACL as an operand can be used.

| Bit 3: MSTP3 | Description | |
|---|---|---|
| 0 | DSP running | (Initial value) |
| 1 | Clock supply to DSP halted | |

Bit 2—Module Stop 2 (MSTP2): Specifies halting the clock supply to the division unit (DIVU). When the MSTP2 bit is set to 1, the supply of the clock to DIVU is halted. When the clock halts, the DIVU registers retain their pre-halt state. This bit should be set when the DIVU is halted.

| Bit 2: MSTP2 | Description | |
|---|---|---|
| 0 | DIVU running | (Initial value) |
| 1 | Clock supply to DIVU halted | |

Bit 1—Module Stop 1 (MSTP1): Specifies halting the clock supply to the 16-bit free-running timer (FRT). When the MSTP1 bit is set to 1, the supply of the clock to the FRT is halted. When the clock halts, all FRT registers are initialized except the FRT interrupt vector register in INTC, which holds its previous value. When MSTP1 is cleared to 0 and the FRT begins running again, its starts operating from its initial state.

| Bit 1: MSTP1 | Description | |
|---|---|---|
| 0 | FRT running | (Initial value) |
| 1 | Clock supply to FRT halted | |

Bit 0—Module Stop 0 (MSTP0): Specifies halting the clock supply to the serial communication interface (SCI). When the MSTP0 bit is set to 1, the supply of the clock to the SCI is halted. When the clock halts, all SCI registers are initialized except the SCI interrupt vector register in INTC, which holds its previous value. When MSTP0 is cleared to 0 and the SCI begins running again, its starts operating from its initial state.

| Bit 0: MSTP0 | Description | |
|---|---|---|
| 0 | SCI running | (Initial value) |
| 1 | Clock supply to SCI halted | |

### 22.2.2 Standby Control Register 2 (SBYCR2)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit name: | — | — | MSTP11<br>TPU | MSTP10<br>SIO2 | MSTP9<br>SIO1 | MSTP8<br>SIO0 | MSTP7<br>SCIF2 | MSTP6<br>SCIF1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

Standby control register 2 (SBYCR2) is an 8-bit read/write register that sets the power-down mode state. SBYCR2 is initialized to H'00 by a reset.

When a write is performed to SBYCR2, the entire chip stops for a maximum of 5 peripheral module clock (Pø) cycles.

Bits 7 and 6—Reserved: These bits always read 0. The write value should always be 0.

Bit 5—Module Stop 11 (MSTP11): Specifies halting the clock supply to the 16-bit timer pulse unit (TPU). When the MSTP11 bit is set to 1, the supply of the clock to the TPU is halted. When the clock halts, the TPU retains its pre-halt state, and the TPU interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP11 is cleared to 0 and the clock supply to the TPU is resumed, the TPU starts operating again.

| Bit 5: MSTP11 | Description | |
|---|---|---|
| 0 | TPU running | (Initial value) |
| 1 | Clock supply to TPU halted | |

Bit 4—Module Stop 10 (MSTP10): Specifies halting the clock supply to SIO channel 2. When the MSTP10 bit is set to 1, the supply of the clock to SIO channel 2 is halted. When the clock halts, SIO channel 2 retains its pre-halt state, and the SIO channel 2 interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP10 is cleared to 0 and the clock supply to SIO channel 2 is restarted, operation starts again.

| Bit 4: MSTP10 | Description | |
|---|---|---|
| 0 | SIO channel 2 running | (Initial value) |
| 1 | Clock supply to SIO channel 2 halted | |

Bit 3—Module Stop 9 (MSTP9): Specifies halting the clock supply to SIO channel 1. When the MSTP9 bit is set to 1, the supply of the clock to SIO channel 1 is halted. When the clock halts, SIO channel 1 retains its pre-halt state, and the SIO channel 1 interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP9 is cleared to 0 and the clock supply to SIO channel 1 is restarted, operation starts again.

| Bit 3: MSTP9 | Description | |
|---|---|---|
| 0 | SIO channel 1 running | (Initial value) |
| 1 | Clock supply to SIO channel 1 halted | |

Bit 2—Module Stop 8 (MSTP8): Specifies halting the clock supply to SIO channel 0. When the MSTP8 bit is set to 1, the supply of the clock to SIO channel 0 is halted. When the clock halts, SIO channel 0 retains its pre-halt state, and the SIO channel 0 interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP8 is cleared to 0 and the clock supply to SIO channel 0 is restarted, operation starts again.

| Bit 2: MSTP8 | Description | |
|---|---|---|
| 0 | SIO channel 0 running | (Initial value) |
| 1 | Clock supply to SIO channel 0 halted | |

Bit 1—Module Stop 7 (MSTP7): Specifies halting the clock supply to SCIF2. When the MSTP7 bit is set to 1, the supply of the clock to SCIF2 is halted. When the clock halts, the SCIF2 registers are initialized, but the SCIF2 interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP7 is cleared to 0 and SCIF2 begins running again, it starts operating from its initial state.

| Bit 1: MSTP7 | Description | |
|---|---|---|
| 0 | SCIF2 running | (Initial value) |
| 1 | Clock supply to SCIF2 halted | |

Bit 0—Module Stop 6 (MSTP6): Specifies halting the clock supply to SCIF1. When the MSTP6 bit is set to 1, the supply of the clock to SCIF1 is halted. When the clock halts, the SCIF1 registers are initialized, but the SCIF1 interrupt vector register in the INTC retains its pre-halt value. Therefore, when MSTP6 is cleared to 0 and SCIF1 begins running again, it starts operating from its initial state.

| Bit 0: MSTP6 | Description | |
|---|---|---|
| 0 | SCIF1 running | (Initial value) |
| 1 | Clock supply to SCIF1 halted | |

## 22.3 Sleep Mode

### 22.3.1 Transition to Sleep Mode

Executing the SLEEP instruction when the SBY bit in SBYCR is 0 causes a transition from the program execution state to sleep mode. Although the CPU halts immediately after executing the SLEEP instruction, the contents of its internal registers remain unchanged. The on-chip peripheral modules continue to run in sleep mode.

### 22.3.2 Canceling Sleep Mode

Sleep mode is canceled by an interrupt, DMA address error, power-on reset, or manual reset.

**Cancellation by an Interrupt:** When an interrupt occurs, sleep mode is canceled and interrupt exception handling is executed. Sleep mode is not canceled if the interrupt cannot be accepted because its priority level is equal to or less than the mask level set in the CPU's status register (SR) or if an interrupt by an on-chip peripheral module is disabled at the peripheral module.

**Cancellation by a DMA Address Error:** If a DMA address error occurs, sleep mode is canceled and DMA address error exception handling is executed.

**Cancellation by a Power-On Reset:** A power-on reset cancels sleep mode.

**Cancellation by a Manual Reset:** A manual reset cancels sleep mode.

## 22.4 Standby Mode

### 22.4.1 Transition to Standby Mode

To enter standby mode, set the SBY bit to 1 in SBYCR1, then execute the SLEEP instruction. The chip switches from the program execution state to standby mode. The NMI interrupt cannot be accepted when the SLEEP instruction is executed, or for the following five cycles. In standby mode, the clock supply to all on-chip peripheral modules is halted as well as the CPU. CPU register contents are held, and some on-chip peripheral modules are initialized.

Do not use the standby mode when the on-chip crystal oscillation circuit is used.

**Table 22.3 Register States in Standby Mode**

| Module | Registers Initialized | Registers that Retain Data | Registers with Undefined Contents |
|---|---|---|---|
| Interrupt controller (INTC) | — | All registers | — |
| User break controller (UBC) | — | All registers | — |
| Bus state controller (BSC) | — | All registers | — |
| Direct memory access controller (DMAC) | DMA channel control register 0, 1<br>DMA operation register | • DMA source address registers 0 and 1<br>• DMA destination address registers 0 and 1<br>• DMA transfer count registers 0 and 1<br>• DMA request/ response selection control registers 0 and 1<br>• Vector number setting registers DMA0 and DMA1 | — |
| Division unit (DIVU) | — | — | All registers |
| Watchdog timer (WDT) | Bits 7–5 of the timer control/status register<br>Reset control/status register | Bits 2–0 of the timer control/status register<br>Timer counter | — |
| 16-bit free-running timer (FRT) | All registers | — | — |
| Serial communication interface (SCI) | All registers | — | — |
| Serial communication interface with FIFO (SCIF1–2) | All registers | — | — |
| Serial I/O (SIO0–2) | — | All registers | — |
| Hitachi user debug interface (H-UDI) | — | All registers | — |
| 16-bit timer pulse unit (TPU) | — | All registers | — |

**Table 22.3 Register States in Standby Mode (cont)**

| Module | Registers Initialized | Registers that Retain Data | Registers with Undefined Contents |
|---|---|---|---|
| Pin function controller (PFC) | — | All registers | — |
| Others | — | Standby control register 1, 2<br>Frequency modification register | — |

### 22.4.2 Canceling Standby Mode

Standby mode is canceled by an NMI interrupt, a power-on reset, or a manual reset.

**Cancellation by an NMI Interrupt:** When a rising edge or falling edge is detected in the NMI signal, after the elapse of the time set in the WDT timer control/status register, clocks are supplied to the entire chip, standby mode is canceled, and NMI exception handling begins. Insure that the interval set for the WDT is at least as long as the oscillation stabilization time. When standby mode is canceled by a falling edge in the NMI signal, insure that the NMI pin goes high when standby mode is entered (when the clock is halted), and goes low on recovering from standby mode when the Iø, Eø, and Pø clocks start after oscillation has stabilized. The low level at the NMI pin should be held for at least 3 cycles following the start of Iø, Eø, and Pø clock output after oscillation has stabilized. When standby mode is canceled by a rising edge in the NMI signal, insure that the NMI pin goes low when standby mode is entered (when the clock is halted), and goes high on recovering from standby mode when the Iø, Eø, and Pø clocks start after oscillation has stabilized. The high level at the NMI pin should be held for at least 3 cycles following the start of clock signal output after oscillation has stabilized.

**Cancellation by a Power-On Reset:** A power-on reset cancels standby mode.

**Cancellation by a Manual Reset:** A manual reset cancels standby mode.

### 22.4.3 Standby Mode Cancellation by NMI Interrupt

The following example describes moving to the standby mode upon the fall of the NMI signal and clearing the standby mode when the NMI signal rises. Figure 22.1 shows the timing.

When the NMI pin level changes from high to low after the NMI edge select bit (NMIE) of the interrupt control register (ICR) has been set to 0 (detect falling edge), an NMI interrupt is accepted. When the NMIE bit is set to 1 (detect rising edge) by the NMI exception service routine, the standby bit (SBY) of the standby control register 1 (SBYCR1) is set to 1 and a SLEEP instruction is executed, the standby mode is entered. The standby mode is cleared the next time the

NMI pin level changes from low level to high level. The level should be held for at least 3 cycles following the start of Iø, Eø, and Pø clock signal output after oscillation has stabilized.

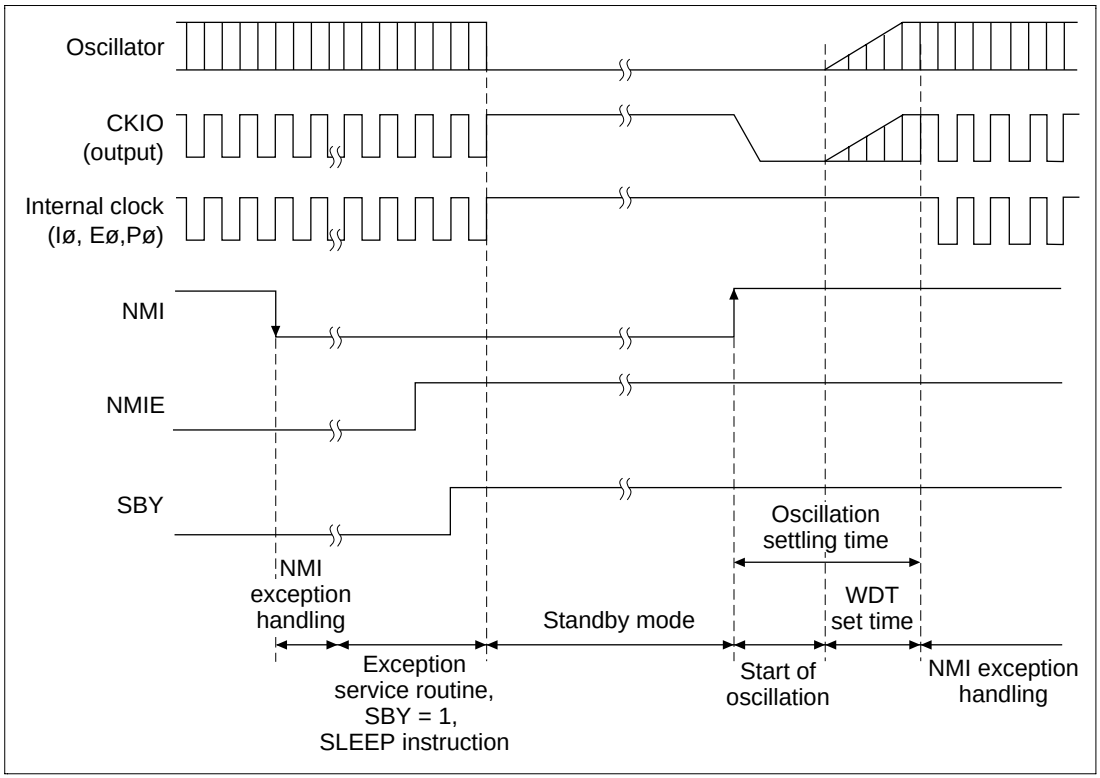



**Figure 22.1 Standby Mode Cancellation by NMI Interrupt**

### 22.4.4 Clock Pause Function

When the clock is input from the CKIO pin, the clock frequency can be modified or the clock stopped. The $\overline{\text{CKPREQ}}$/CKM pin is provided for this purpose. Note that clock pauses are not accepted while the watchdog timer (WDT) is operating (i.e. when the timer enable bit (TME) in the WDT's timer control/status register (WTCSR) is 1). When the clock pause request function is used, the standby bit (SBY) in the standby control register 1 (SBYCR1) must be set to 1 before inputting the request signal. The clock pause function is used as described below.

1. Set the TME bit in the watchdog timer's WTCSR register to 0, and set the SBY bit in SBYCR1 to 1.
2. Apply a low level to the $\overline{\text{CKPREQ}}$/CKM pin.
3. When the chip enters the standby state internally, a low level is output from the $\overline{\text{CKPACK}}$ pin.
4. After confirming that the $\overline{\text{CKPACK}}$ pin has gone low, perform clock halting or frequency modification.

5. To cancel the clock pause state (standby state), apply a high level to the $\overline{\text{CKPREQ}}$/CKM pin. (Inside the chip , the standby state is canceled by detecting a rising edge at the $\overline{\text{CKPREQ}}$/CKM pin.)
6. When PLL circuit 1 is operational, the WDT starts counting up inside the chip. When PLL circuit 1 is halted, the WDT is not activated.
7. When the internal clock stabilizes, the $\overline{\text{CKPACK}}$ pin goes high, giving external notification that the chip can be operated.

The standby state, all on-chip peripheral module states, and all pin states during clock pause are the same as in the normal standby mode. Figure 22.2 shows the timing chart for the clock pause function.

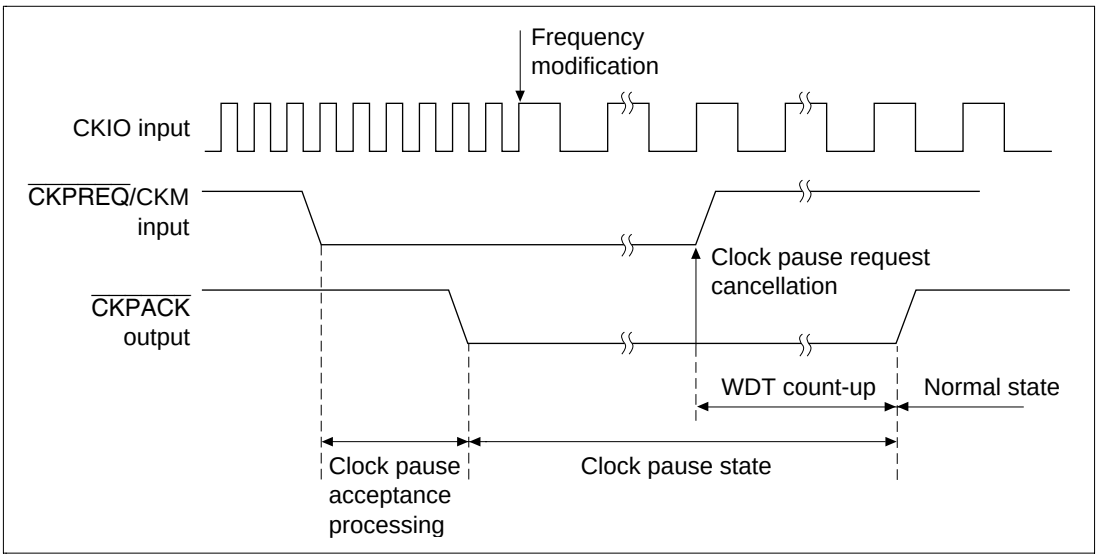



**Figure 22.2 Clock Pause Function Timing Chart (PLL Circuit 1 Operating)**

**Note:** After the $\overline{\text{CKPREQ}}$/CKM pin is driven low, do not access on-chip X/Y memory until the chip recovers from the clock pause state (standby state) and $\overline{\text{CKPACK}}$ output goes high.Figure 22.3 shows the clock pause function timing chart when the PLL circuit is halted.

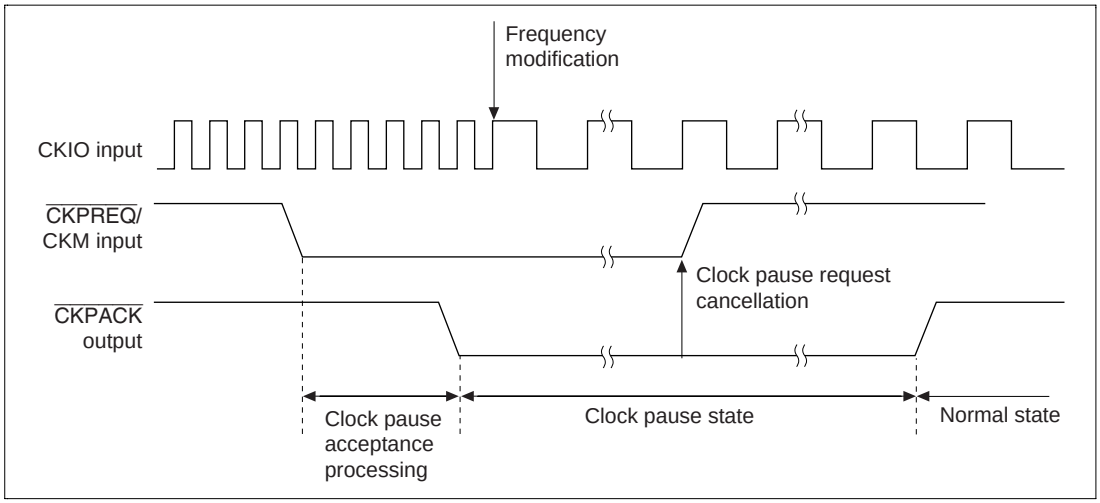



**Figure 22.3 Clock Pause Function Timing Chart (PLL Circuit 1 Halted)**

The clock pause state can be canceled by means of NMI input, in the same way as the normal standby state. The clock pause request should be canceled within four CKIO clock cycles after NMI input. Figure 22.4 shows the timing chart for clock pause state cancellation by means of NMI input (in the case of rising edge detection).

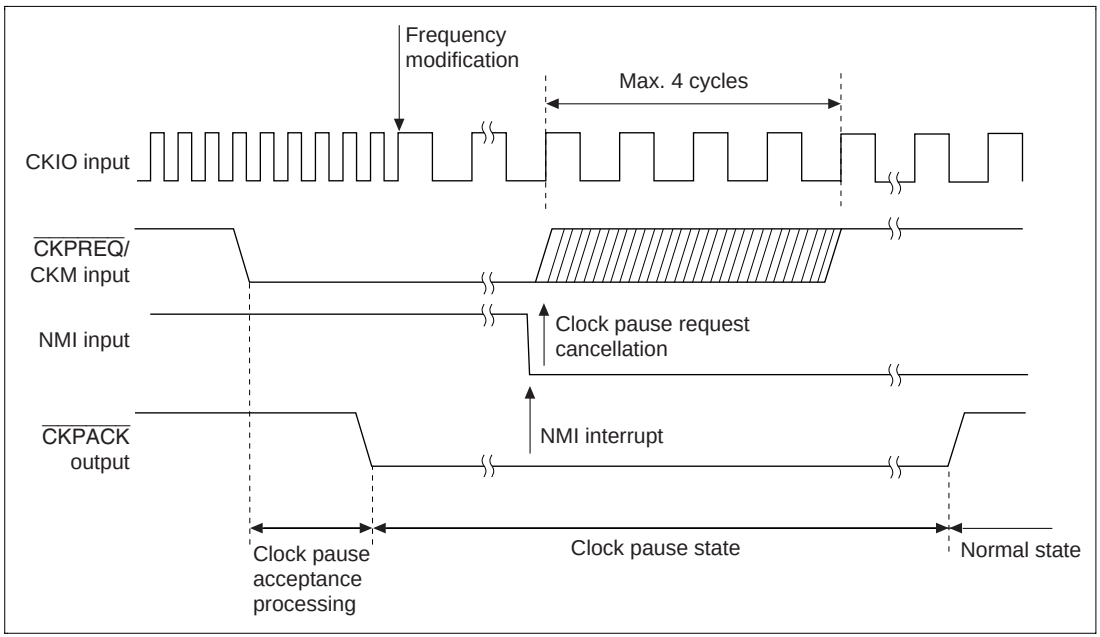



**Figure 22.4 Clock Pause Function Timing Chart (Cancellation by NMI Input)**

### 22.4.5 Notes on Standby Mode

1. When the chip enters standby mode during use of the cache, disable the cache before making the mode transition. Initialize the cache beforehand when the cache is used after returning to standby mode. The contents of the on-chip RAM are not retained in standby mode when cache is used as on-chip RAM.
2. If an on-chip peripheral register is written in the 10 clock cycles before the chip transits to standby mode, read the register before executing the SLEEP instruction.
3. When using clock mode 0, 1, or 2, the CKIO pin is the clock output pin. Note the following when standby mode is used in these clock modes. When standby mode is canceled by NMI, an unstable clock is output from the CKIO pin during the oscillation settling time after NMI input. This also applies to clock output in the case of cancellation by a power-on reset or manual reset. Power-on reset and manual reset input should be continued for a period at least equal to for the oscillation settling time.
4. Do not use the standby mode when the on-chip crystal oscillation circuit is used.

## 22.5 Module Standby Function

### 22.5.1 Transition to Module Standby Function

By setting one of bits MSTP11–MSTP0 to 1 in standby control register 1 or 2, the supply of the clock to the corresponding on-chip peripheral module or DSP unit can be halted. This function can be used to reduce the power consumption. Do not perform read/write operations for a module in module standby mode.

With the module standby function, the external pins of the DMAC and SIO0–SIO2 on-chip peripheral modules retain their states prior to halting, as do DMAC, DSP, DIVU, and SIO0–SIO2 registers. The external pins of the FRT, SCIF1–2, TPU, and SCI are reset and all their registers are initialized.

An on-chip peripheral module corresponding to a module standby bit must not be switched to the module standby state while it is running. Also, interrupts from a module placed in the module stop state should be disabled.

### 22.5.2 Clearing the Module Standby Function

Clear the module standby function by clearing the MSTP11–MSTP0 bits, or by a power-on reset or manual reset.

# Section 23 Electrical Characteristics

## 23.1 Absolute Maximum Ratings

Table 23.1 shows the absolute maximum ratings.

**Table 23.1 Absolute Maximum Ratings**

| Item | Symbol | Value | Unit |
|---|---|---|---|
| Power supply voltage | Vcc | –0.3 to +4.3 | V |
| Input voltage | Vin | –0.3 to Vcc +0.3 | V |
| Operating temperature | Topr | –20 to +75 | °C |
| Storage temperature | Tstg | –55 to +125 | °C |

Note: Permanent damage to the chip may result if the maximum ratings are exceeded.

## 23.2 DC Characteristics

Tables 23.2 and 23.3 show the DC characteristics.

**Table 23.2 DC Characteristics**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | | Symbol | Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Power supply voltage | | Vcc | 3.0 | 3.3 | 3.6 | V | |
| Input high voltage | $\overline{RES}$, NMI, MD4–MD0, | $V_{IH}$ | Vcc × 0.9 | — | Vcc + 0.3 | V | Standby mode |
| | $\overline{TRST}$, $\overline{CKPREQ}$/CKM, PB14/SRS0/RXD2, PB11/STS0/RXD1 | | Vcc × 0.9 | — | Vcc + 0.3 | V | Normal operation |
| | EXTAL, CKIO | | Vcc × 0.9 | — | Vcc + 0.3 | V | |
| | Other input pins | | Vcc × 0.7 | — | Vcc + 0.3 | V | |
| Input low voltage | $\overline{RES}$, NMI, MD4–MD0, | $V_{IL}$ | –0.3 | — | Vcc × 0.1 | V | Standby mode |
| | $\overline{TRST}$, $\overline{CKPREQ}$/CKM, PB14/SRS0/RXD2, PB11/STS0/RXD1 | | –0.3 | — | Vcc × 0.1 | V | Normal operation |
| | Other input pins | | –0.3 | — | Vcc × 0.2 | V | |
| Input leakage current | All input pins | \| Iin \| | — | — | 1.0 | µA | Vin = 0.5 to Vcc – 0.5 V |
| 3-state current leakage | All I/O and output pins (off state) | \| $I_{STi}$ \| | — | — | 1.0 | µA | Vin = 0.5 to Vcc – 0.5 V |
| Output high voltage | All output pins | $V_{OH}$ | Vcc – 0.5 | — | — | V | $I_{OH}$ = –200 µA |
| | | | Vcc – 1.0 | — | — | V | $I_{OH}$ = –1 mA |
| Output low voltage | All output pins | $V_{OL}$ | — | — | 0.4 | V | $I_{OL}$ = 1.6 mA |
| Pin capacitance | CAP1, CAP2 | Cin | — | — | 40 | pF | Vin = 0 V<br>f = 1 MHz |
| | All other input pins | | — | — | 15 | pF | Ta = 25°C |

**Table 23.2 DC Characteristics (cont)**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | | Symbol | Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Current dissipation | Normal operation | Icc | — | — | 250 | mA | 3.6 V, CPU operating clock = 60 MHz, DMAC used |
| | | | — | — | 250 | mA | 3.6 V, CPU operating clock = 60 MHz, DMAC not used |
| | Sleep mode | | — | — | 130 | mA | 3.6 V, CPU operating clock = 60 MHz, peripheral modules not used |
| | Standby mode | | — | 5 | 20 | µA | Ta ≤ 50°C |
| | | | — | — | 300 | µA | 50°C < Ta |

Notes: 1. Do not leave the PLLVcc and PLLVss pins open when the PLL circuit is not used. Connect the PLLVcc pin to Vcc and the PLLVss pin to Vss.
2. The current dissipation values are for VIH min = Vcc – 0.5 V and VIL max = 0.5 V with all output pins unloaded.

**Table 23.3 Permissible Output Currents**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Permissible output low current (per pin) | $I_{OL}$ | — | — | 2.0 | mA |
| Permissible output low current (total) | $\sum I_{OL}$ | — | — | 80 | mA |
| Permissible output high current (per pin) | $-I_{OH}$ | — | — | 2.0 | mA |
| Permissible output high current (total) | $\sum(-I_{OH})$ | — | — | 25 | mA |

Note: To protect chip reliability, do not exceed the output current values in table 23.3.

## 23.3 AC Characteristics

In principle, input is synchronous. Unless specified otherwise, ensure that the setup time and hold times for each input signal are observed.

**Table 23.4 Maximum Operating Frequencies**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | | Symbol | Min | Typ | Max | Unit | Notes |
|---|---|---|---|---|---|---|---|
| Operating frequency | CPU, DSP | f | 1 | — | 60 | MHz | $t_{Icyc}$ |
| | External bus (SDRAM not used) | | 1 | — | 30 | | $t_{Ecyc}$ |
| | External bus (SDRAM used) | | 1 | — | 60 | | $t_{Ecyc}$ |
| | Peripheral modules | | 1 | — | 30 | | $t_{Pcyc}$ |

### 23.3.1 Clock Timing

**Table 23.5 Clock Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| EXTAL clock input frequency | fEX | 1 | 30 | MHz | 23.1 |
| EXTAL clock input cycle time | tEXcyc | 33.3 | 1000 | ns | |
| EXTAL clock input low-level pulse width | tEXL | 8*[1], 12*[2] | — | ns | |
| EXTAL clock input high-level pulse width | tEXH | 8*[1], 12*[2] | — | ns | |
| EXTAL clock input rise time | tEXR | — | 4 | ns | |
| EXTAL clock input fall time | tEXF | — | 4 | ns | |
| CKIO clock input frequency | fCKI | 1 | 30 | MHz | 23.2 |
| CKIO clock input cycle time | tCKIcyc | 33.3 | 1000 | ns | |
| CKIO clock input low-level pulse width | tCKIL | 8*[3], 12*[4] | — | ns | |
| CKIO clock input high-level pulse width | tCKIH | 8*[3], 12*[4] | — | ns | |
| CKIO clock input rise time | tCKIR | — | 4 | ns | |
| CKIO clock input fall time | tCKIF | — | 4 | ns | |
| CKIO clock output frequency | fOP | 1 | 60 | MHz | 23.3 |
| CKIO clock output cycle time | tcyc | 16.7 | 1000 | ns | |
| CKIO clock output low-level pulse width | tCKOL | 3 | — | ns | |
| CKIO clock output high-level pulse width | tCKOH | 3 | — | ns | |
| CKIO clock rise time | tCKOR | — | 5 | ns | |
| CKIO clock fall time | tCKOF | — | 5 | ns | |
| Power-on oscillation stabilization time | tOSC1 | 10 | — | ms | 23.4 |
| Standby recovery oscillation stabilization time 1 | tOSC2 | 10 | — | ms | 23.5 |
| Standby recovery oscillation stabilization time 2 | tOSC3 | 10 | — | ms | 23.6 |
| PLL synchronization stabilization time | tPLL | 1 | — | ms | 23.7 |

Notes: 1. When PLL circuit 2 is operating
2. When PLL circuit 2 is not used
3. When PLL circuit 1 is operating
4. When PLL circuit 1 is not used

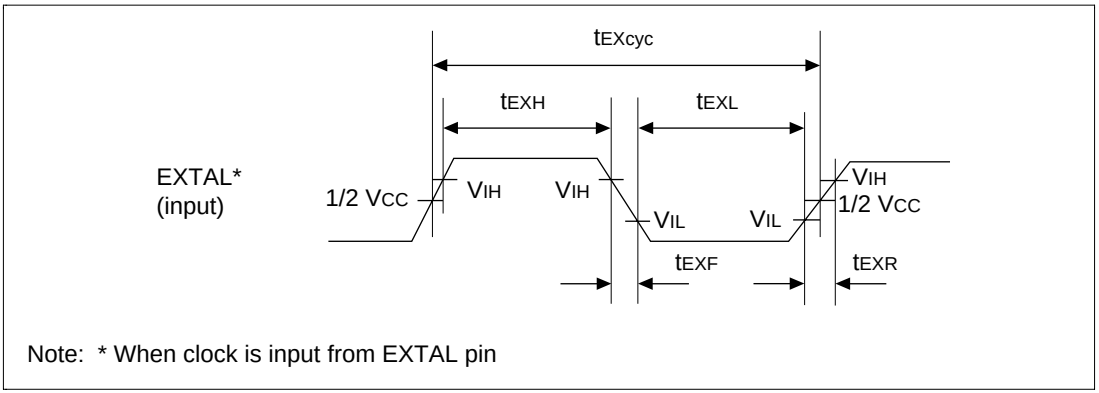



**Figure 23.1 EXTAL Clock Input Timing**

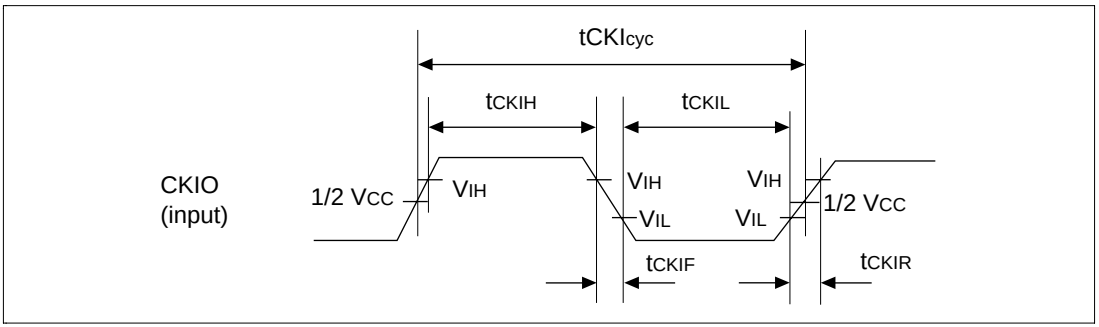



**Figure 23.2 CKIO Clock Input Timing**

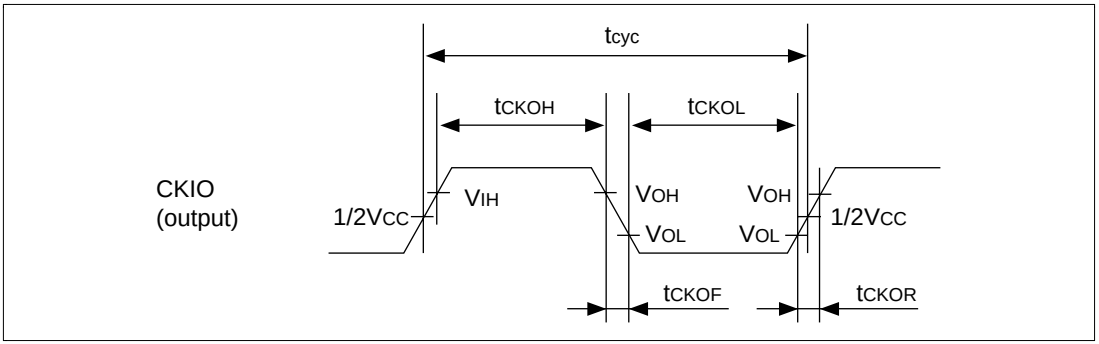
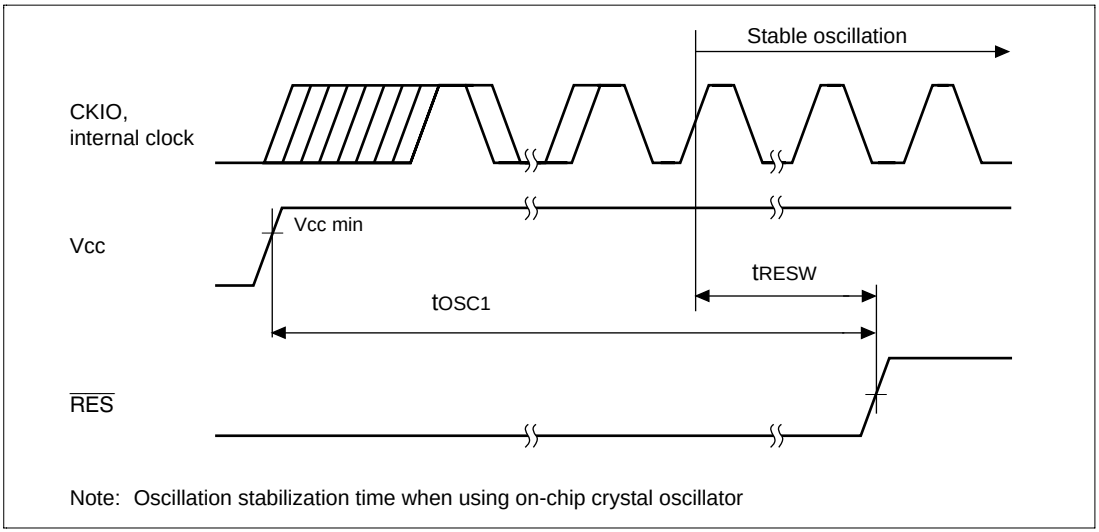
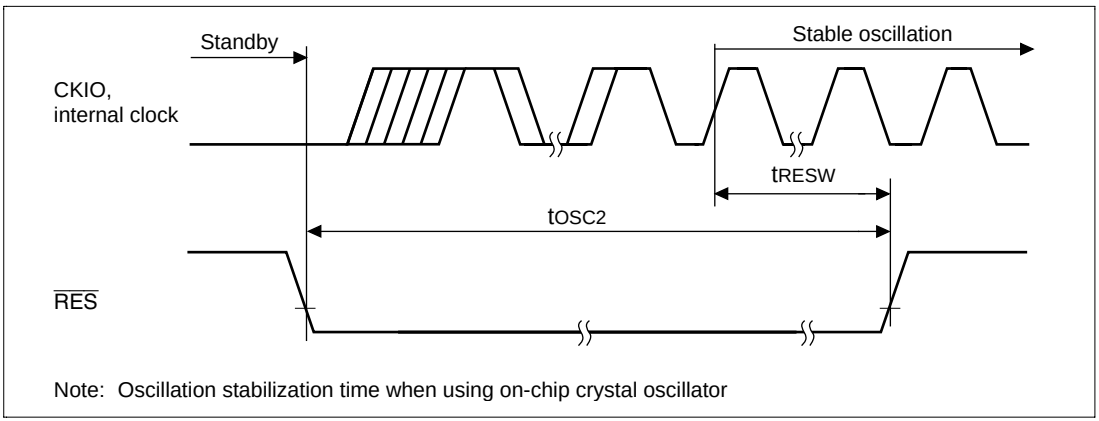



**Figure 23.3 CKIO Clock Output Timing**




Note: Oscillation stabilization time when using on-chip crystal oscillator

**Figure 23.4 Power-On Oscillation Stabilization Time at Power-On**




Note: Oscillation stabilization time when using on-chip crystal oscillator

**Figure 23.5 Oscillation Stabilization Time after Standby Recovery (Recovery by $\overline{\text{RES}}$)**

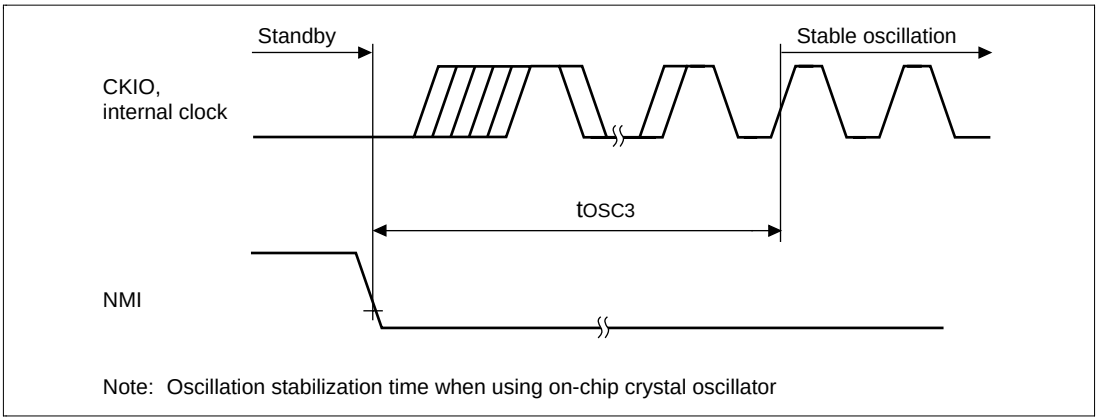



**Figure 23.6 Oscillation Stabilization Time after Standby Recovery (Recovery by NMI)**

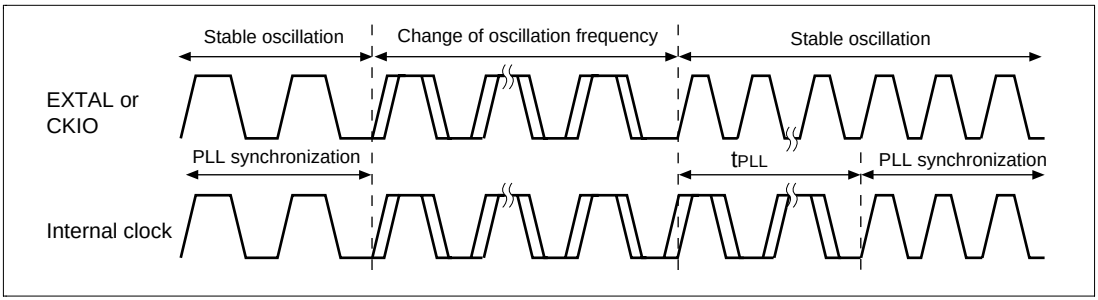



**Figure 23.7 PLL Synchronization Stabilization Time**

### 23.3.2 Control Signal Timing

**Table 23.6 Control Signal Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{RES}}$ rise and fall time | tRESr, tRESf | — | 200 | ns | 23.8 |
| $\overline{\text{RES}}$ pulse width | tRESW | 20 | — | tpcyc | |
| NMI reset setup time | tNMIRS | tpcyc + 10 | — | ns | |
| NMI reset hold time | tNMIRH | tpcyc + 10 | — | ns | |
| NMI rise and fall time | tNMIr, tNMIf | — | 200 | ns | |
| $\overline{\text{RES}}$ setup time* | tRESS | 3tEcyc + 40 | — | ns | 23.9 |
| NMI setup time* | tNMIS | 40 | — | ns | |
| $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ setup time* | tIRLS | 30 | — | ns | |
| NMI hold time | tNMIH | 20 | — | ns | |
| $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ hold time* | tIRLH | 20 | — | ns | |
| $\overline{\text{BRLS}}$ setup time | tBLSS | 7 | — | ns | 23.10 |
| $\overline{\text{BRLS}}$ hold time | tBLSH | 8 | — | ns | |
| $\overline{\text{BGR}}$ delay time | tBGRD | — | 15 | ns | |
| Bus tri-state delay time | tBOFF | 0 | 35 | ns | |
| Bus buffer on time | tBON | 0 | 35 | ns | |

Note: * The $\overline{\text{RES}}$, NMI, and $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ signals are asynchronous inputs. If the setup times shown here are observed, a transition is judged to have occurred at the fall of the clock; if the setup times cannot be observed, recognition may be delayed until the next fall of the clock.

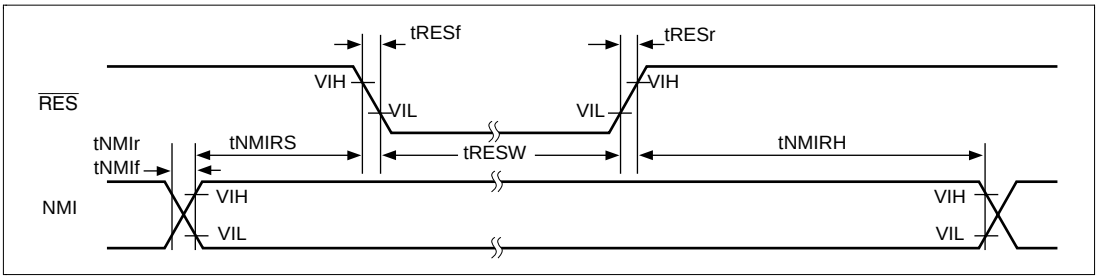



**Figure 23.8 Reset Input Timing**

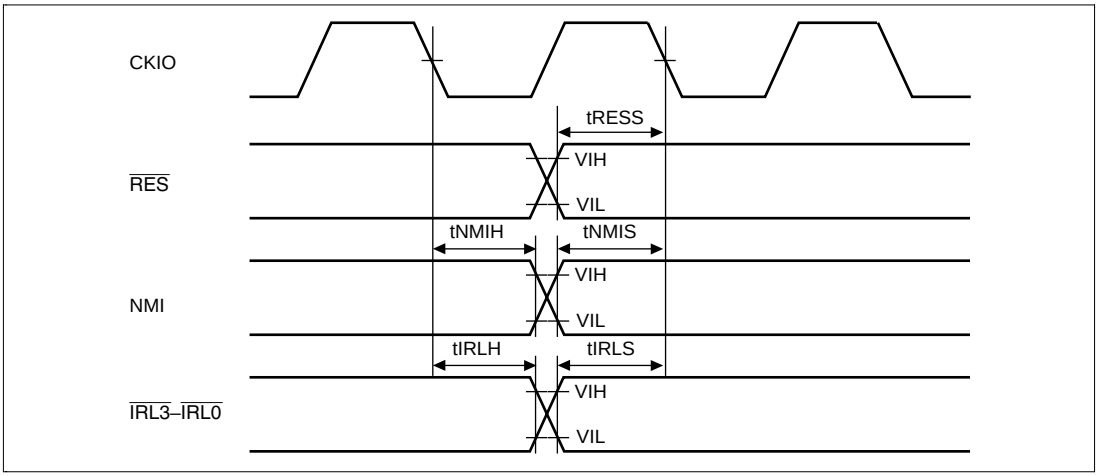



**Figure 23.9 Interrupt Signal Input Timing**

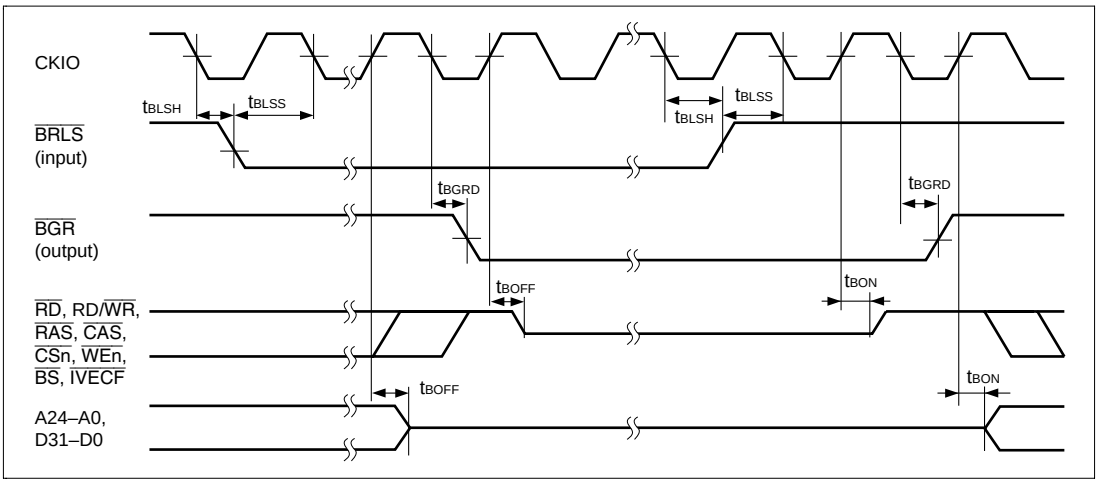



**Figure 23.10 Bus Release Timing**

### 23.3.3 Bus Timing

**Table 23.7 PLL-On Bus Timing [Modes 0 and 4]**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Address delay time | tAD | 1 | 20 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24 to 28, 30 to 34, 37 to 40, 42 to 44 |
| $\overline{BS}$ delay time | tBSD | — | 15 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24, 25, 28, 30, 31, 33, 34, 39, 42 to 44 |
| $\overline{CS}$ delay time 1 | tCSD1 | 1 | 14 | ns | 23.11, 12, 15, 16, 18, 20, 22 to 25, 28, 30 to 34, 39, 41, 42 |
| $\overline{CS}$ delay time 2 | tCSD2 | — | 14 | ns | 23.11, 12, 33, 34, 39, 42 |
| Read/write delay time | tRWD | 1 | 14 | ns | 23.11, 12, 15 ,16, 18, 20 to 22, 24, 25, 28 to 34, 39, 42 to 44 |
| Read strobe delay time 1 | tRSD1 | — | 14 | ns | 23.11, 12, 15, 16, 22, 30, 33, 34, 37, 39, 40, 42 to 44 |
| Read data setup time 1 | tRDS1 | 8 | — | ns | 23.11, 33, 37, 42 to 44 |
| Read data setup time 2 (EDO) | tRDS2 | 8 | — | ns | 23.39, 40 |
| Read data setup time 3 (SDRAM) | tRDS3 | 8 | — | ns | 23.15, 16 |
| Read data hold time 2 | tRDH2 | 2 | — | ns | 23.11, 42 |
| Read data hold time 4 (SDRAM) | tRDH4 | 6 | — | ns | 23.15, 16 |
| Read data hold time 5 (DRAM) | tRDH5 | 2 | — | ns | 23.33, 37 |
| Read data hold time 6 (EDO) | tRDH6 | 6 | — | ns | 23.39, 40 |
| Read data hold time 7 (EDO) | tRDH7 | 1 | — | ns | 23.39 |
| Read data hold time 8 (interrupt vector) | tRDH8 | 2 | — | ns | 23.43, 44 |
| Write enable delay time 1 | tWED1 | — | 14 | ns | 23.11, 12 |
| Write data delay time 1 (except Eø: Iø = 1:1) | tWDD1 | — | 22 | ns | 23.12, 22, 24, 26, 34, 38 |
| Write data delay time 2 (Eø: Iø = 1:1) | tWDD2 | — | 14 | ns | 23.25, 27 |
| Write data hold time 1 | tWDH1 | 1 | — | ns | 23.12, 22, 24 to 27, 34, 38 |
| Data buffer on time | tDON | — | 15 | ns | 23.12, 22, 24, 25, 34 |
| Data buffer off time | tDOF | — | 15 | ns | 23.12, 22, 24, 25, 34 |
| DACK delay time 1 | tDACD1 | — | 14 | ns | 23.11, 12, 15, 18, 20, 22, 24, 25, 28, 33, 34, 37 to 40, 42 |
| DACK delay time 2 | tDACD2 | — | 14 | ns | 23.11, 12, 33, 34, 37 to 40, 42 |

**Table 23.7 PLL-On Bus Timing [Modes 0 and 4] (cont)**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{WAIT}}$ setup time | tWTS | 7 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{WAIT}}$ hold time | tWTH | 8 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{RAS}}$ delay time 1 (SDRAM) | tRASD1 | 1 | 14 | ns | 23.15 to 18, 20 to 25, 28 to 32 |
| $\overline{\text{RAS}}$ delay time 2 (DRAM, EDO) | tRASD2 | — | 14 | ns | 23.33, 34, 39, 41 |
| $\overline{\text{RAS}}$ delay time 3 (EDO) | tRASD3 | — | 14 | ns | 23.39 |
| $\overline{\text{CAS}}$ delay time 1 (SDRAM) | tCASD1 | 1 | 14 | ns | 23.15, 16, 17, 18, 22 to 28, 30 to 32, 42 |
| $\overline{\text{CAS}}$ delay time 2 (DRAM) | tCASD2 | — | 14 | ns | 23.33, 34, 37 to 41 |
| DQM delay time | tDQMD | 1 | 14 | ns | 23.15, 16, 18 to 20, 22, 24 to 29 |
| CKE delay time | tCKED | 1 | 14 | ns | 23.32 |
| $\overline{\text{OE}}$ delay time 1 | tOED1 | — | 14 | ns | 23.39 |
| $\overline{\text{OE}}$ delay time 2 | tOED2 | — | 14 | ns | 23.39 |
| $\overline{\text{IVECF}}$ delay time | tIVD | — | 15 | ns | 23.43, 44 |
| Row address setup time | tASR | 0 | — | ns | 23.33, 34, 39 |
| Column address setup time | tASC | 0 | — | ns | 23.33, 34, 37, 38, 39 |
| Data input setup time | tDS | 0 | — | ns | 23.34, 38 |
| Read/write address setup time | tAS | 0 | — | ns | 23.11, 12 |
| REFOUT delay time | tREFOD | — | 15 | ns | 23.46 |

**Table 23.8 PLL-On Bus Timing [Modes 1 and 5]**

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Address delay time | tAD | 1 + β | 14 + α | ns | 23.11, 12, 15, 16, 18, 20, 22, 24 to 28, 30 to 34, 37 to 40, 42 to 44 |
| $\overline{BS}$ delay time | tBSD | — | 15 + α | ns | 23.11, 12, 15, 16, 18, 20, 22, 24, 25, 28, 30, 31, 33, 34, 39, 42 to 44 |
| $\overline{CS}$ delay time 1 | tCSD1 | 1 + β | 14 + α | ns | 23.11, 12, 15, 16, 18, 20, 22 to 25, 28, 30 to 34, 39, 41, 42 |
| $\overline{CS}$ delay time 2 | tCSD2 | — | 14 + α | ns | 23.11, 12, 33, 34, 39, 42 |
| Read/write delay time | tRWD | 1 + β | 14 + α | ns | 23.11, 12, 15 ,16, 18, 20 to 22, 24, 25, 28 to 34, 39, 42 to 44 |
| Read strobe delay time 1 | tRSD1 | — | 14 + α | ns | 23.11, 12, 15, 16, 22, 30, 33, 34, 37, 39, 40, 42 to 44 |
| Read data setup time 1 | tRDS1 | 8 - β | — | ns | 23.11, 33, 37, 42 to 44 |
| Read data setup time 2 (EDO) | tRDS2 | 8 - β | — | ns | 23.39, 40 |
| Read data setup time 3 (SDRAM) | tRDS3 | 8 - β | — | ns | 23.15, 16 |
| Read data hold time 2 | tRDH2 | 0 | — | ns | 23.11, 42 |
| Read data hold time 4 (SDRAM) | tRDH4 | 3 + α | — | ns | 23.15, 16 |
| Read data hold time 5 (DRAM) | tRDH5 | 0 | — | ns | 23.33, 37 |
| Read data hold time 6 (EDO) | tRDH6 | 3 + α | — | ns | 23.39, 40 |
| Read data hold time 7 (EDO) | tRDH7 | 1 | — | ns | 23.39 |
| Read data hold time 8 (interrupt vector) | tRDH8 | 2 | — | ns | 23.43, 44 |
| Write enable delay time 1 | tWED1 | — | 14 + α | ns | 23.11, 12 |
| Write data delay time 1 (except Eø: Iø = 1:1) | tWDD1 | — | 22 + α | ns | 23.12, 22, 24, 26, 34, 38 |
| Write data delay time 2 (Eø: Iø = 1:1) | tWDD2 | — | 14 + α | ns | 23.25, 27 |
| Write data hold time 1 | tWDH1 | 1 + β | — | ns | 23.12, 22, 24 to 27, 34, 38 |
| Data buffer on time | tDON | — | 15 + α | ns | 23.12, 22, 24, 25, 34 |
| Data buffer off time | tDOF | — | 15 + α | ns | 23.12, 22, 24, 25, 34 |
| DACK delay time 1 | tDACD1 | — | 14 + α | ns | 23.11, 12, 15, 18, 20, 22, 24, 25, 28, 33, 34, 37 to 40, 42 |
| DACK delay time 2 | tDACD2 | — | 14 + α | ns | 23.11, 12, 33, 34, 37 to 40, 42 |

**Table 23.8 PLL-On Bus Timing [Modes 1 and 5] (cont)**

| | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{WAIT}}$ setup time | tWTS | 10 - β | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{WAIT}}$ hold time | tWTH | 5 + α | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{RAS}}$ delay time 1 (SDRAM) | tRASD1 | 1 + β | 14 + α | ns | 23.15 to 18, 20 to 25, 28 to 32 |
| $\overline{\text{RAS}}$ delay time 2 (DRAM, EDO) | tRASD2 | — | 14 | ns | 23.33, 34, 39, 41 |
| $\overline{\text{RAS}}$ delay time 3 (EDO) | tRASD3 | — | 14 + α | ns | 23.39 |
| $\overline{\text{CAS}}$ delay time 1 (SDRAM) | tCASD1 | 1 + β | 14 + α | ns | 23.15, 16, 17, 18, 22 to 28, 30 to 32, 42 |
| $\overline{\text{CAS}}$ delay time 2 (DRAM) | tCASD2 | — | 14 + α | ns | 23.33, 34, 37 to 41 |
| DQM delay time | tDQMD | 1 + β | 14 + α | ns | 23.15, 16, 18 to 20, 22, 24 to 29 |
| CKE delay time | tCKED | 1 + β | 14 + α | ns | 23.32 |
| $\overline{\text{OE}}$ delay time 1 | tOED1 | — | 14 + α | ns | 23.39 |
| $\overline{\text{OE}}$ delay time 2 | tOED2 | — | 14 + α | ns | 23.39 |
| $\overline{\text{IVECF}}$ delay time | tIVD | — | 15 + α | ns | 23.43, 44 |
| Row address setup time | tASR | 0 | — | ns | 23.33, 34, 39 |
| Column address setup time | tASC | 0 | — | ns | 23.33, 34, 37, 38, 39 |
| Data input setup time | tDS | 0 | — | ns | 23.34, 38 |
| Read/write address setup time | tAS | 0 | — | ns | 23.11, 12 |
| REFOUT delay time | tREFOD | — | 15 + α | ns | 23.46 |

Note: $\alpha = 1/4\phi + 3$ (ns), $\beta = 1/4\phi - 3$ (ns), where ø is the system clock

**Table 23.9 PLL-Off Bus Timing [Mode 2]**

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Address delay time | tAD | — | 19 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24 to 28, 30 to 34, 37 to 40, 42 to 44 |
| $\overline{BS}$ delay time | tBSD | — | 20 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24, 25, 28, 30, 31, 33, 34, 39, 42 to 44 |
| $\overline{CS}$ delay time 1 | tCSD1 | — | 19 | ns | 23.11, 12, 15, 16, 18, 20, 22 to 25, 28, 30 to 34, 39, 41, 42 |
| $\overline{CS}$ delay time 2 | tCSD2 | — | 19 | ns | 23.11, 12, 33, 34, 39, 42 |
| Read/write delay time | tRWD | — | 19 | ns | 23.11, 12, 15 ,16, 18, 20 to 22, 24, 25, 28 to 34, 39, 42 to 44 |
| Read strobe delay time 1 | tRSD1 | — | 19 | ns | 23.11, 12, 15, 16, 22, 30, 33, 34, 37, 39, 40, 42 to 44 |
| Read data setup time 1 | tRDS1 | 8 | — | ns | 23.11, 33, 37, 42 to 44 |
| Read data setup time 2 (EDO) | tRDS2 | 8 | — | ns | 23.39, 40 |
| Read data setup time 3 (SDRAM) | tRDS3 | 8 | — | ns | 23.15, 16 |
| Read data hold time 2 | tRDH2 | 0 | — | ns | 23.11, 42 |
| Read data hold time 4 (SDRAM) | tRDH4 | 8 | — | ns | 23.15, 16 |
| Read data hold time 5 (DRAM) | tRDH5 | 0 | — | ns | 23.33, 37 |
| Read data hold time 6 (EDO) | tRDH6 | 8 | — | ns | 23.39, 40 |
| Read data hold time 7 (EDO) | tRDH7 | 1 | — | ns | 23.39 |
| Read data hold time 8 (interrupt vector) | tRDH8 | 2 | — | ns | 23.43, 44 |
| Write enable delay time 1 | tWED1 | — | 19 | ns | 23.11, 12 |
| Write data delay time 1 (except Eø: Iø = 1:1) | tWDD1 | — | 27 | ns | 23.12, 22, 24, 26, 34, 38 |
| Write data delay time 2 (Eø: Iø = 1:1) | tWDD2 | — | 19 | ns | 23.25, 27 |
| Write data hold time 1 | tWDH1 | 1 | — | ns | 23.12, 22, 24 to 27, 34, 38 |
| Data buffer on time | tDON | — | — | ns | 23.12, 22, 24, 25, 34 |
| Data buffer off time | tDOF | — | 20 | ns | 23.12, 22, 24, 25, 34 |
| DACK delay time 1 | tDACD1 | — | 20 | ns | 23.11, 12, 15, 18, 20, 22, 24, 25, 28, 33, 34, 37 to 40, 42 |
| DACK delay time 2 | tDACD2 | — | 19 | ns | 23.11, 12, 33, 34, 37 to 40, 42 |

**Table 23.9 PLL-Off Bus Timing [Mode 2] (cont)**

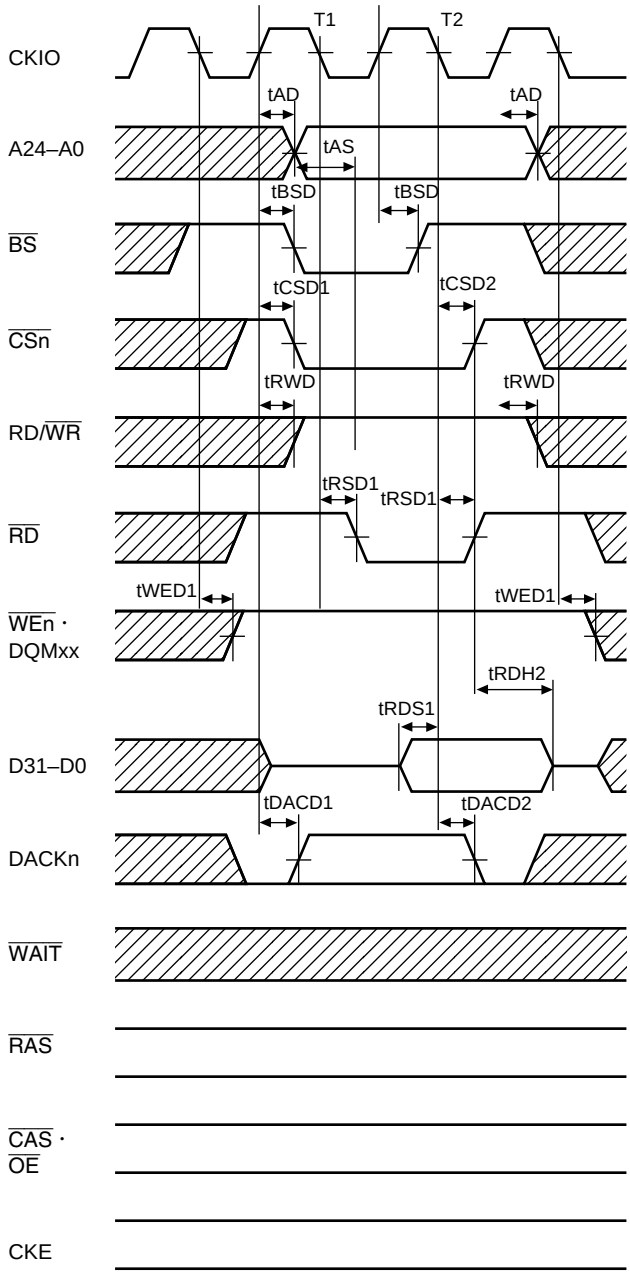
| | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{WAIT}}$ setup time | tWTS | 10 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{WAIT}}$ hold time | tWTH | 10 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{RAS}}$ delay time 1 (SDRAM) | tRASD1 | — | 19 | ns | 23.15 to 18, 20 to 25, 28 to 32 |
| $\overline{\text{RAS}}$ delay time 2 (DRAM, EDO) | tRASD2 | — | 19 | ns | 23.33, 34, 39, 41 |
| $\overline{\text{RAS}}$ delay time 3 (EDO) | tRASD3 | — | 19 | ns | 23.39 |
| $\overline{\text{CAS}}$ delay time 1 (SDRAM) | tCASD1 | — | 19 | ns | 23.15, 16, 17, 18, 22 to 28, 30 to 32, 42 |
| $\overline{\text{CAS}}$ delay time 2 (DRAM) | tCASD2 | — | 19 | ns | 23.33, 34, 37 to 41 |
| DQM delay time | tDQMD | — | 19 | ns | 23.15, 16, 18 to 20, 22, 24 to 29 |
| CKE delay time | tCKED | — | 19 | ns | 23.32 |
| $\overline{\text{OE}}$ delay time 1 | tOED1 | — | 19 | ns | 23.39 |
| $\overline{\text{OE}}$ delay time 2 | tOED2 | — | 19 | ns | 23.39 |
| $\overline{\text{IVECF}}$ delay time | tIVD | — | 20 | ns | 23.43, 44 |
| Row address setup time | tASR | 0 | — | ns | 23.33, 34, 39 |
| Column address setup time | tASC | 0 | — | ns | 23.33, 34, 37, 38, 39 |
| Data input setup time | tDS | 0 | — | ns | 23.34, 38 |
| Read/write address setup time | tAS | 0 | — | ns | 23.11, 12 |
| REFOUT delay time | tREFOD | — | 20 | ns | 23.46 |

**Table 23.10 PLL-Off Bus Timing [Mode 6]**

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Address delay time | tAD | — | 27 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24 to 28, 30 to 34, 37 to 40, 42 to 44 |
| $\overline{BS}$ delay time | tBSD | — | 28 | ns | 23.11, 12, 15, 16, 18, 20, 22, 24, 25, 28, 30, 31, 33, 34, 39, 42 to 44 |
| $\overline{CS}$ delay time 1 | tCSD1 | — | 27 | ns | 23.11, 12, 15, 16, 18, 20, 22 to 25, 28, 30 to 34, 39, 41, 42 |
| $\overline{CS}$ delay time 2 | tCSD2 | — | 27 | ns | 23.11, 12, 33, 34, 39, 42 |
| Read/write delay time | tRWD | — | 27 | ns | 23.11, 12, 15 ,16, 18, 20 to 22, 24, 25, 28 to 34, 39, 42 to 44 |
| Read strobe delay time 1 | tRSD1 | — | 27 | ns | 23.11, 12, 15, 16, 22, 30, 33, 34, 37, 39, 40, 42 to 44 |
| Read data setup time 1 | tRDS1 | 8 | — | ns | 23.11, 33, 37, 42 to 44 |
| Read data setup time 2 (EDO) | tRDS2 | 8 | — | ns | 23.39, 40 |
| Read data setup time 3 (SDRAM) | tRDS3 | 8 | — | ns | 23.15, 16 |
| Read data hold time 2 | tRDH2 | 0 | — | ns | 23.11, 42 |
| Read data hold time 4 (SDRAM) | tRDH4 | 16 | — | ns | 23.15, 16 |
| Read data hold time 5 (DRAM) | tRDH5 | 0 | — | ns | 23.33, 37 |
| Read data hold time 6 (EDO) | tRDH6 | 16 | — | ns | 23.39, 40 |
| Read data hold time 7 (EDO) | tRDH7 | 1 | — | ns | 23.39 |
| Read data hold time 8 (interrupt vector) | tRDH8 | 2 | — | ns | 23.43, 44 |
| Write enable delay time 1 | tWED1 | — | 19 | ns | 23.11, 12 |
| Write data delay time 1 (except Eø: Iø = 1:1) | tWDD1 | — | 27 | ns | 23.12, 22, 24, 26, 34, 38 |
| Write data delay time 2 (Eø: Iø = 1:1) | tWDD2 | — | 19 | ns | 23.25, 27 |
| Write data hold time 1 | tWDH1 | 1 | — | ns | 23.12, 22, 24 to 27, 34, 38 |
| Data buffer on time | tDON | — | 28 | ns | 23.12, 22, 24, 25, 34 |
| Data buffer off time | tDOF | — | 28 | ns | 23.12, 22, 24, 25, 34 |
| DACK delay time 1 | tDACD1 | — | 27 | ns | 23.11, 12, 15, 18, 20, 22, 24, 25, 28, 33, 34, 37 to 40, 42 |
| DACK delay time 2 | tDACD2 | — | 27 | ns | 23.11, 12, 33, 34, 37 to 40, 42 |

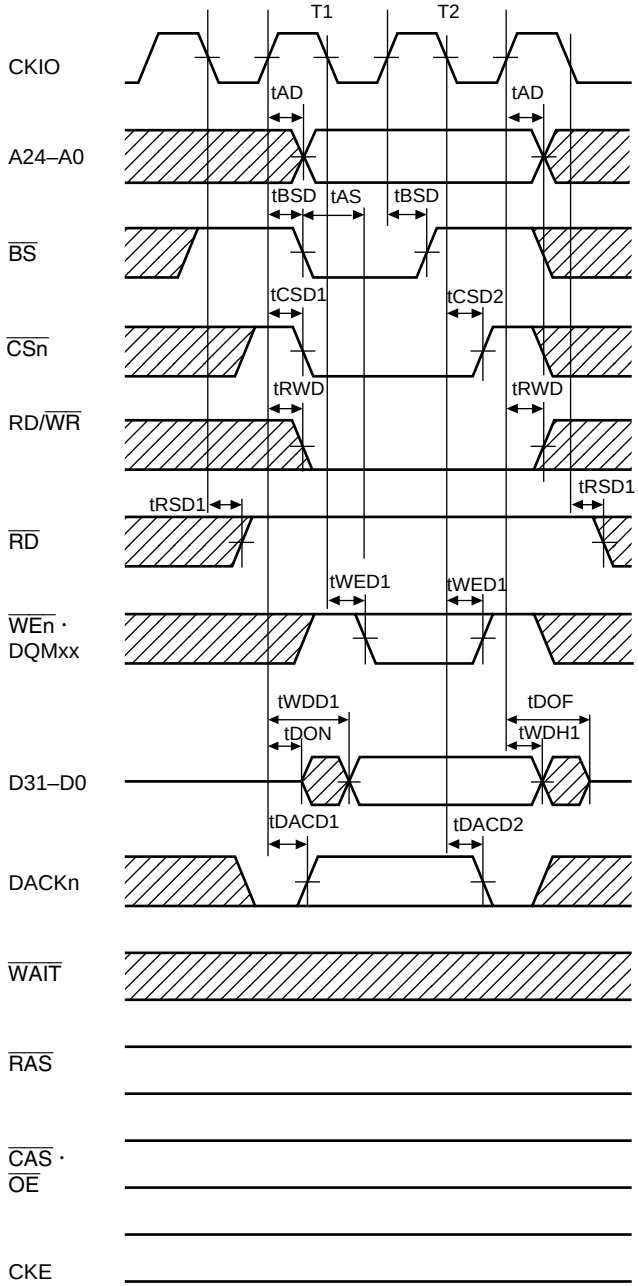
**Table 23.10 PLL-Off Bus Timing [Mode 6] (cont)**

| | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{WAIT}}$ setup time | tWTS | 10 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{WAIT}}$ hold time | tWTH | 18 | — | ns | 23.13, 14, 35, 36, 42 to 45 |
| $\overline{\text{RAS}}$ delay time 1 (SDRAM) | tRASD1 | — | 27 | ns | 23.15 to 18, 20 to 25, 28 to 32 |
| $\overline{\text{RAS}}$ delay time 2 (DRAM, EDO) | tRASD2 | — | 27 | ns | 23.33, 34, 39, 41 |
| $\overline{\text{RAS}}$ delay time 3 (EDO) | tRASD3 | — | 27 | ns | 23.39 |
| $\overline{\text{CAS}}$ delay time 1 (SDRAM) | tCASD1 | — | 27 | ns | 23.15, 16, 17, 18, 22 to 28, 30 to 32, 42 |
| $\overline{\text{CAS}}$ delay time 2 (DRAM) | tCASD2 | — | 27 | ns | 23.33, 34, 37 to 41 |
| DQM delay time | tDQMD | — | 27 | ns | 23.15, 16, 18 to 20, 22, 24 to 29 |
| CKE delay time | tCKED | — | 27 | ns | 23.32 |
| $\overline{\text{OE}}$ delay time 1 | tOED1 | — | 27 | ns | 23.39 |
| $\overline{\text{OE}}$ delay time 2 | tOED2 | — | 27 | ns | 23.39 |
| $\overline{\text{IVECF}}$ delay time | tIVD | — | 28 | ns | 23.43, 44 |
| Row address setup time | tASR | 0 | — | ns | 23.33, 34, 39 |
| Column address setup time | tASC | 0 | — | ns | 23.33, 34, 37, 38, 39 |
| Data input setup time | tDS | 0 | — | ns | 23.34, 38 |
| Read/write address setup time | tAS | 0 | — | ns | 23.11, 12 |
| REFOUT delay time | tREFOD | — | 28 | ns | 23.46 |



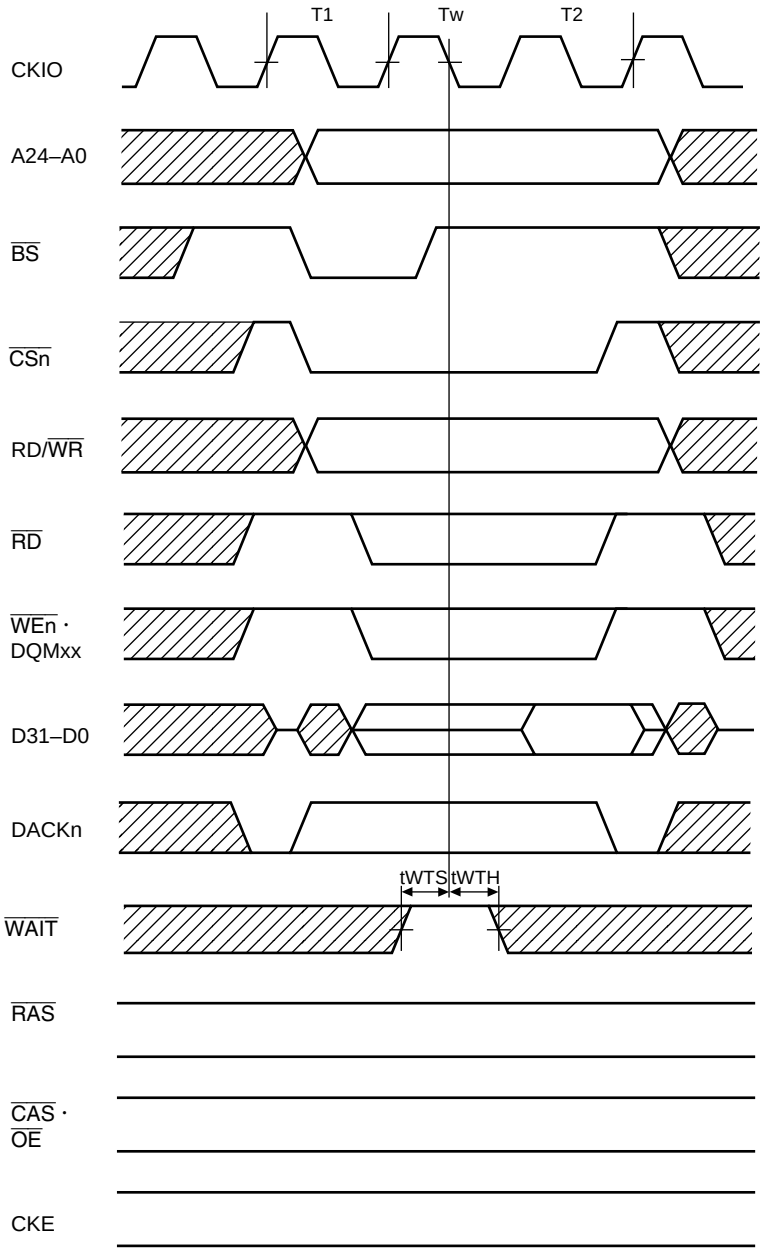

Notes: 1. $t_{RDH2}$ is measured from the rise of $\overline{CSn}$ or $\overline{RD}$, whichever comes first.
2. DACKn waveform when active-high is specified

**Figure 23.11 Basic Read Cycle (No Wait)**



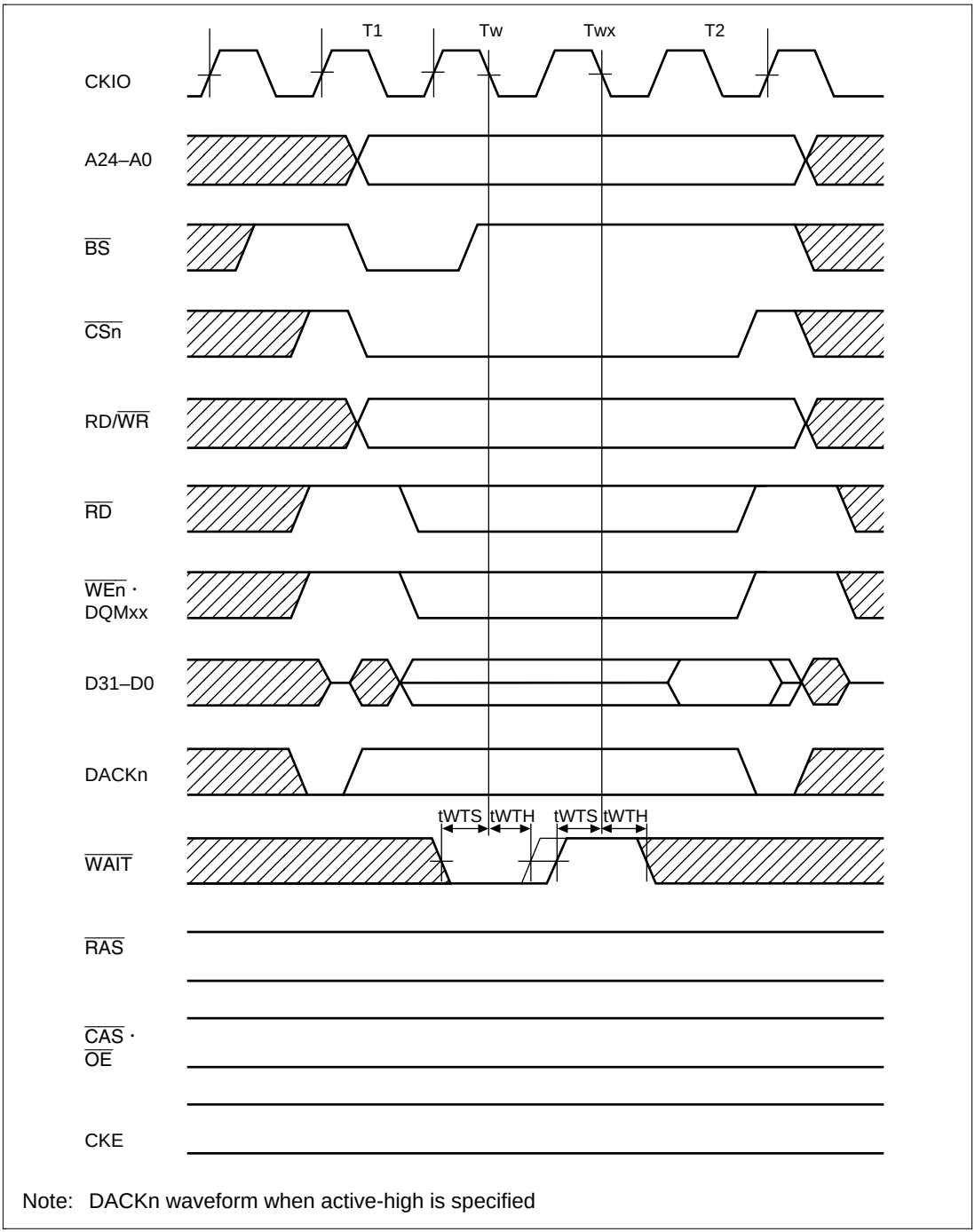

Note: DACKn waveform when active-high is specified

**Figure 23.12 Basic Write Cycle (No Wait)**



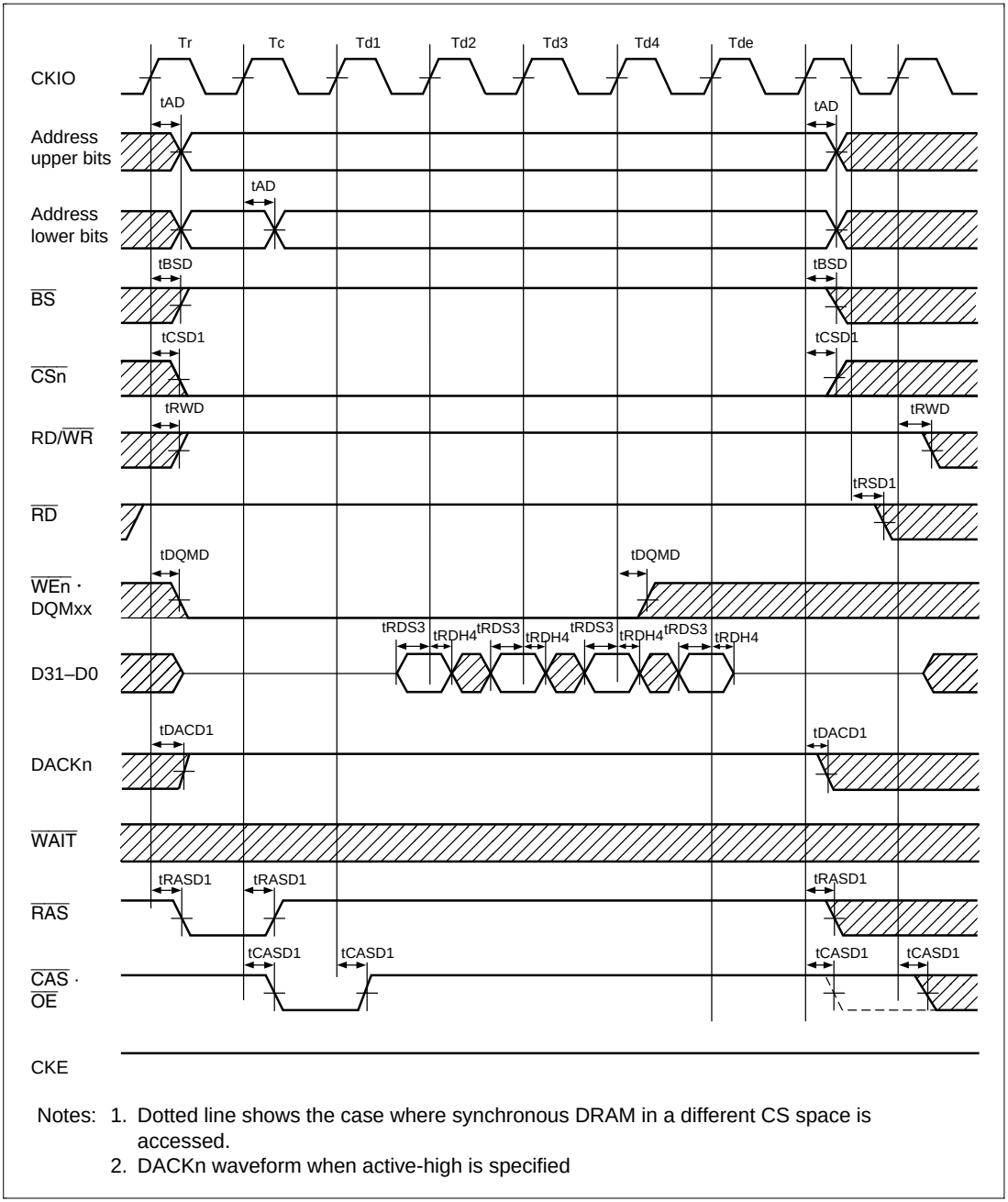

Note: DACKn waveform when active-high is specified

**Figure 23.13 Basic Bus Cycle (1 Wait Cycle)**



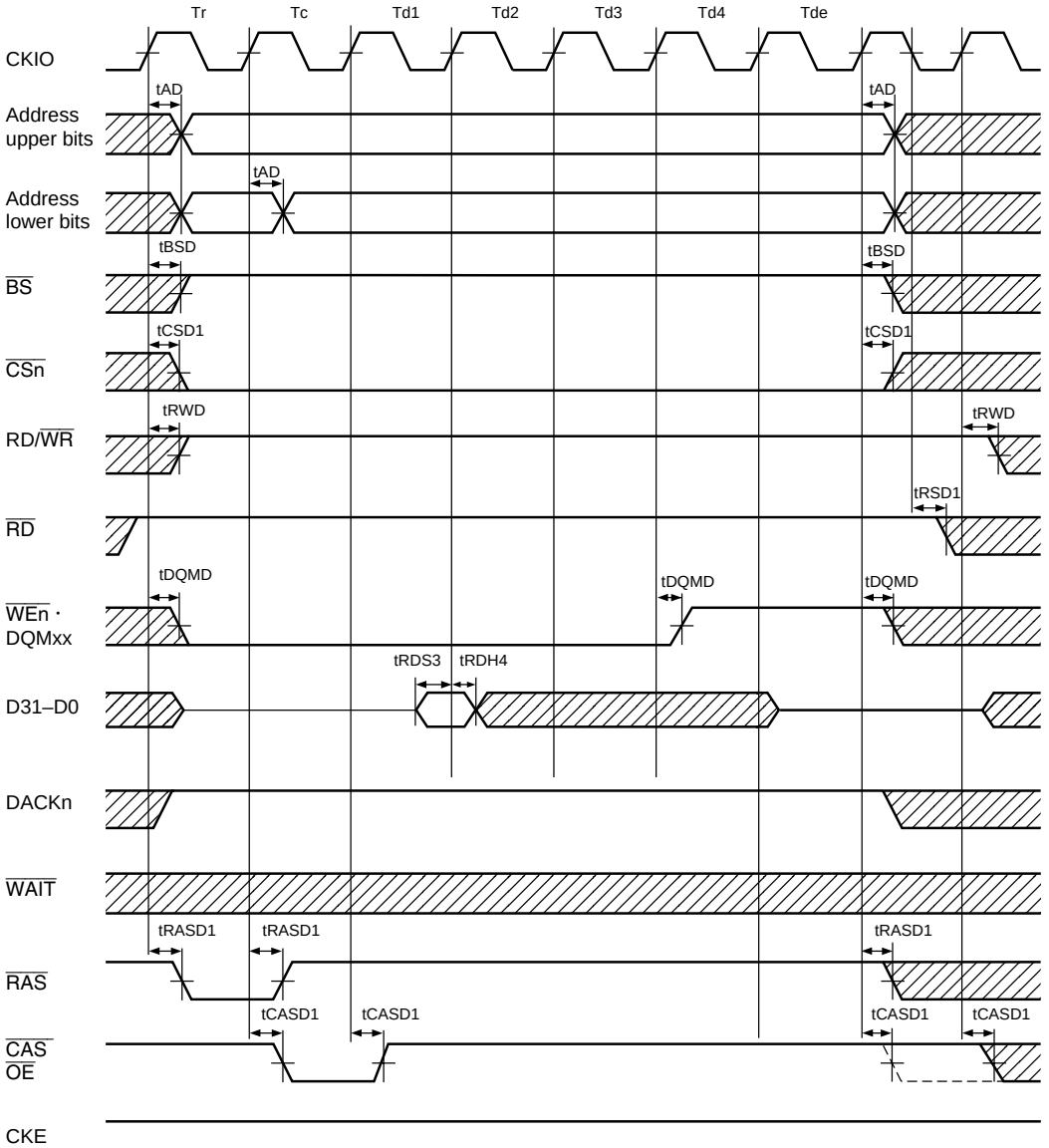

Note: DACKn waveform when active-high is specified

**Figure 23.14 Basic Bus Cycle (External Wait Input)**



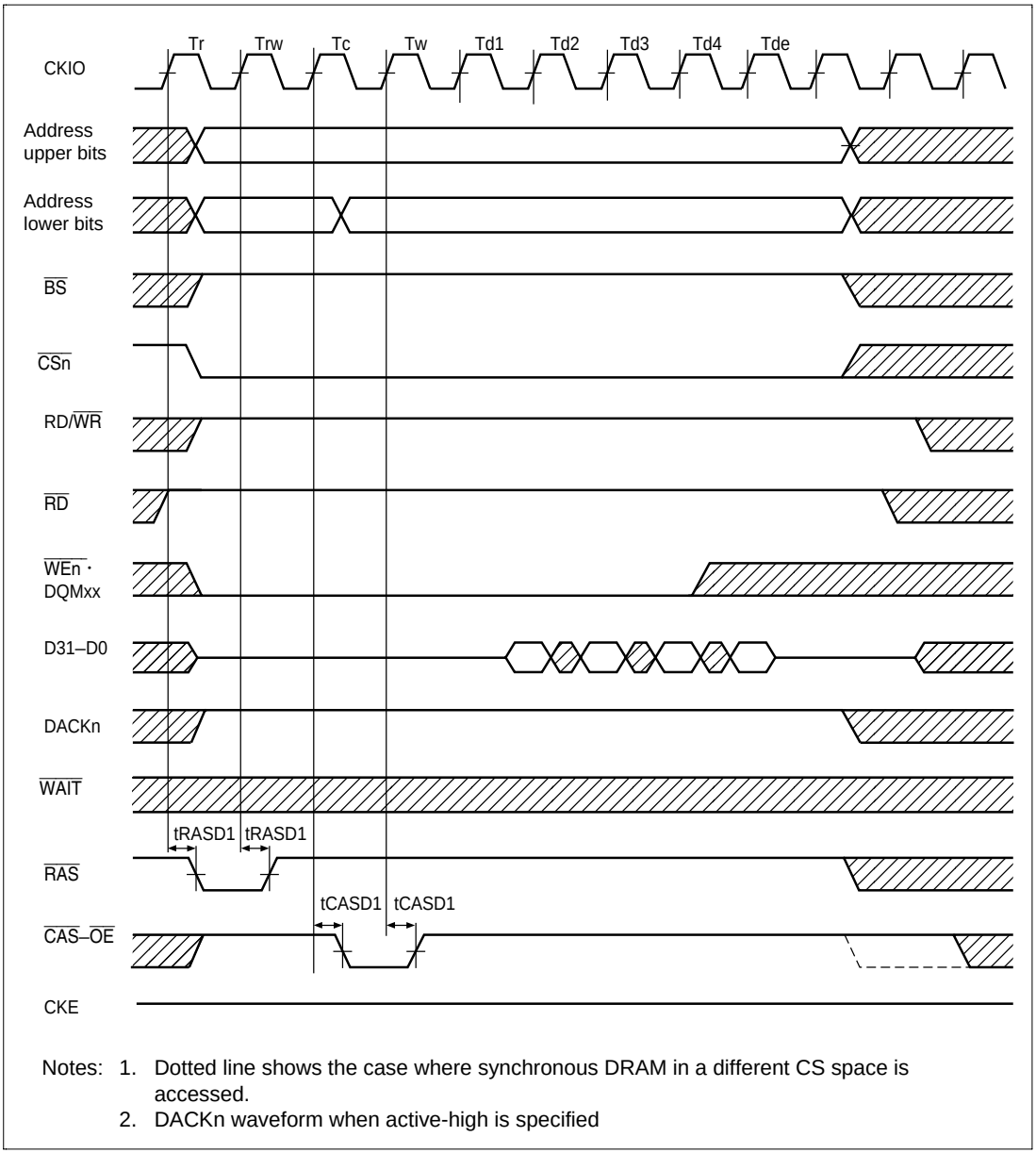

Notes: 1. Dotted line shows the case where synchronous DRAM in a different CS space is accessed.
2. DACKn waveform when active-high is specified

**Figure 23.15 Synchronous DRAM Read Bus Cycle (RCD = 1 Cycle, CAS Latency = 1 Cycle, Burst = 4)**




Notes: 1. Dotted line shows the case where synchronous DRAM in a different CS space is accessed.
2. DACKn waveform when active-high is specified

**Figure 23.16 Synchronous DRAM Single Read Bus Cycle (RCD = 1 Cycle, CAS Latency = 1 Cycle, Burst = 4)**




Notes: 1. Dotted line shows the case where synchronous DRAM in a different CS space is accessed.
2. DACKn waveform when active-high is specified

**Figure 23.17 Synchronous DRAM Read Bus Cycle (RCD = 2 Cycles, CAS Latency = 2 Cycles, Burst = 4)**

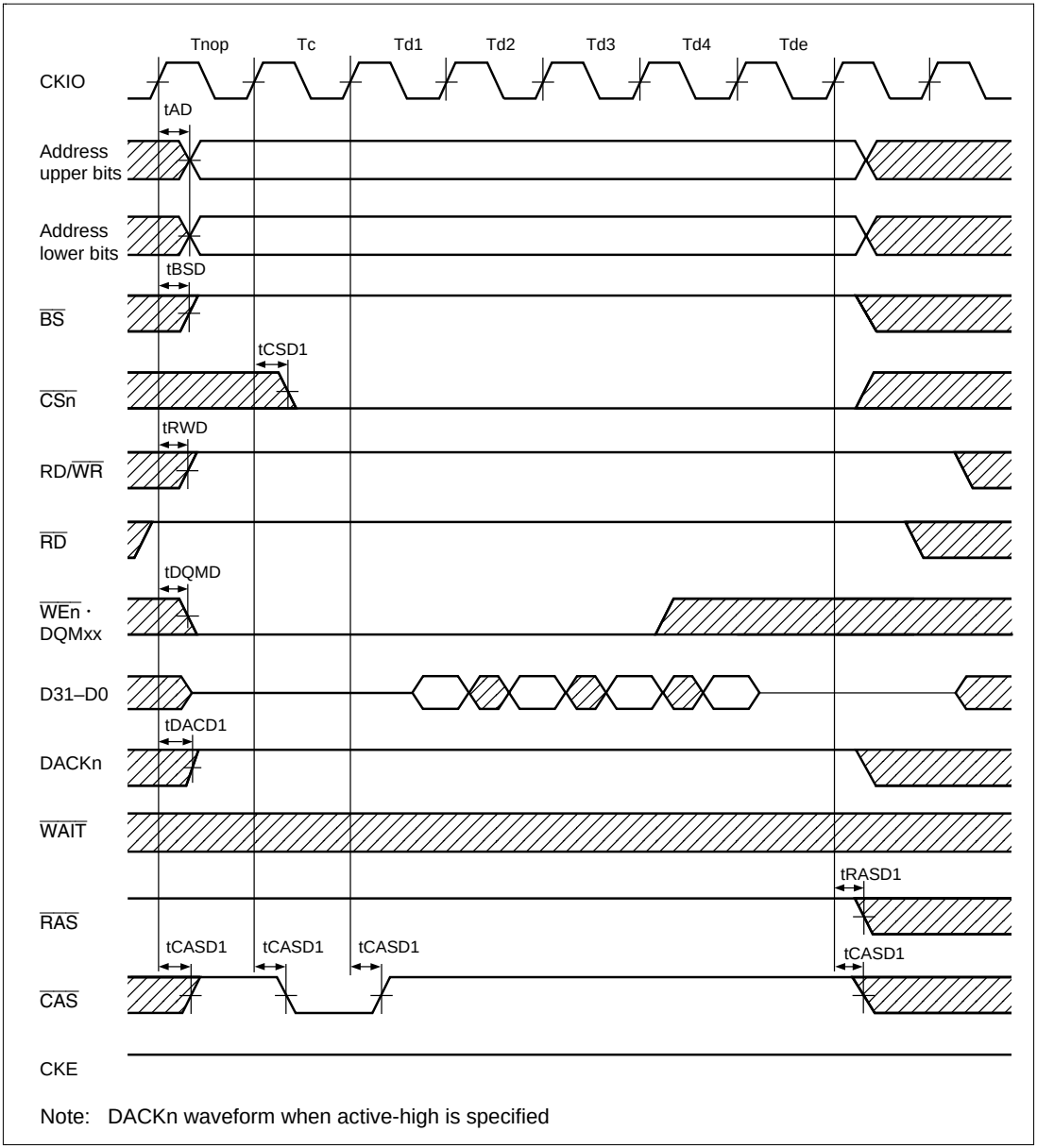



**Figure 23.18 Synchronous DRAM Read Bus Cycle (Bank Active, Same Row Access, CAS Latency = 1 Cycle)**

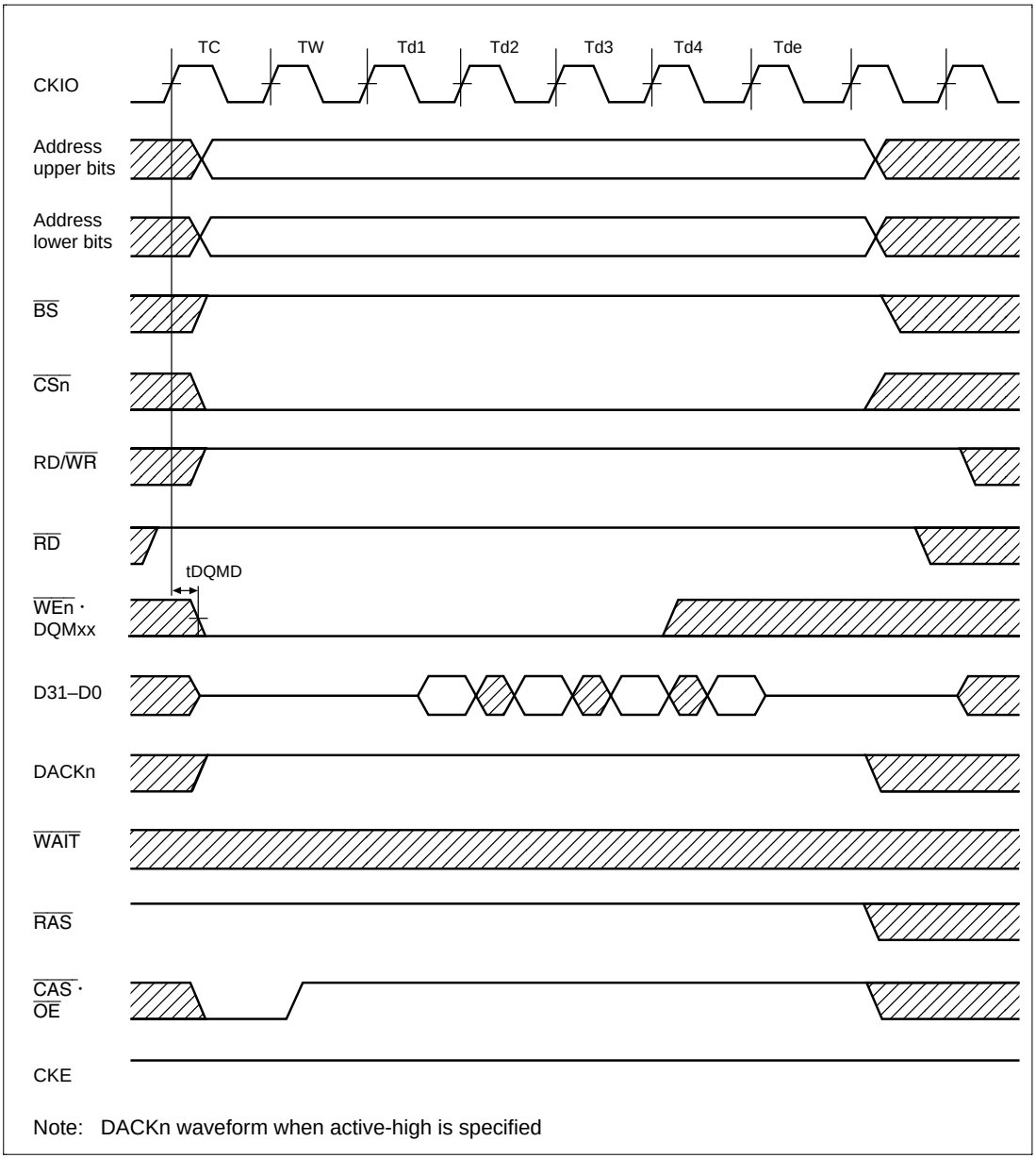



**Figure 23.19 Synchronous DRAM Read Bus Cycle (Bank Active, Same Row Access, CAS Latency = 2 Cycles)**

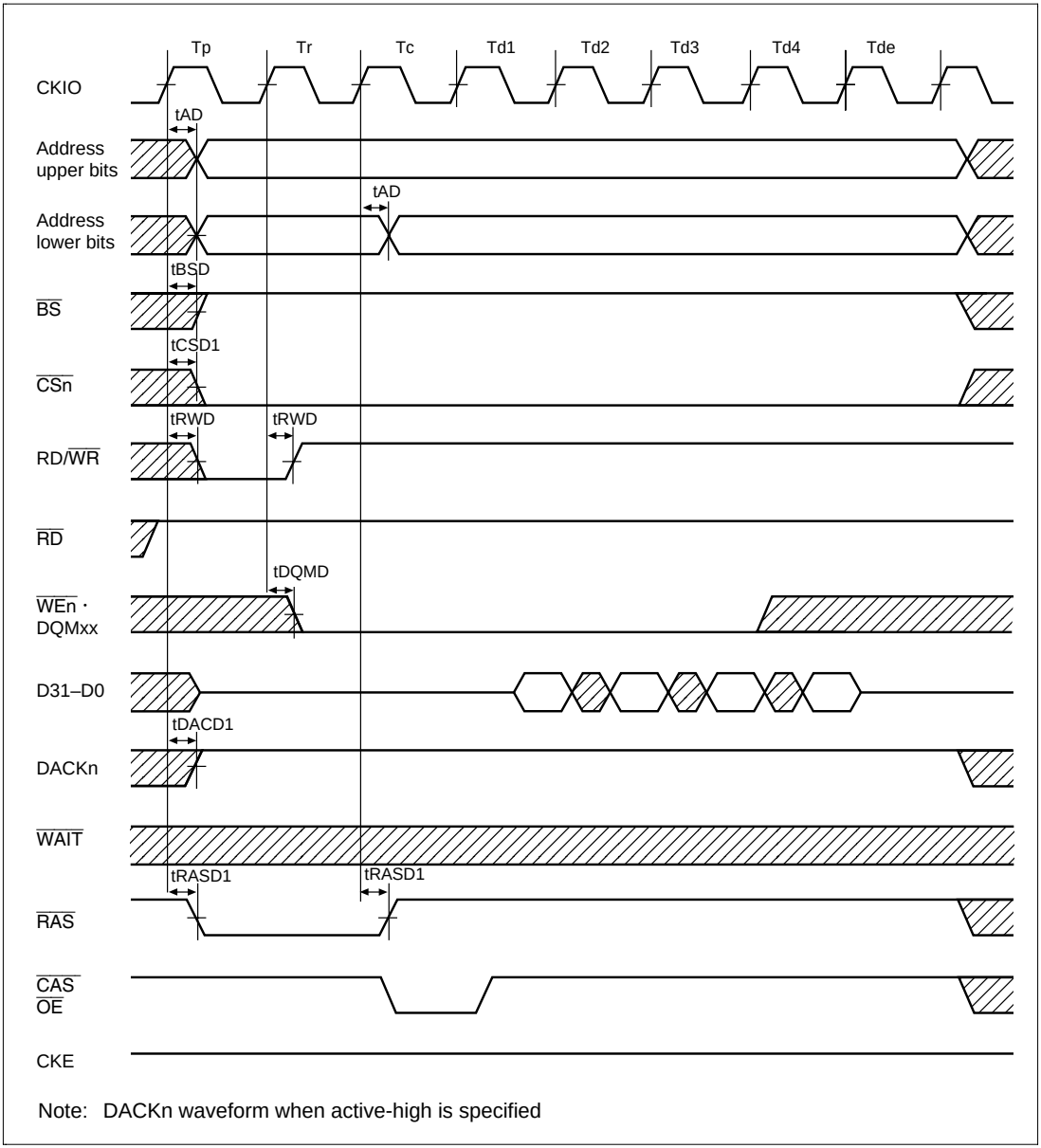



**Figure 23.20 Synchronous DRAM Read Bus Cycle**
**(Bank Active, Different Row Access, TRP = 1 Cycle, RCD = 1 Cycle, CAS Latency = 1 Cycle)**

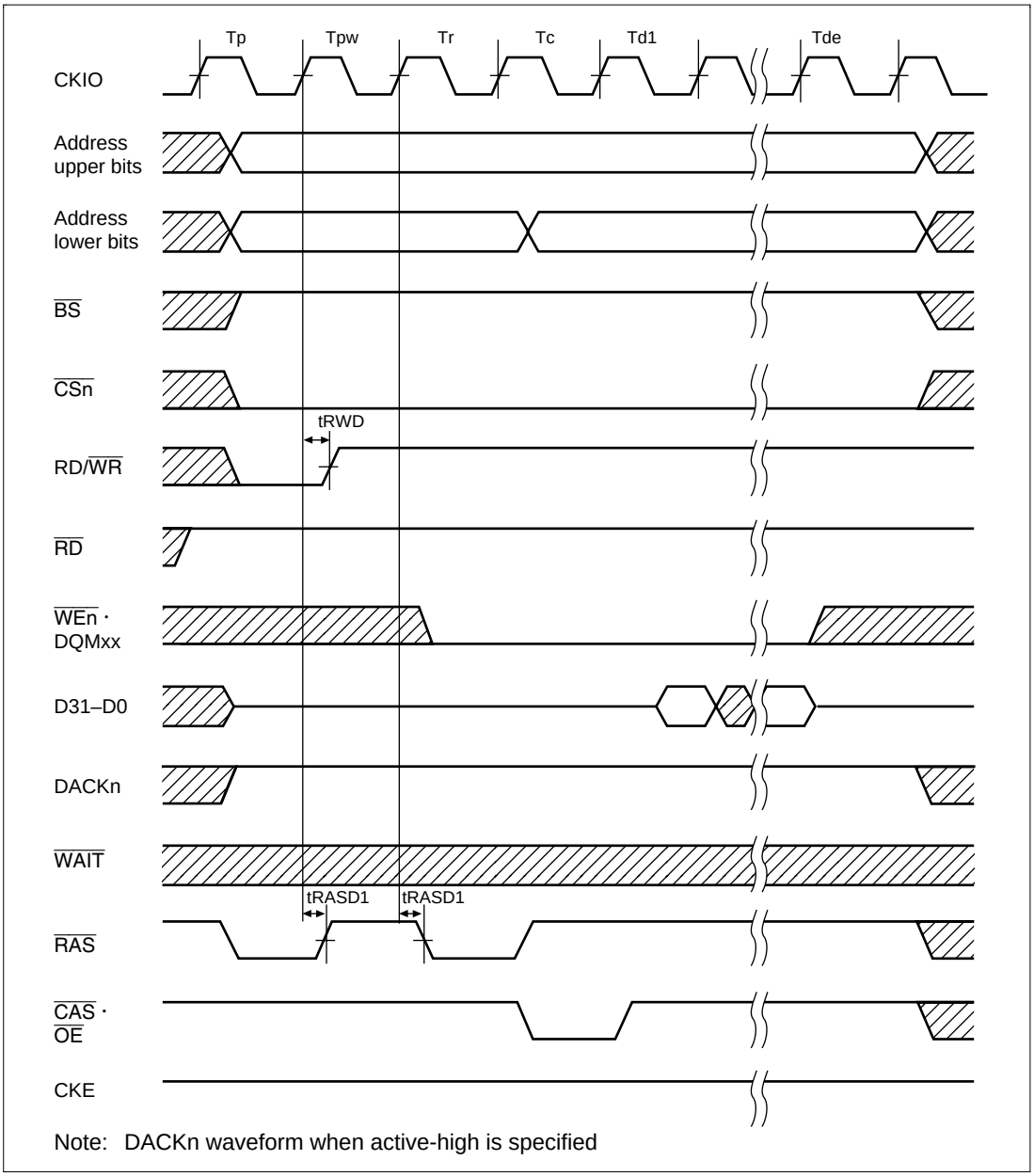
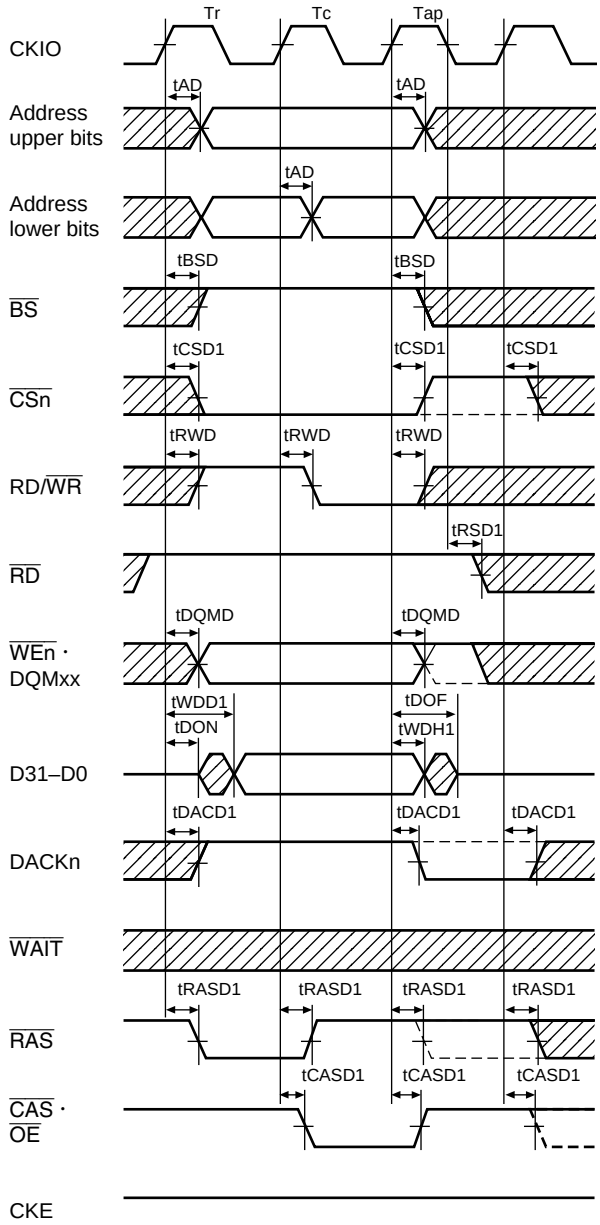
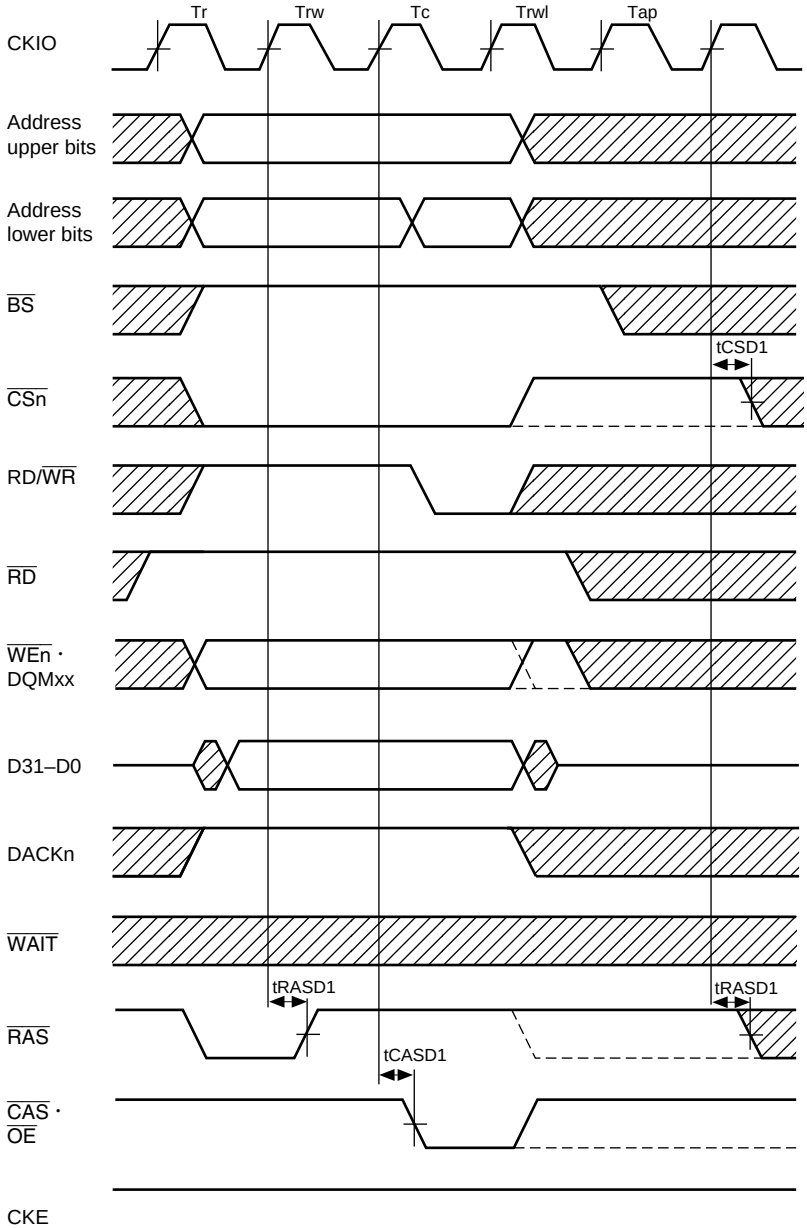



**Figure 23.21 Synchronous DRAM Read Bus Cycle (Bank Active, Different Row Access, TRP = 2 Cycles, RCD = 1 Cycle, CAS Latency = 1 Cycle)**



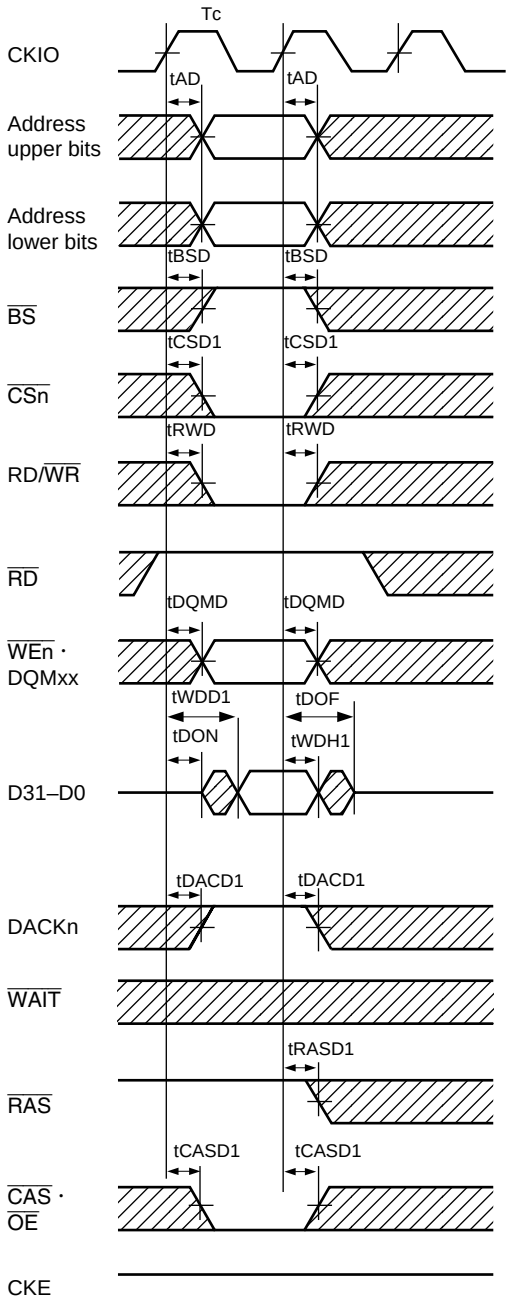

Notes: 1. Dotted line shows the case where synchronous DRAM in a different CS space is accessed.
2. DACKn waveform when active-high is specified

**Figure 23.22 Synchronous DRAM Write Bus Cycle (RASD = 0, RCD = 1 Cycle, TRWL = 1 Cycle)**



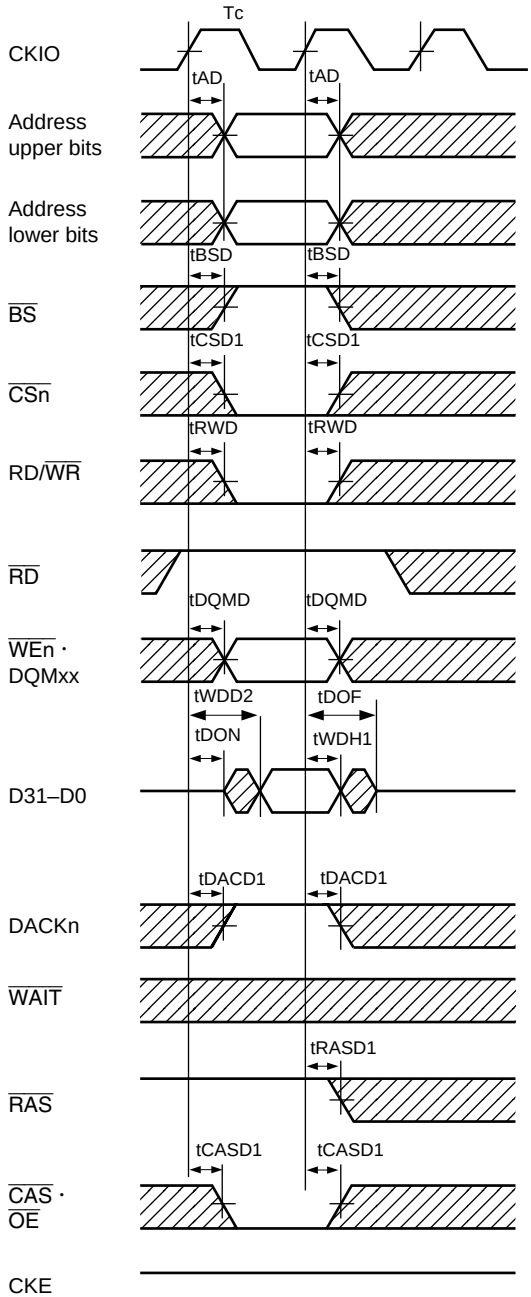

Notes: 1. Dotted line shows the case where synchronous DRAM in a different CS space is accessed.
2. DACKn waveform when active-high is specified

**Figure 23.23 Synchronous DRAM Write Bus Cycle (RASD = 0, RCD = 2 Cycles, TRWL = 2 Cycles)**



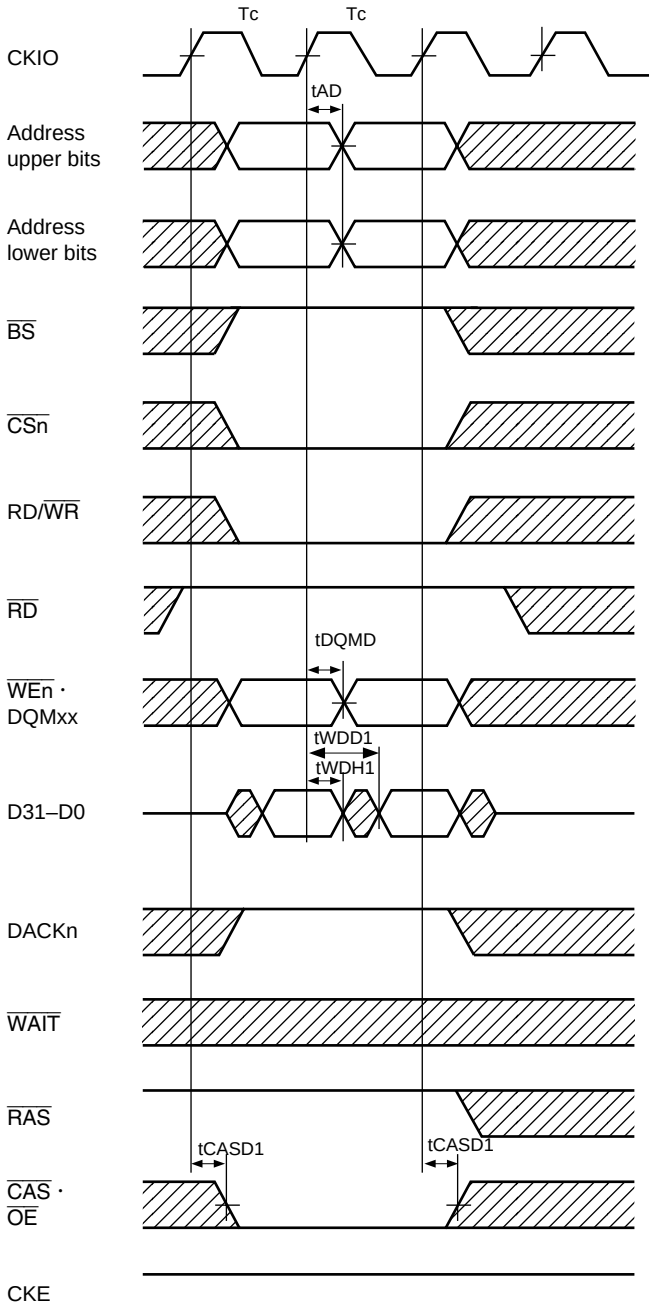

Note: DACKn waveform when active-high is specified

**Figure 23.24 Synchronous DRAM Write Bus Cycle (Bank Active, Same Row Access, Iϕ:Eϕ ≠ 1:1)**



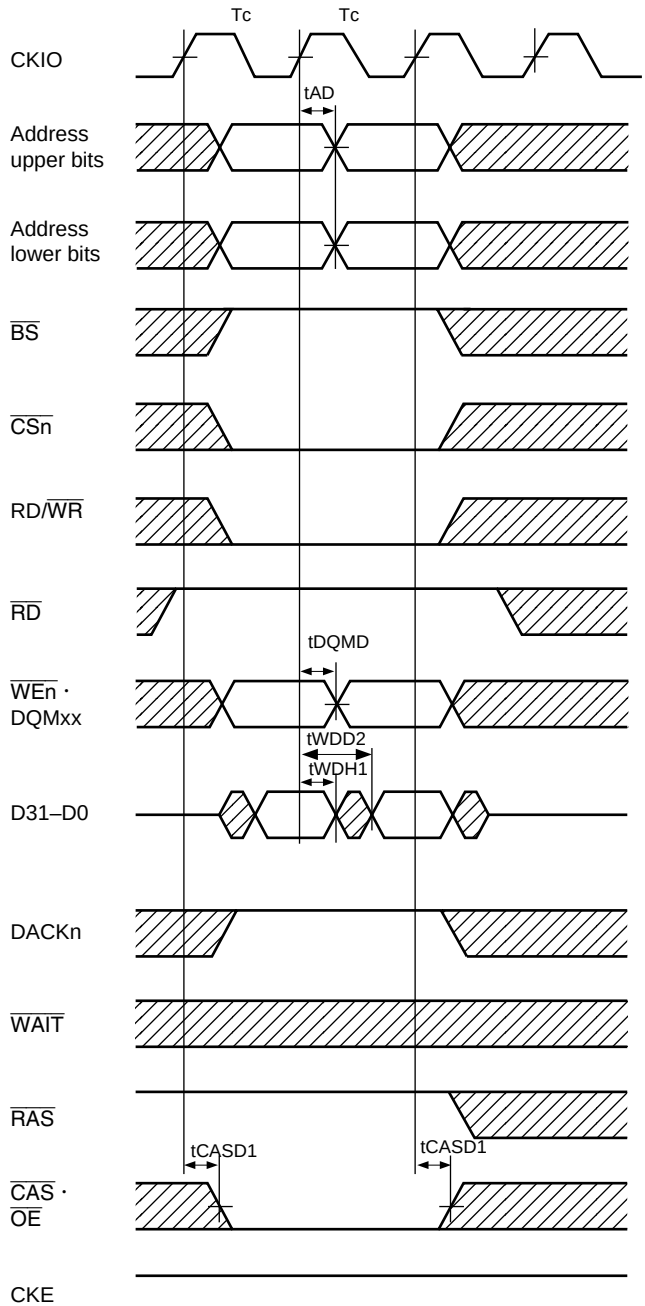

Note: DACKn waveform when active-high is specified

**Figure 23.25 Synchronous DRAM Write Cycle (Bank Active, Same Row Access, Iϕ:Eϕ = 1:1)**



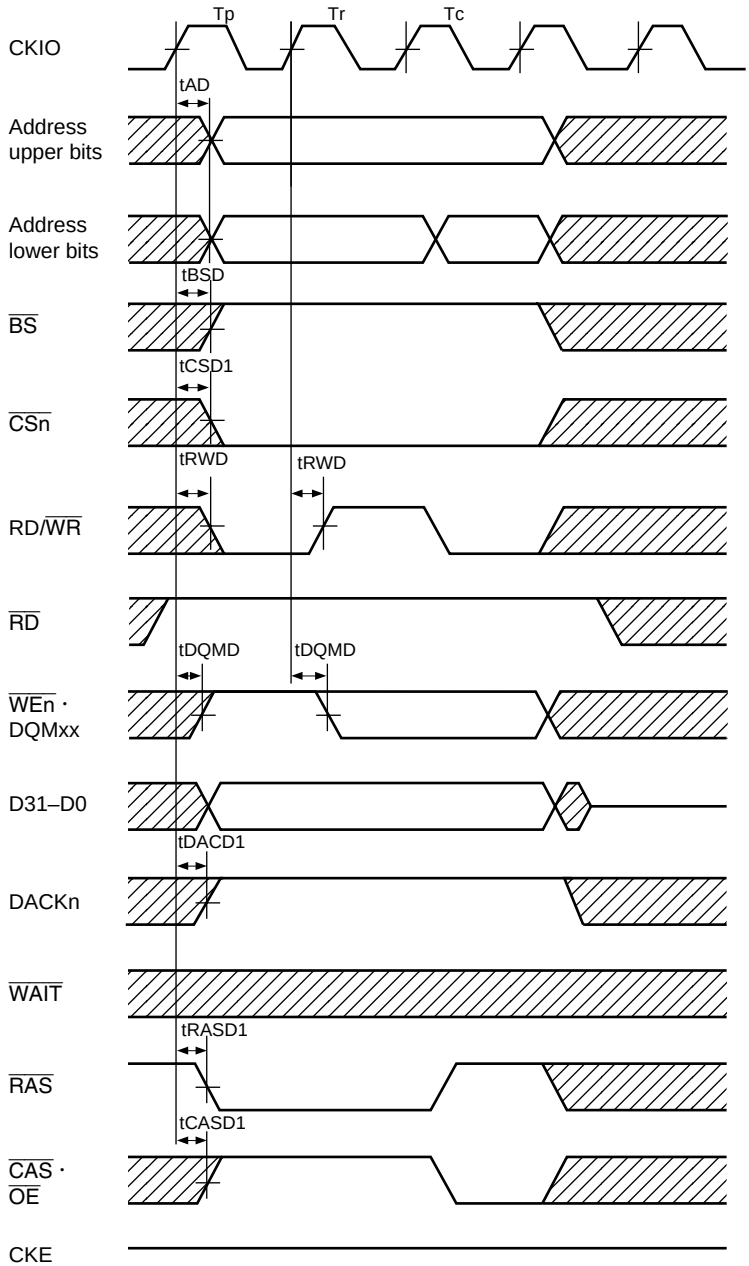

Note: DACKn waveform when active-high is specified

**Figure 23.26 Synchronous DRAM Continuous Write Cycle (Bank Active, Same Row Access, Iϕ:Eϕ ≠ 1:1)**



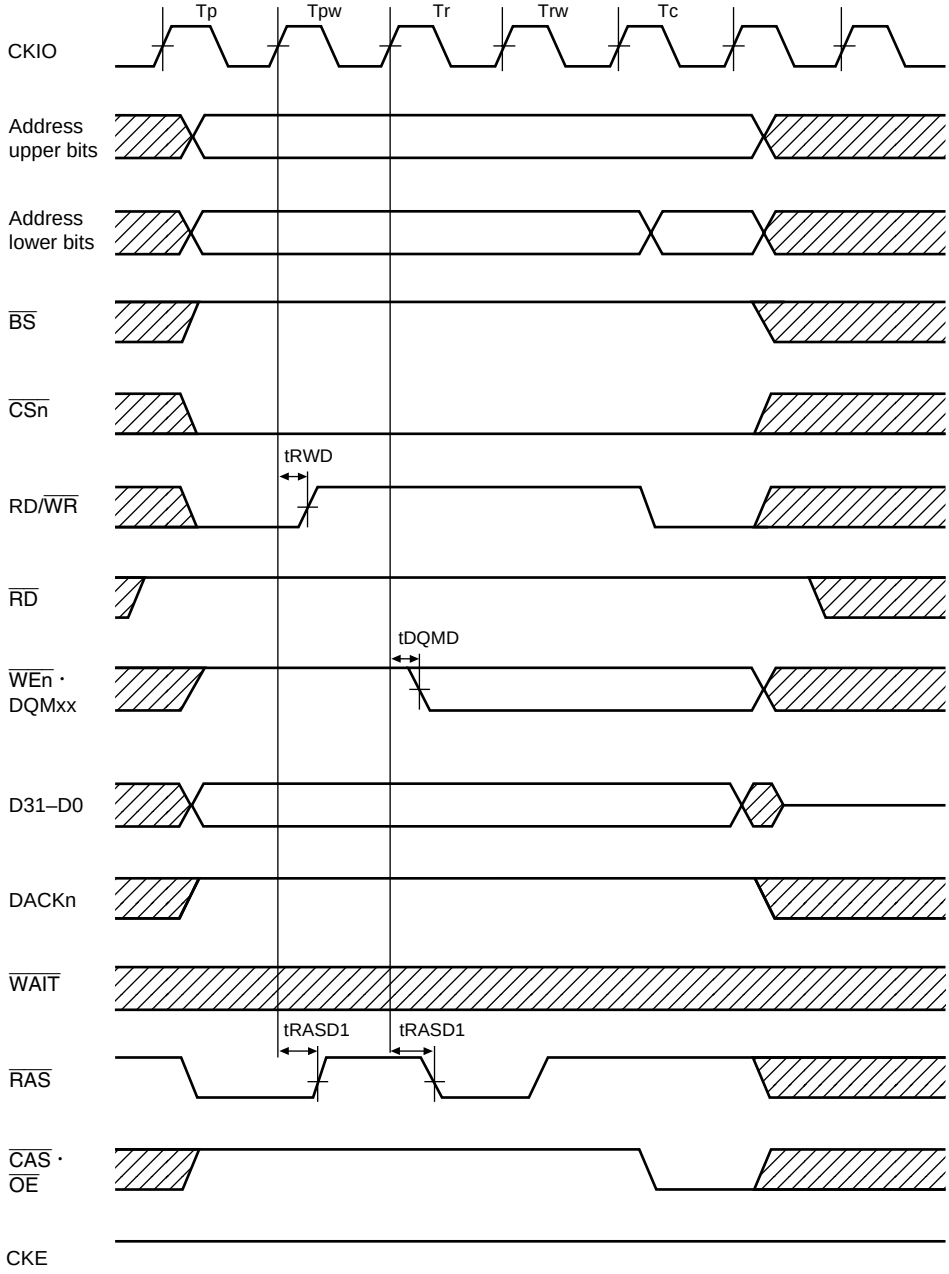

Note: DACKn waveform when active-high is specified

**Figure 23.27 Synchronous DRAM Continuous Write Cycle (Bank Active, Same Row Access, Iϕ:Eϕ = 1:1)**



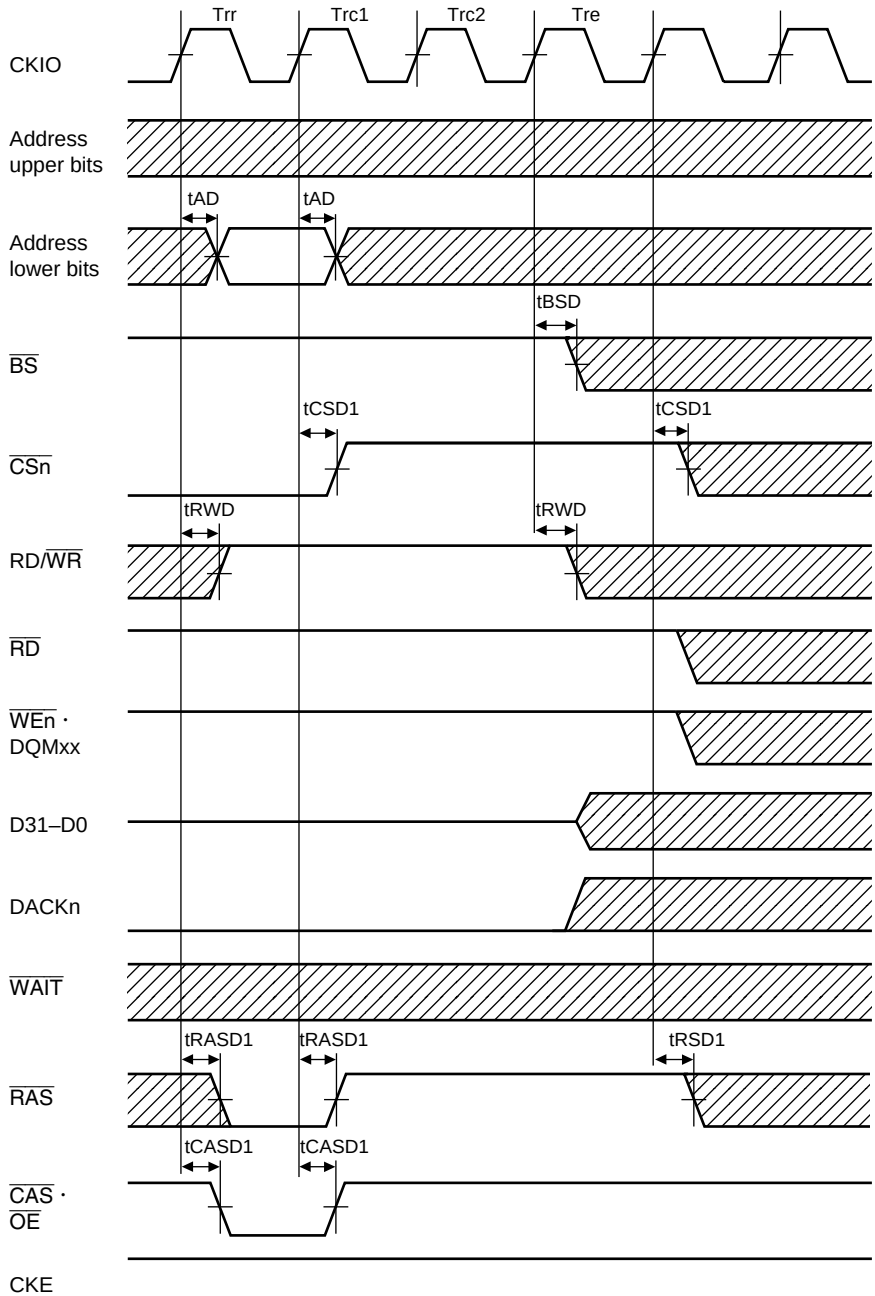

Note: DACKn waveform when active-high is specified

**Figure 23.28 Synchronous DRAM Write Bus Cycle (Bank Active, Different Row Access, TRP = 1 Cycle, RCD = 1 Cycle)**




Note: DACKn waveform when active-high is specified

**Figure 23.29 Synchronous DRAM Write Bus Cycle (Bank Active, Different Row Access, TRP = 2 Cycles, RCD = 2 Cycles)**




Note: An auto-refresh cycle is always preceded by a precharge cycle. The number of cycles between the two is determined by the number of cycles specified by TRP.

**Figure 23.30 Synchronous DRAM Auto-Refresh Cycle (TRAS = 4 Cycles)**

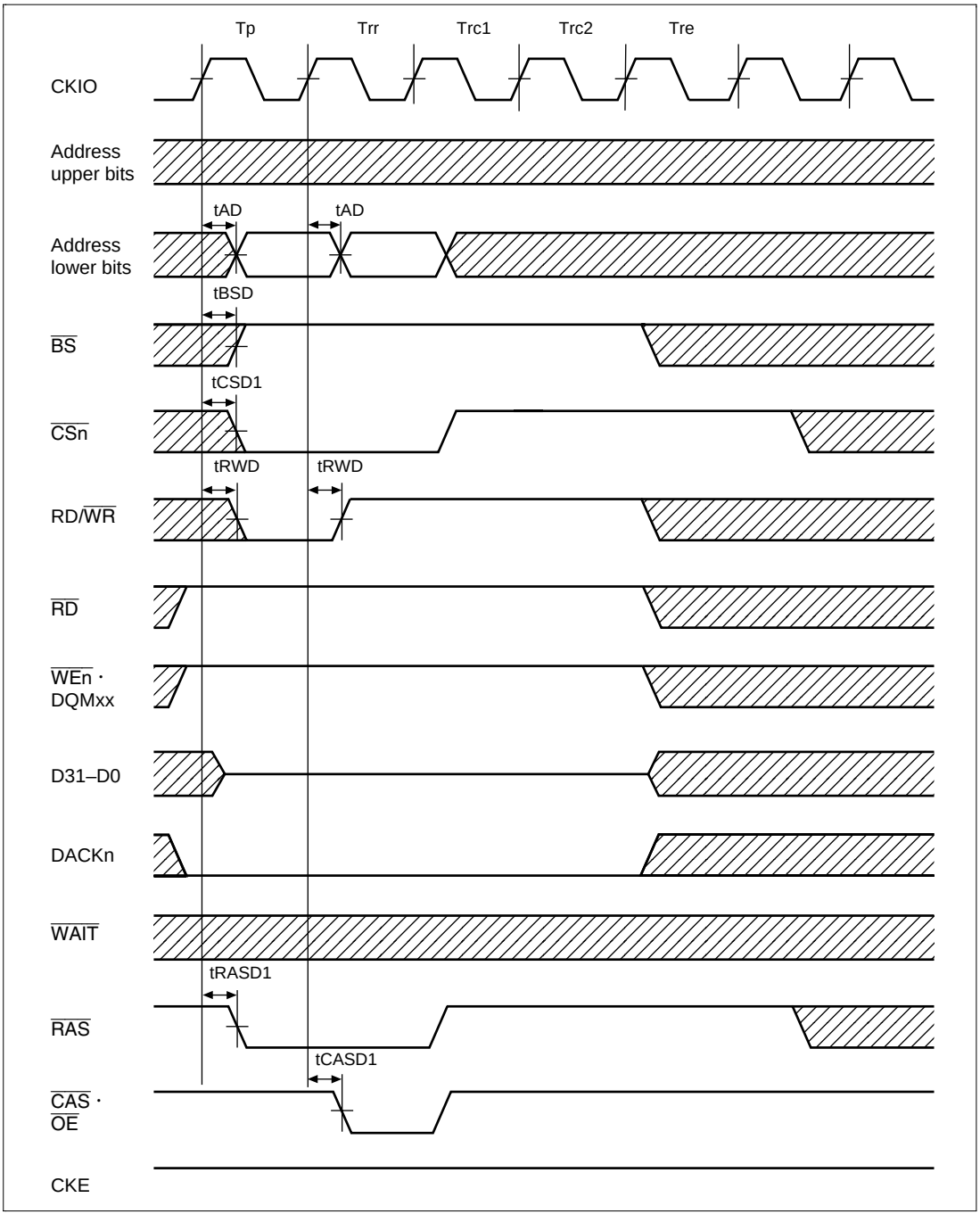
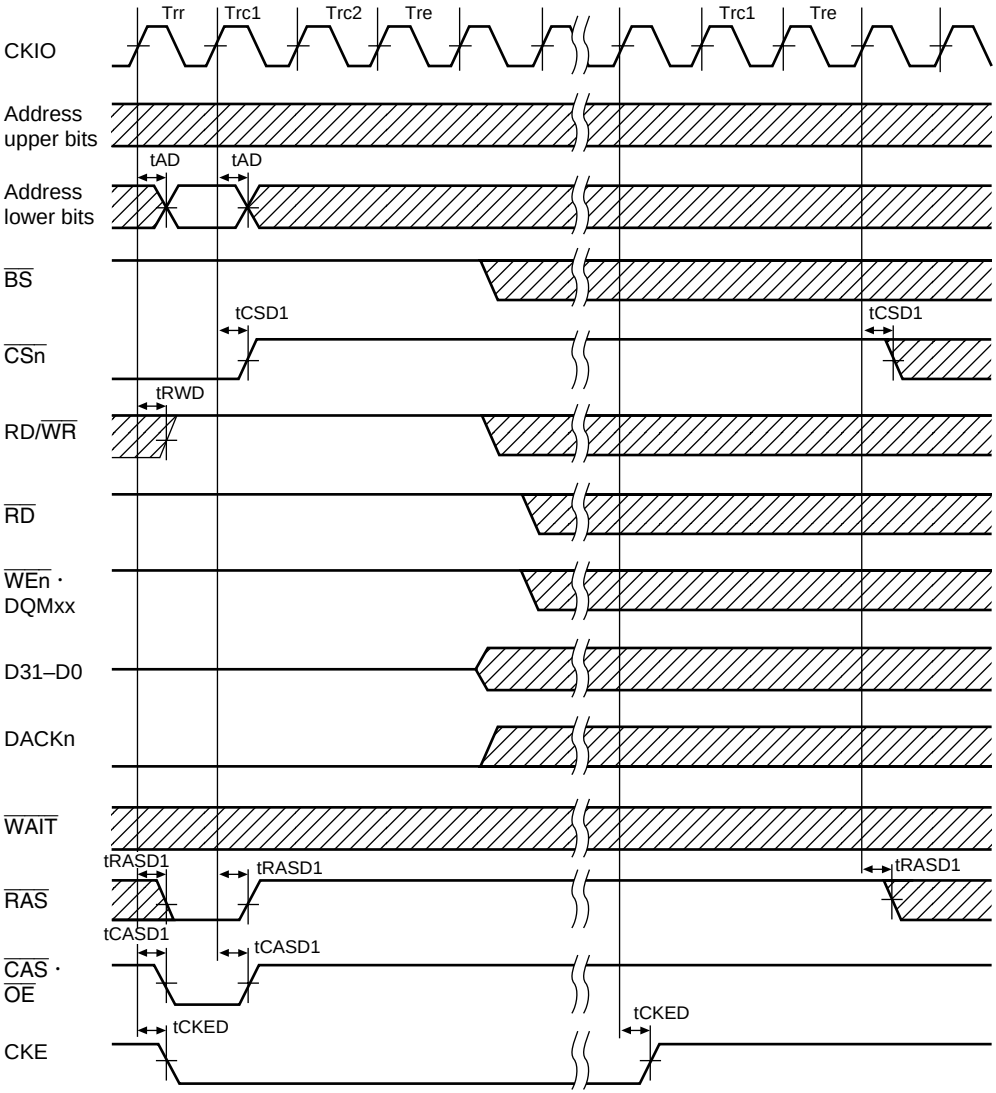
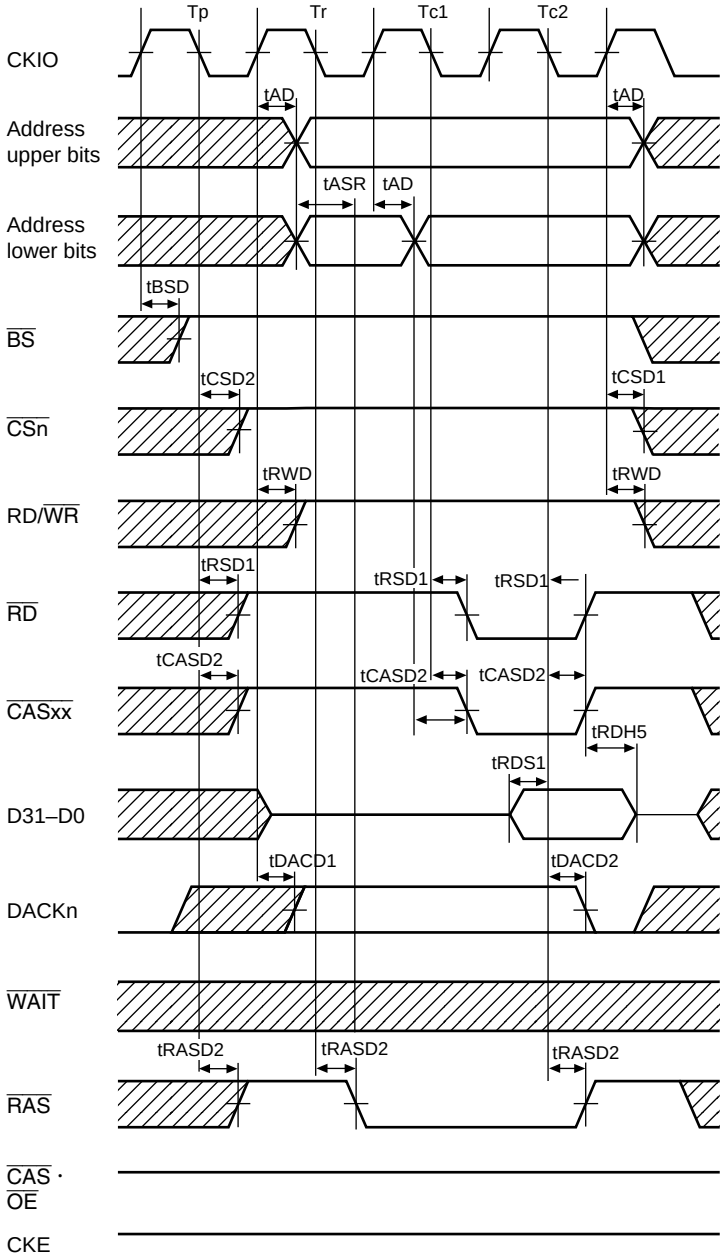



**Figure 23.31 Synchronous DRAM Auto-Refresh Cycle (Shown from Precharge Cycle, TRP = 1 Cycle, TRAS = 4 Cycles)**



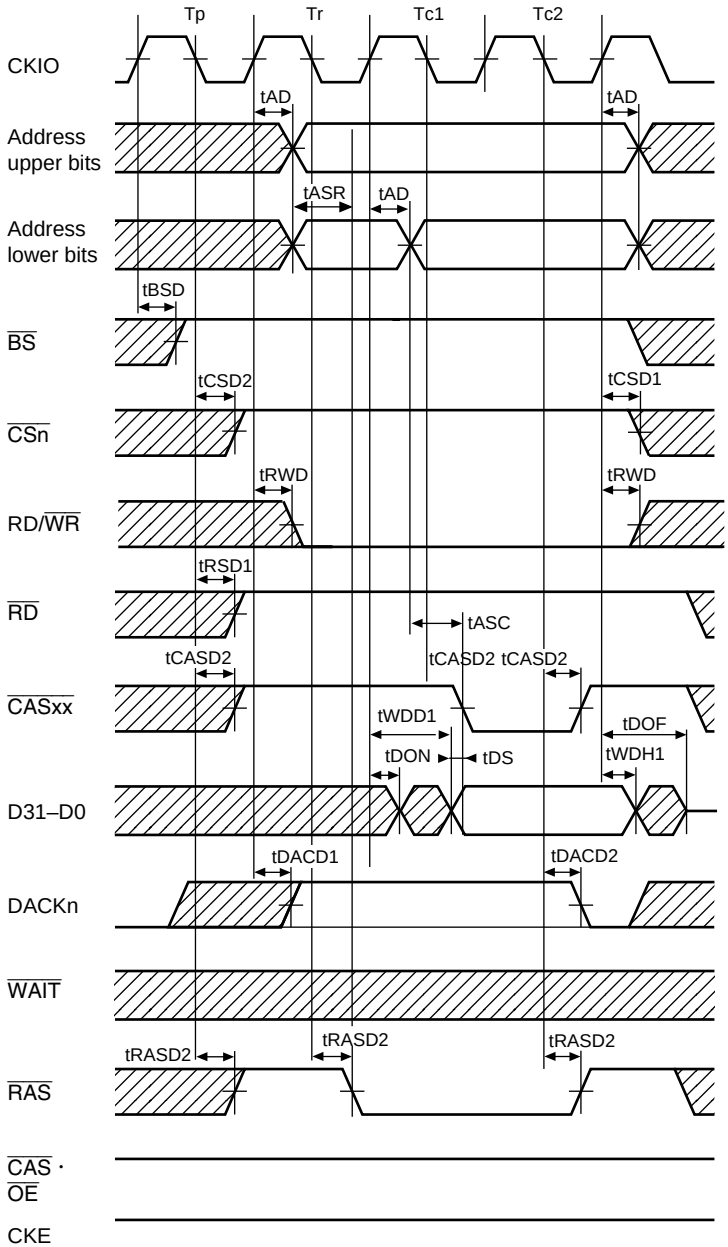

Note: A self-refresh cycle is always preceded by a precharge cycle. The number of cycles between the two is determined by the number of cycles specified by TRP.

**Figure 23.32 Synchronous DRAM Self-Refresh Cycle (TRAS = 3)**




Notes: 1. $t_{RDH5}$ is measured from the rise of $\overline{RD}$ or $\overline{CASxx}$, whichever comes first.
2. DACKn waveform when active-high is specified

**Figure 23.33 DRAM Read Cycle (TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**




Note: DACKn waveform when active-high is specified

**Figure 23.34 DRAM Write Cycle (TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**

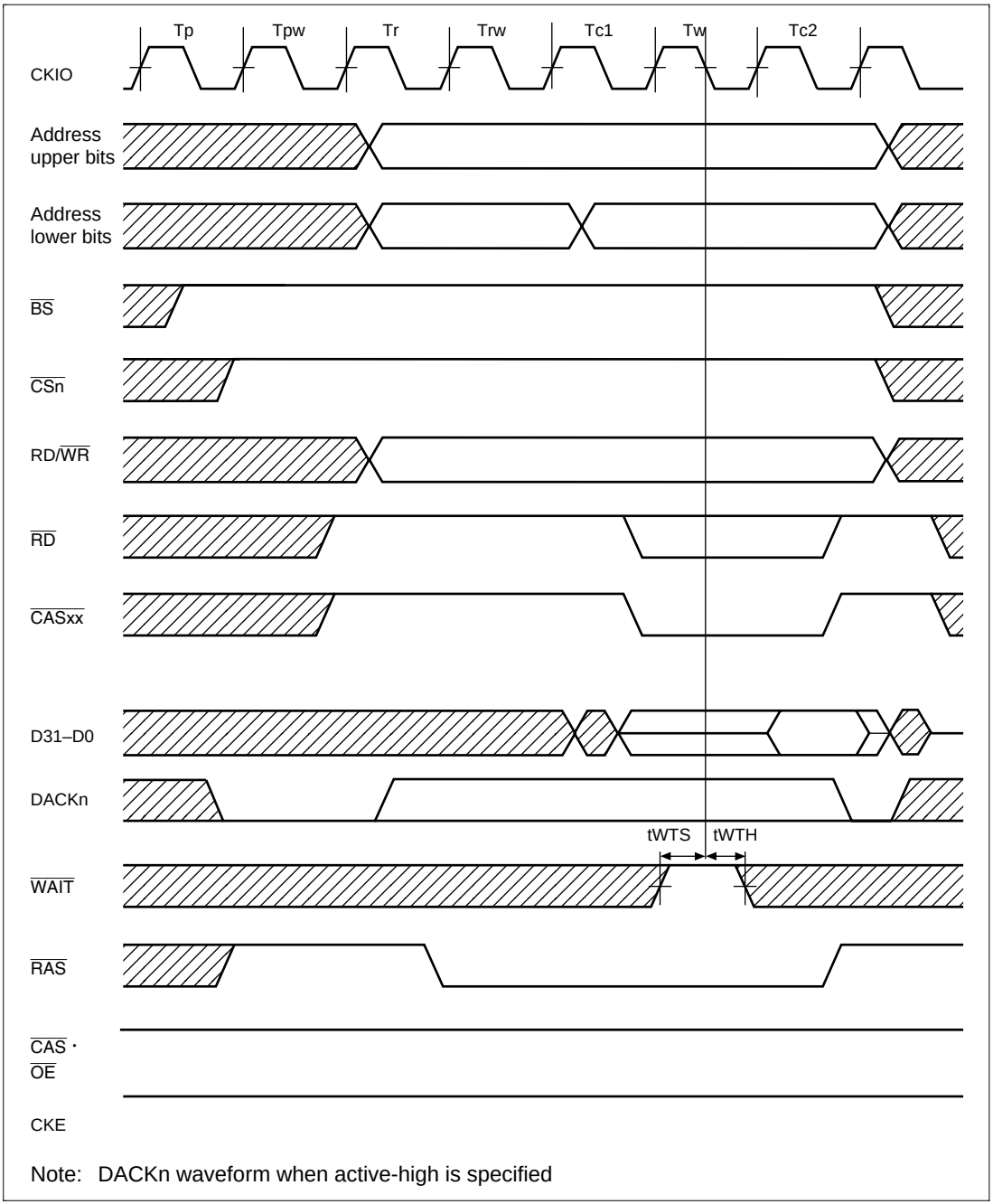
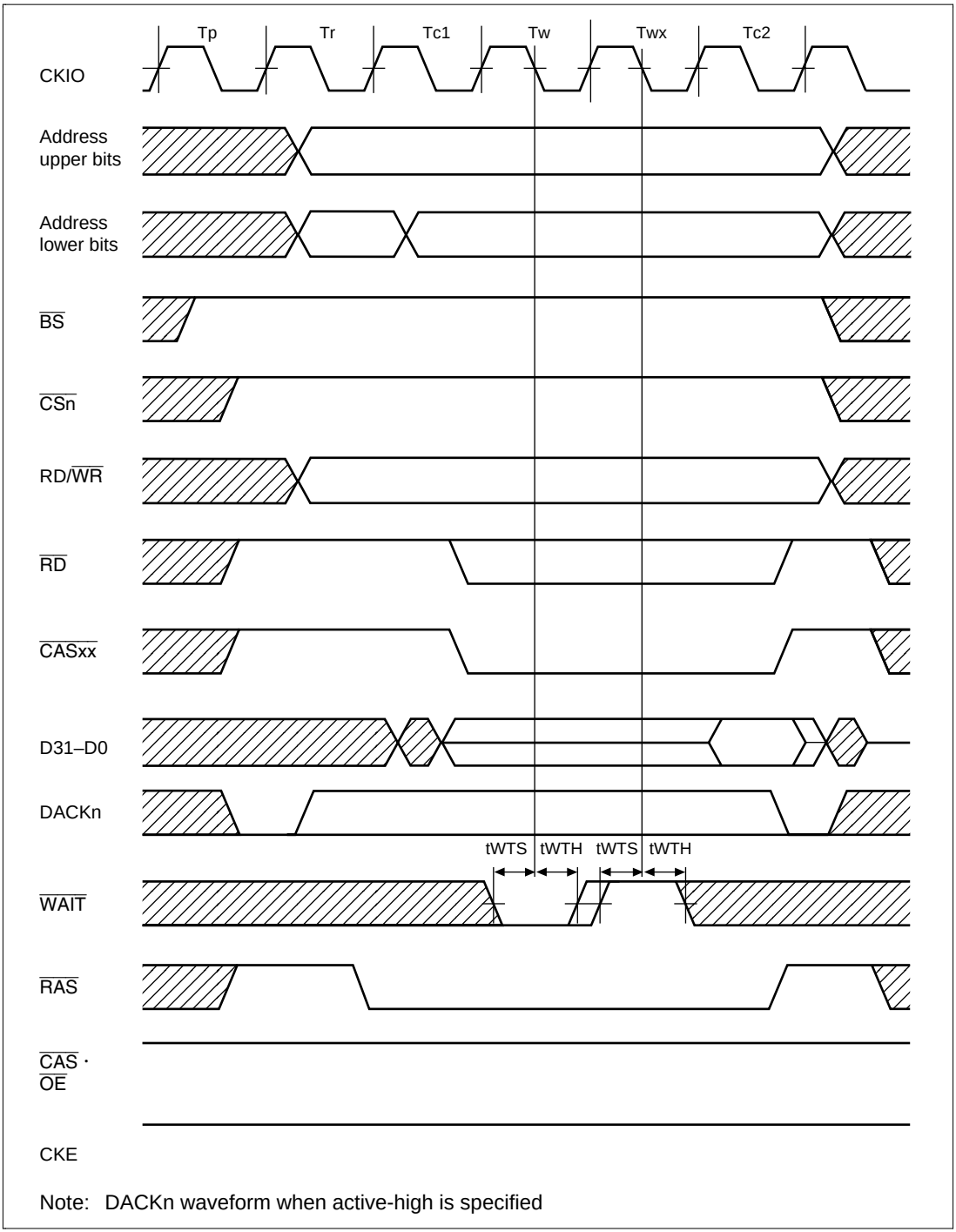
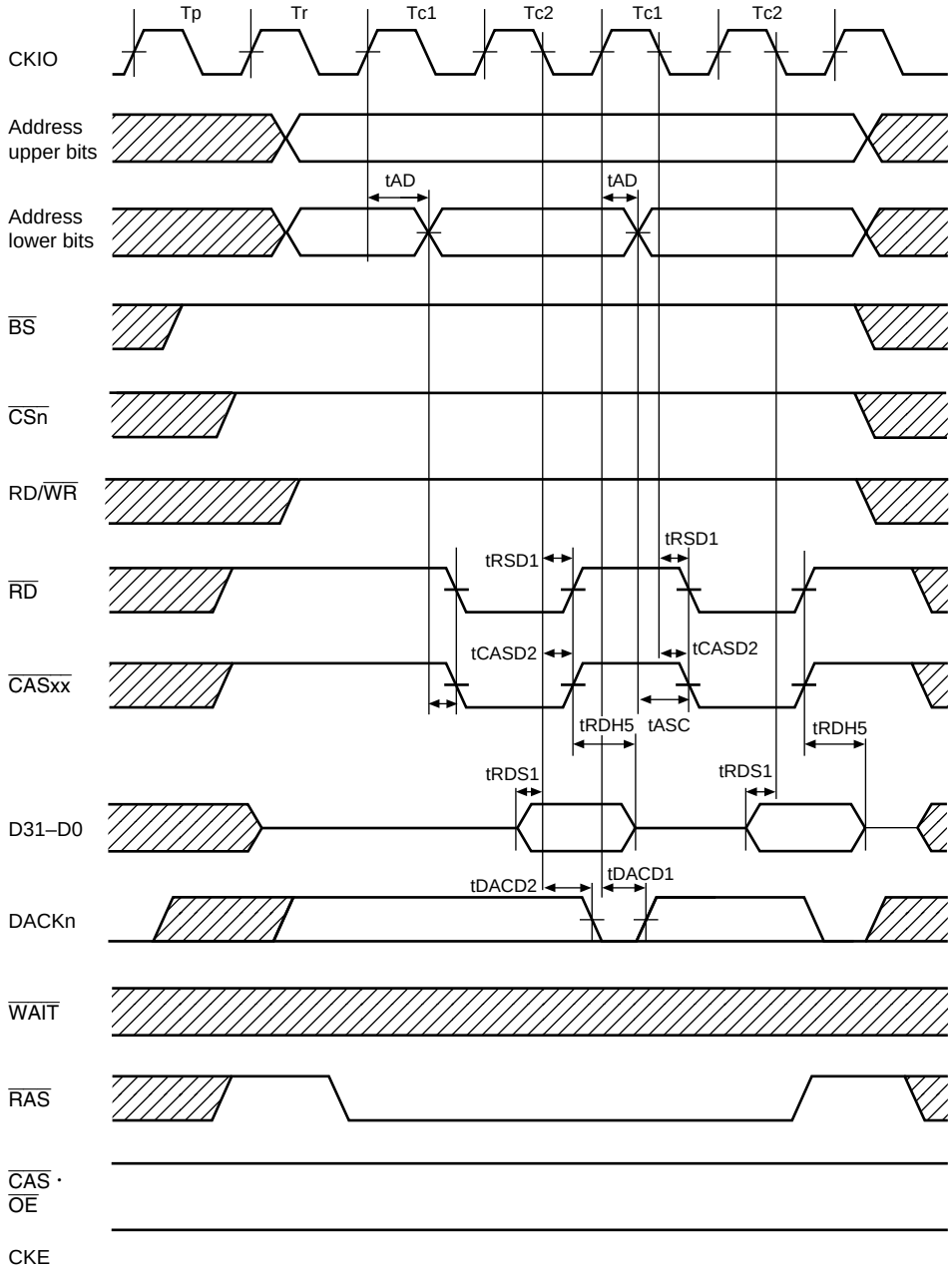



**Figure 23.35 DRAM Bus Cycle (TRP = 2 Cycles, RCD = 2 Cycles, 1 Wait)**



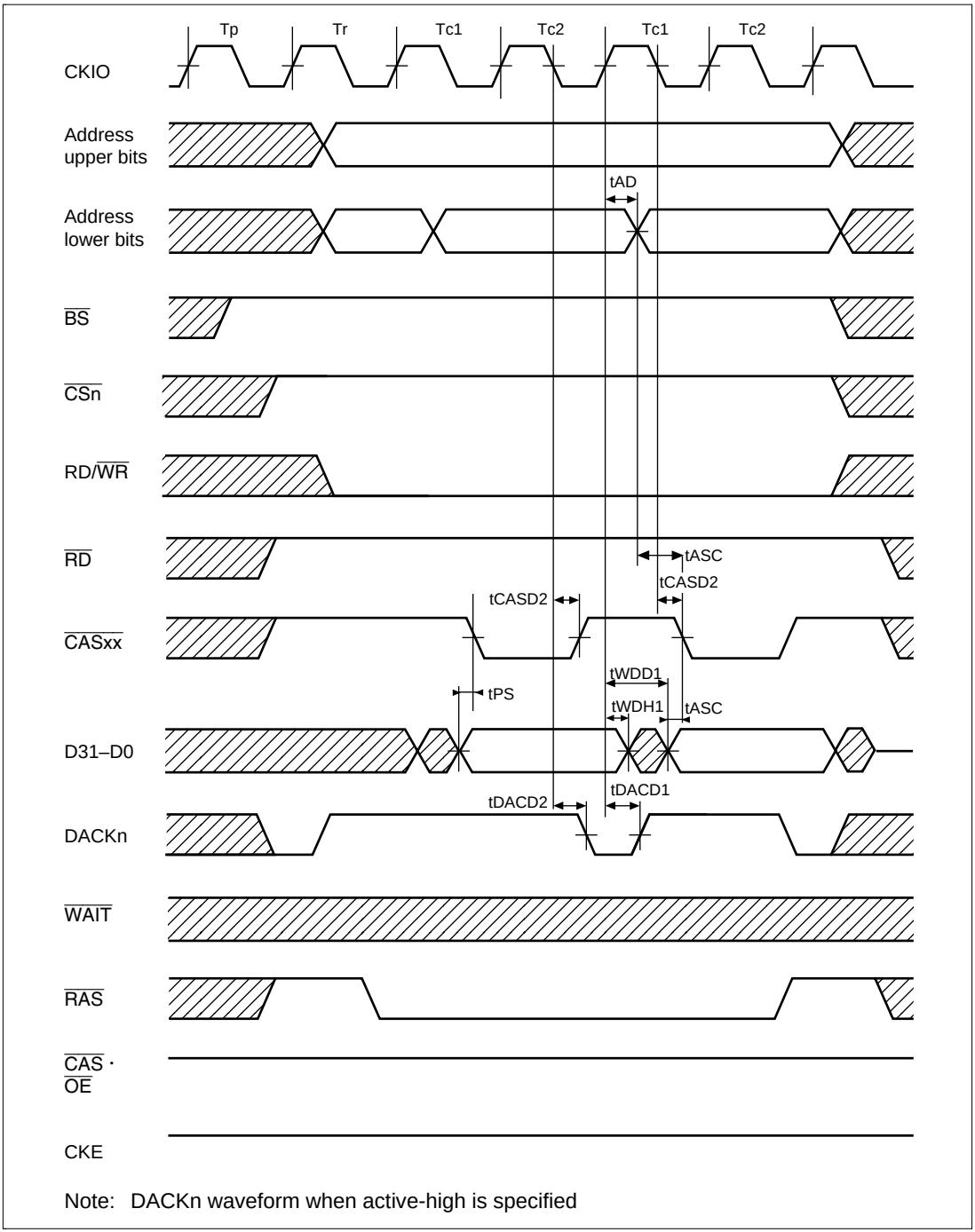

Note: DACKn waveform when active-high is specified

**Figure 23.36 DRAM Bus Cycle**
**(TRP = 1 Cycle, RCD = 1 Cycle, External Wait Input)**



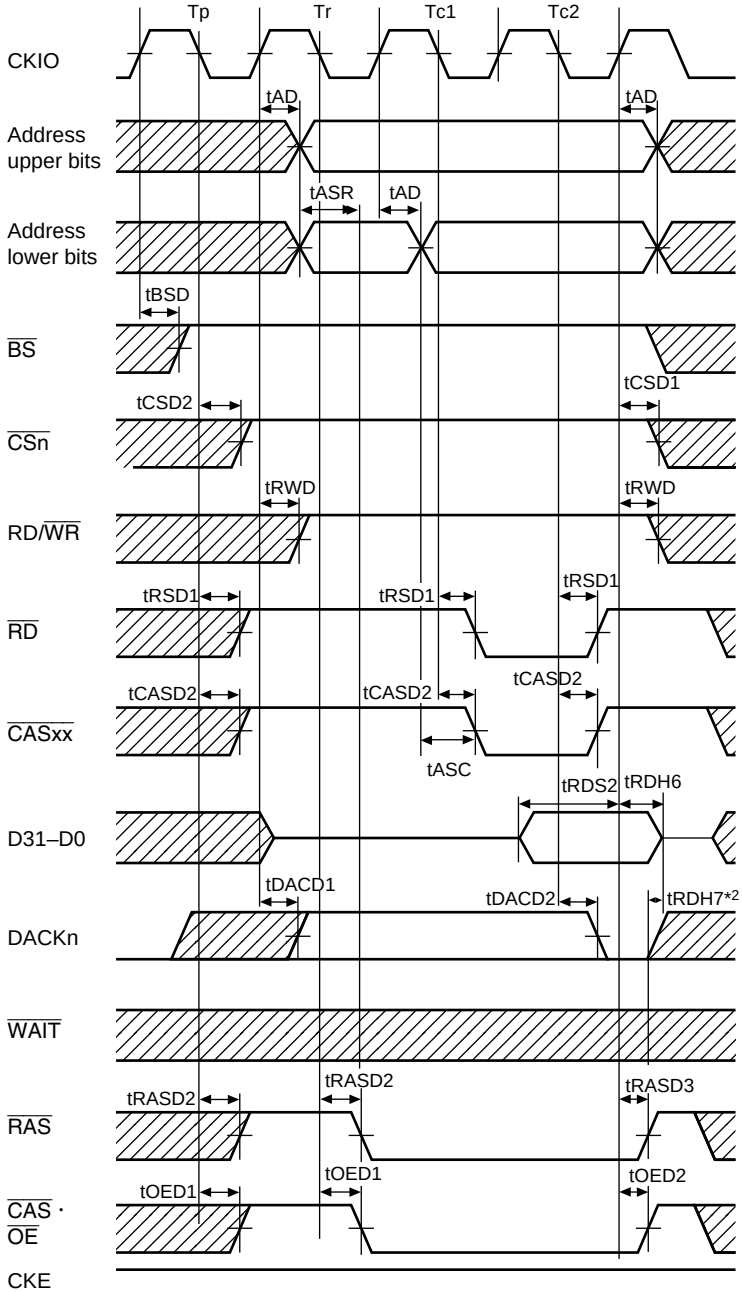

Notes: 1. $t_{RDH5}$ is measured from the rise of $\overline{RD}$ or $\overline{CASxx}$, whichever comes first.
2. DACKn waveform when active-high is specified

**Figure 23.37 DRAM Burst Read Cycle (TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**




Note: DACKn waveform when active-high is specified

**Figure 23.38 DRAM Burst Write Cycle (TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**




Notes: 1. DACKn waveform when active-high is specified
2. $t_{RDH7}$ is measured from the rise of $\overline{RAS}$ or $\overline{CAS} \cdot \overline{OE}$, whichever comes first.

**Figure 23.39 EDO Read Cycle**
**(TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**

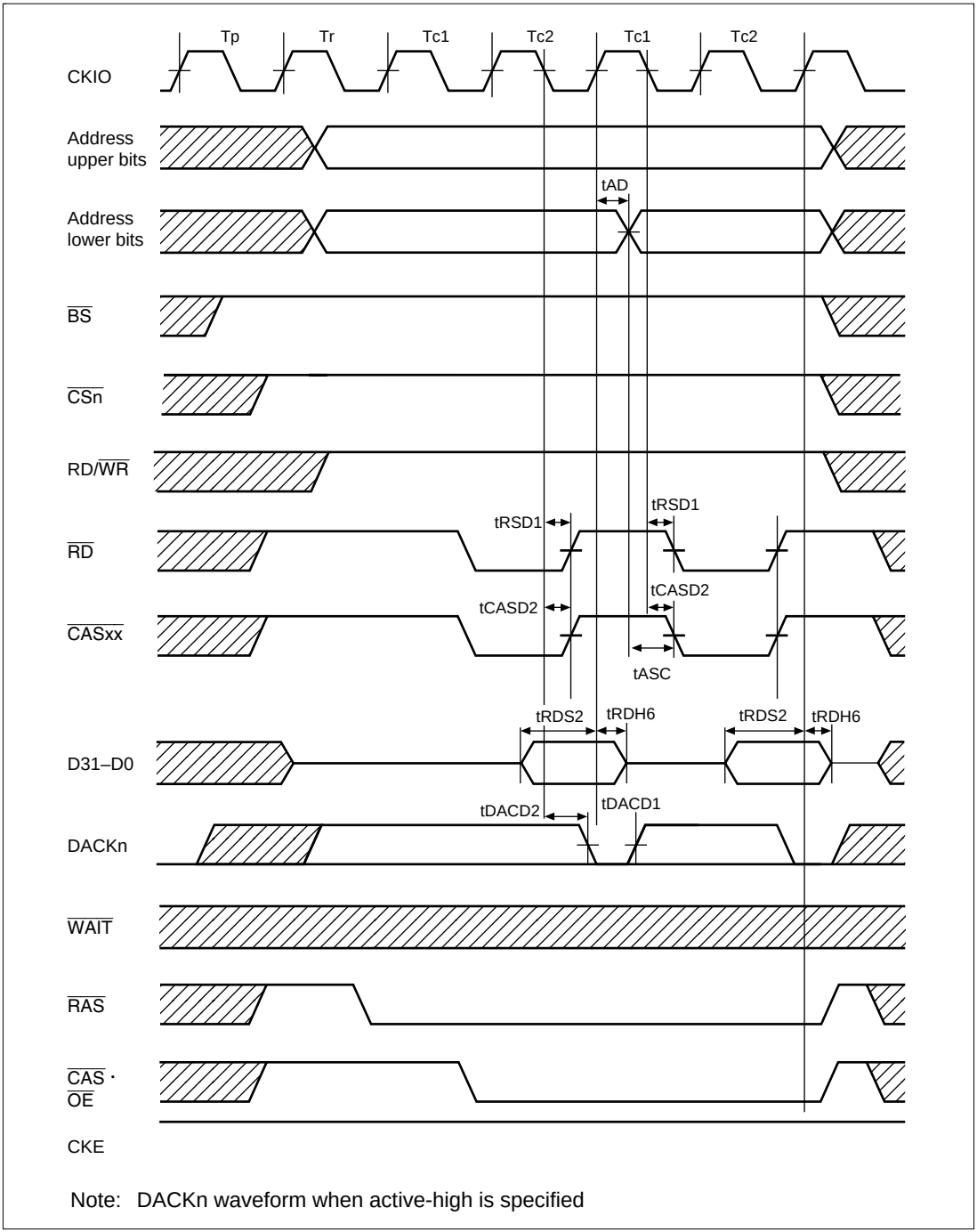



**Figure 23.40 EDO Burst Read Cycle (TRP = 1 Cycle, RCD = 1 Cycle, No Wait)**

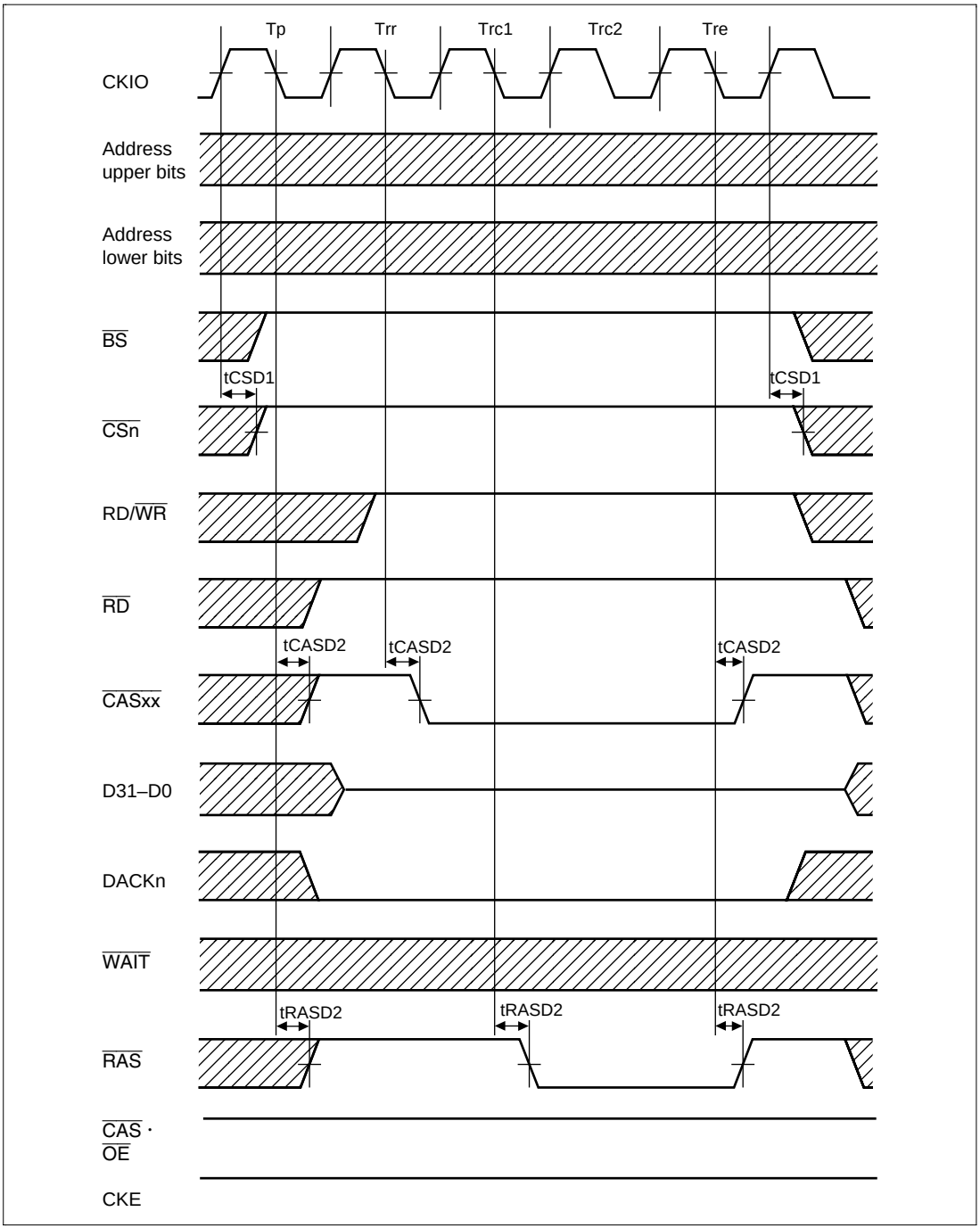
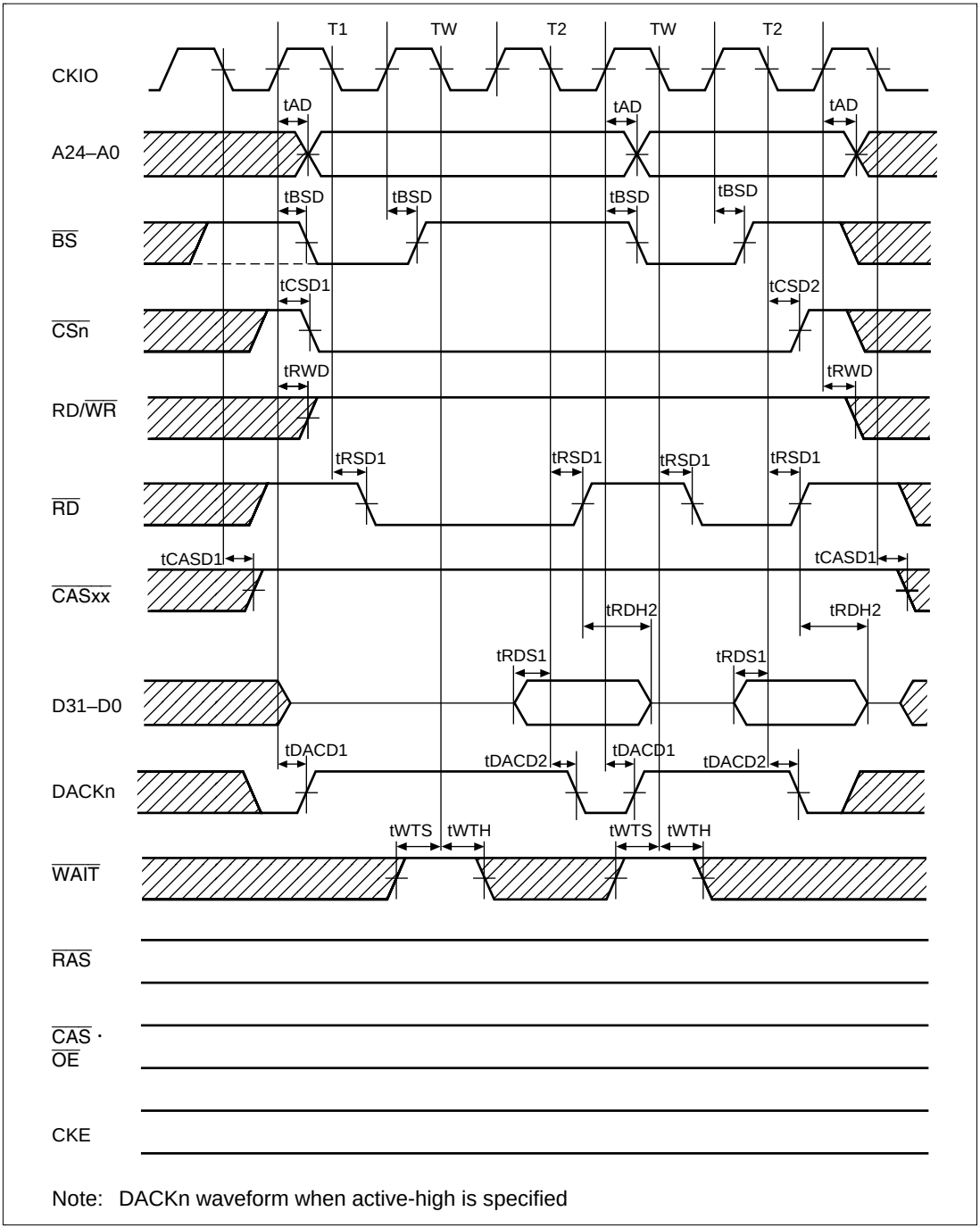



**Figure 23.41 DRAM CAS-Before-RAS Refresh Cycle (TRP = 1 Cycle, TRAS = 2 Cycles)**




Note: DACKn waveform when active-high is specified

**Figure 23.42 Burst ROM Read Cycle (Wait = 1)**

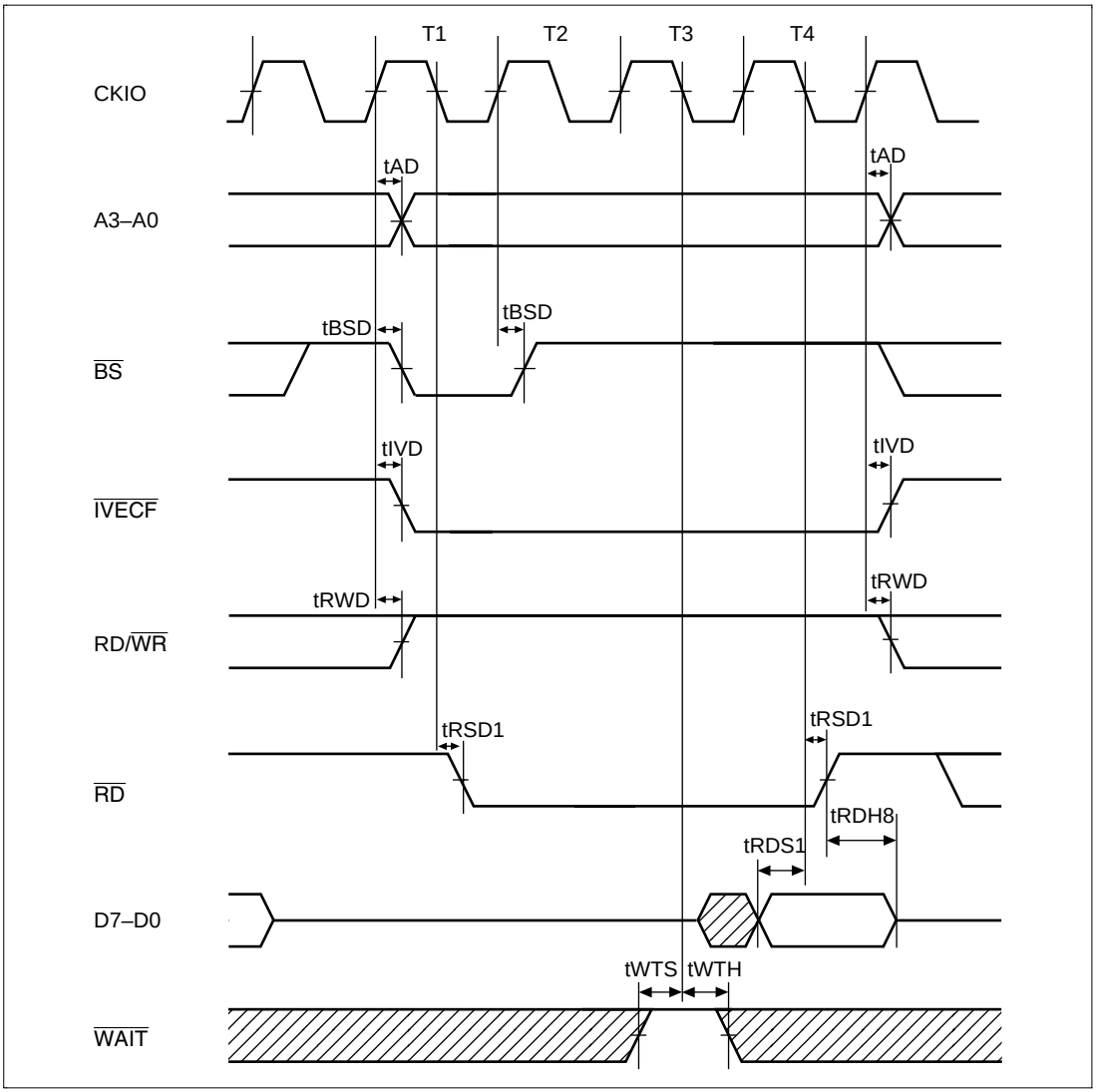



**Figure 23.43 Interrupt Vector Fetch Cycle**
**(No Wait, Iϕ:Eϕ = 1:1)**

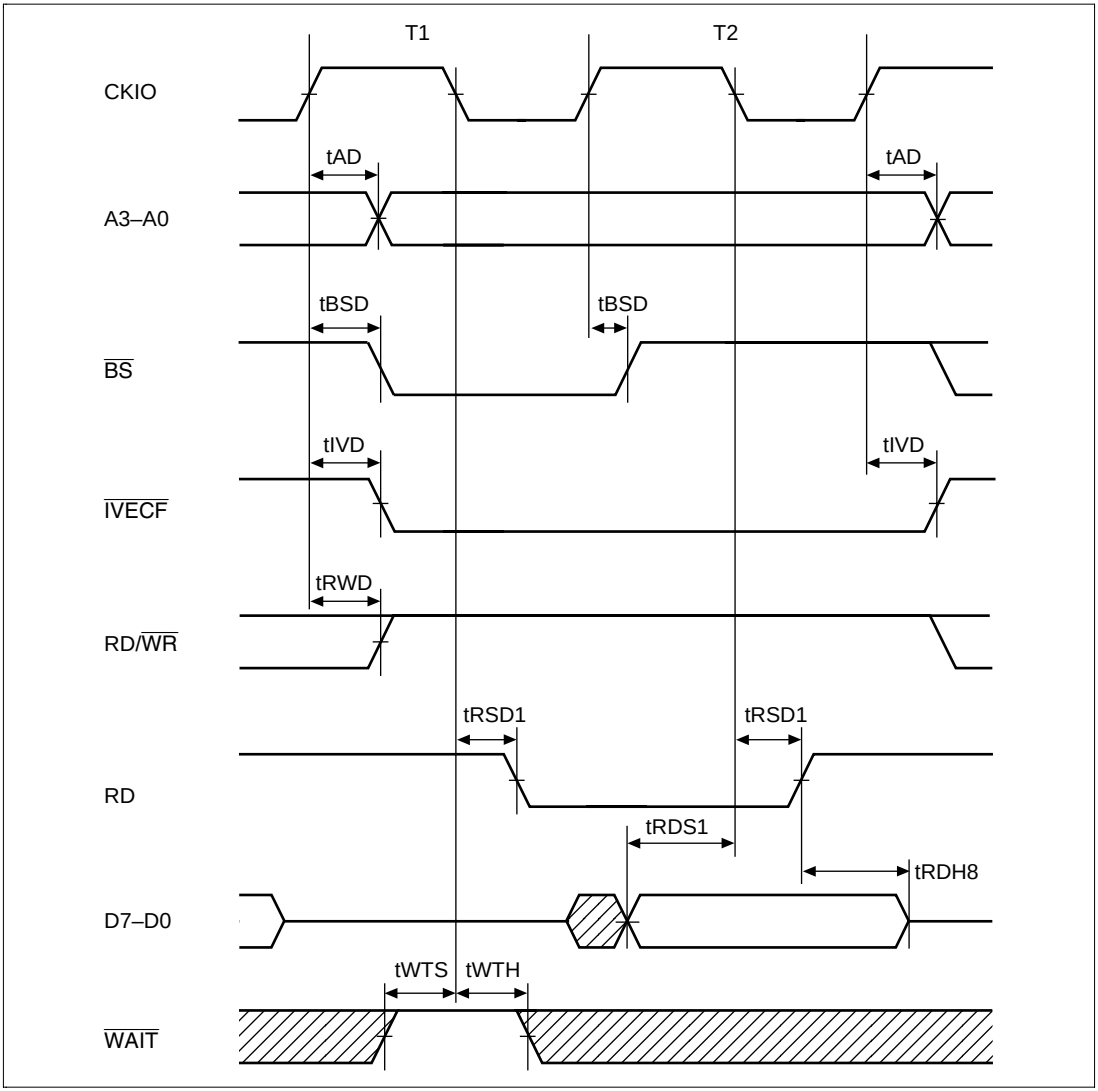



**Figure 23.44 Interrupt Vector Fetch Cycle**
**(No Wait, Iϕ:Eϕ ≠ 1:1)**

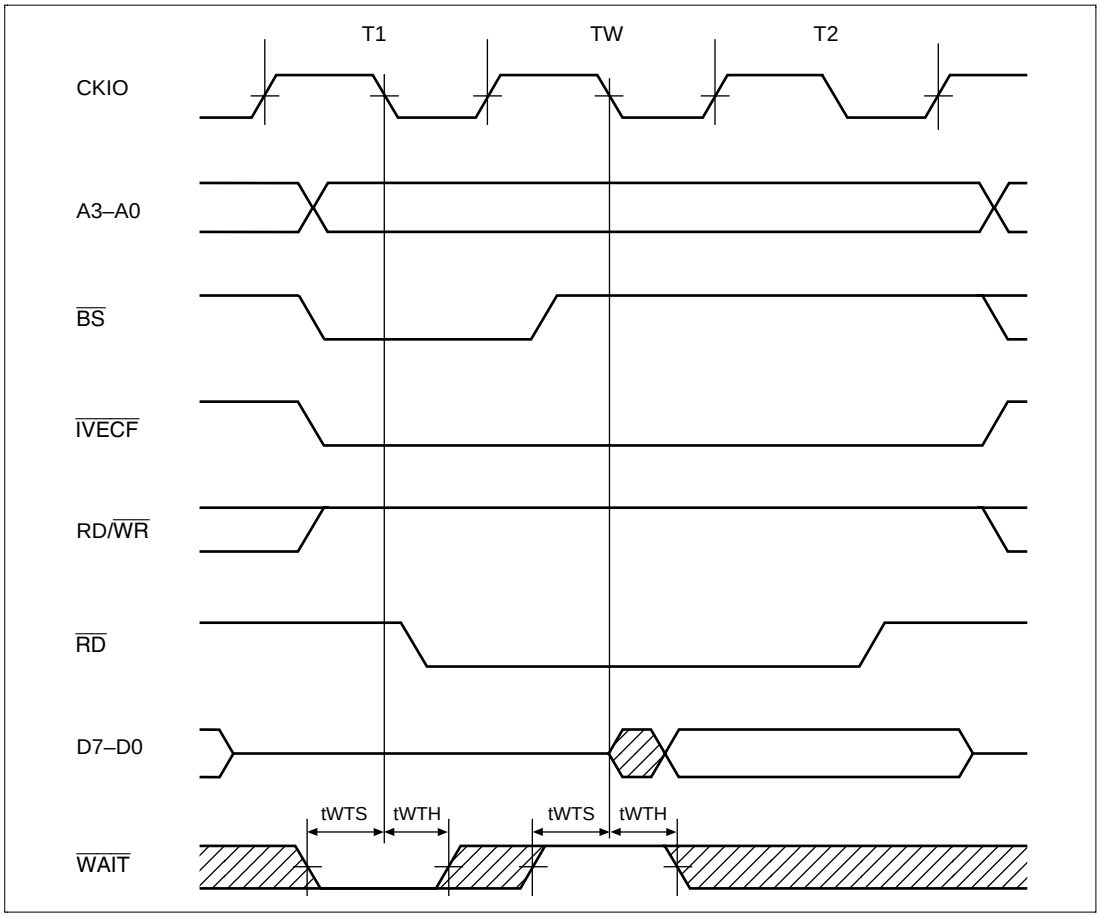



**Figure 23.45 Interrupt Vector Fetch Cycle (External Wait Input, Iϕ:Eϕ ≠ 1:1)**

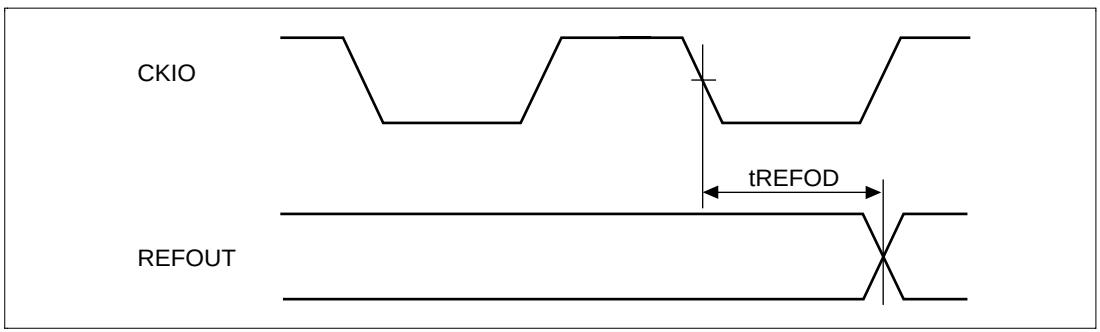



**Figure 23.46 REFOUT delay time**

### 23.3.4 Direct Memory Access Controller Timing

**Table 23.11 Direct Memory Access Controller Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| DREQ0, DREQ1 setup time | tDRQS | 10 | — | ns | 23.47 |
| DREQ0, DREQ1 hold time | tDRQH | 5 | — | ns | |

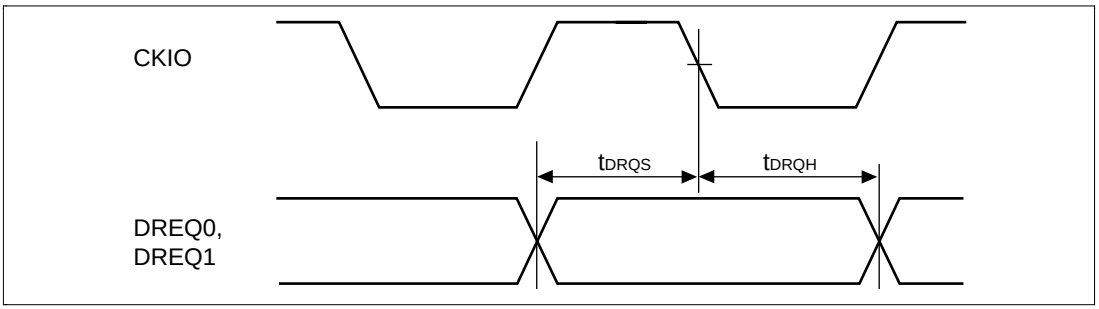



**Figure 23.47 DREQ0, DREQ1 Input Timing**

### 23.3.5 Free-Running Timer Timing

**Table 23.12 Free-Running Timer Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Output compare output delay time | tFOCD | — | 100 | ns | 23.48, 23.49 |
| Input capture input setup time (tEcyc:tPcyc = 1:1) | tFICS | 50 | — | ns | 23.48 |
| Input capture input setup time (tEcyc:tPcyc = 1:2) | tFICS | tcyc + 50 | — | ns | 23.49 |
| Input capture input setup time (tEcyc:tPcyc = 1:4) | tFICS | 3tcyc + 50 | — | ns | 23.49 |
| Input capture input hold time | tFICH | 50 | — | ns | 23.48, 23.49 |
| Timer clock input setup time (tEcyc:tPcyc = 1:1) | tFCKS | 50 | — | ns | 23.50 |
| Timer clock input setup time (tEcyc:tPcyc = 1:2) | tFCKS | tcyc + 50 | — | ns | 23.51 |
| Timer clock input setup time (tEcyc:tPcyc = 1:4) | tFCKS | 3tcyc + 50 | — | ns | 23.51 |
| Timer clock pulse width (single edge specified) | tFCKWH | 4.5 | — | tpcyc | 23.50, 23.51 |
| Timer clock pulse width (both edges specified) | tFCKWL | 8.5 | — | tpcyc | |

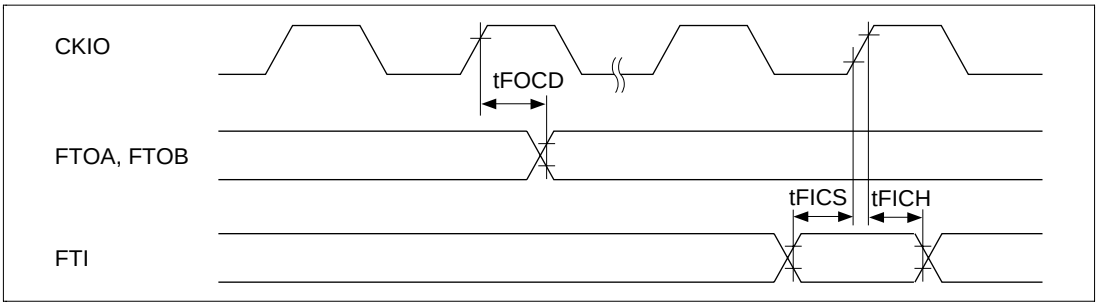



**Figure 23.48 FRT Input/Output Timing (tEcyc:tPcyc = 1:1)**

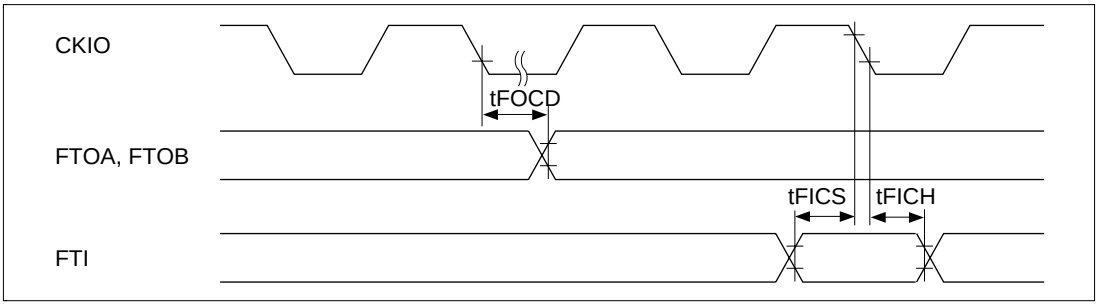



**Figure 23.49 FRT Input/Output Timing (tEcyc:tPcyc ≠ 1:1)**

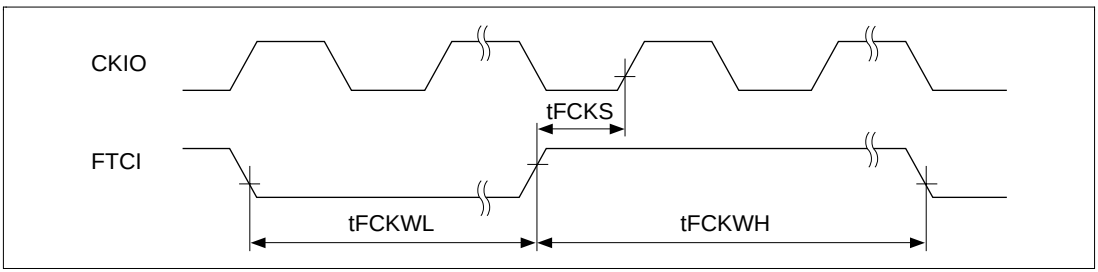



**Figure 23.50 FRT Clock Input Timing (tEcyc:tPcyc = 1:1)**

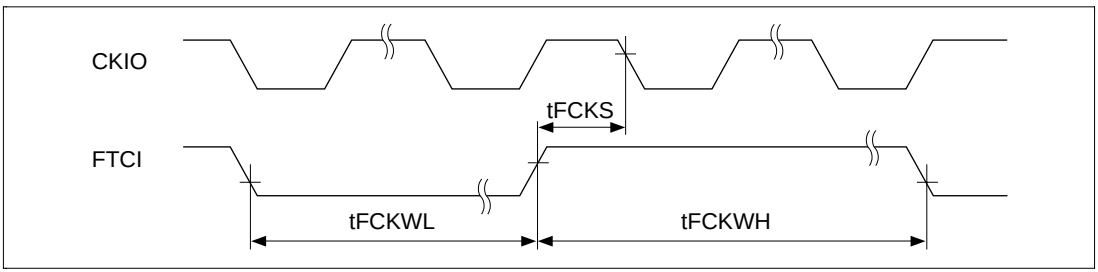



**Figure 23.51 FRT Clock Input Timing (tEcyc:tPcyc ≠ 1:1)**

### 23.3.6 Serial Communication Interface Timing

**Table 23.13 Serial Communication Interface Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Input clock cycle | tscyc | 4 | — | tpcyc | 23.52 |
| Input clock cycle (synchronous mode) | tscyc | 6 | — | tpcyc | 23.53 |
| Input clock pulse width | tSCKW | 0.4 | 0.6 | tcscyc | 23.52 |
| Transmit data delay time (synchronous mode) | tTXD | — | 100 | ns | 23.53 |
| Receive data setup time (synchronous mode) | tRXS | 100 | — | ns | |
| Receive data hold time (synchronous mode) | tRXH | 100 | — | ns | |
| $\overline{\text{RTS}}$ delay time | tRTSD | — | 100 | ns | 23.54 |
| $\overline{\text{CTS}}$ setup time (synchronous mode) | tCTSS | 100 | — | ns | |
| $\overline{\text{CTS}}$ hold time (synchronous mode) | tCTSH | 100 | — | ns | |

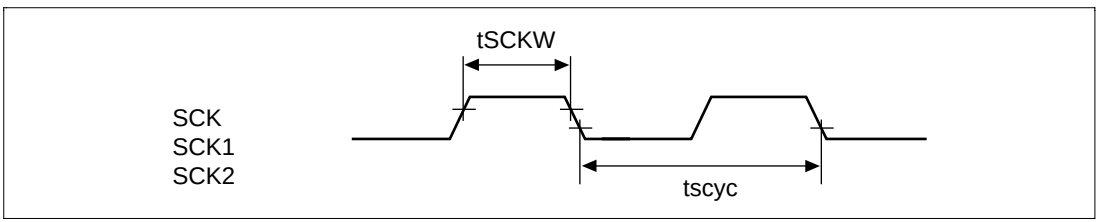



**Figure 23.52 Input Clock Input/Output Timing**

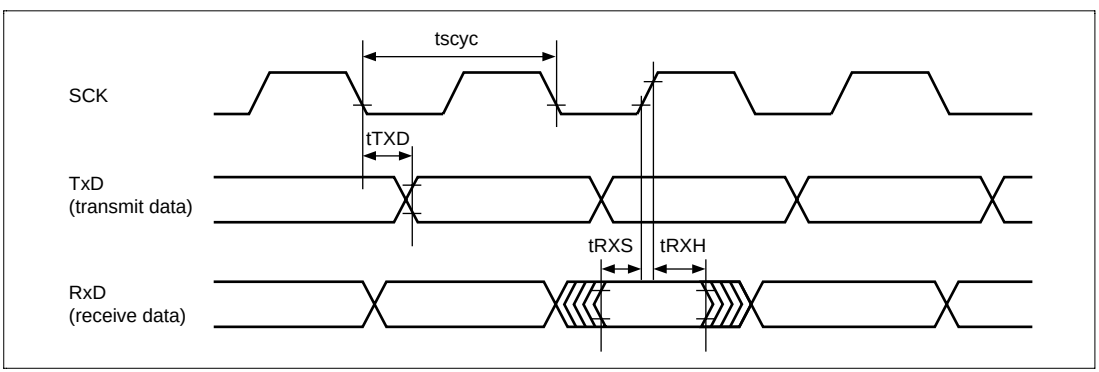



**Figure 23.53 SCI Input/Output Timing (Synchronous Mode)**

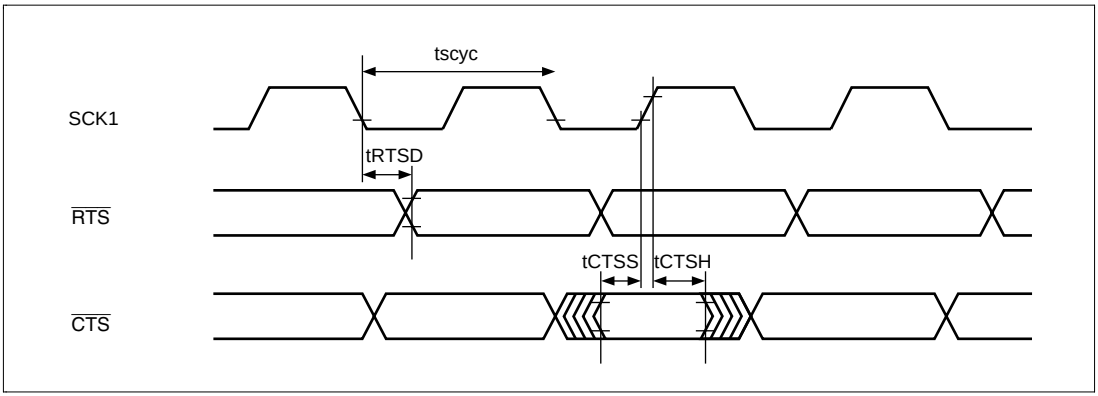



**Figure 23.53 $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ Input/Output Timing**

**Table 23.14 16-Bit Timer-Pulse Unit**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|---|
| Timer output delay time | | tTOCD | — | 100 | ns | 23.55, 23.56 |
| Timer input setup time (tEcyc:tPcyc = 1:1) | | tTICS | 50 | — | ns | |
| Timer input setup time (tEcyc:tPcyc = 1:2) | | tTICS | tcyc + 50 | — | ns | 23.55, 23.56 |
| Timer input setup time (tEcyc:tPcyc = 1:4) | | tTICS | 3tcyc + 50 | — | ns | |
| Timer clock input setup time (tEcyc:tPcyc = 1:1) | | tTCKS | 50 | — | ns | 23.57 |
| Timer clock input setup time (tEcyc:tPcyc = 1:2) | | tTCKS | tcyc + 50 | — | ns | |
| Timer clock input setup time (tEcyc:tPcyc = 1:4) | | tTCKS | 3tcyc + 50 | — | ns | |
| Timer clock pulse width | Single edge specified | tTCKWH | 1.5 | — | tcyc | |
| | Both edges specified | tTCKWL | 2.5 | — | | |

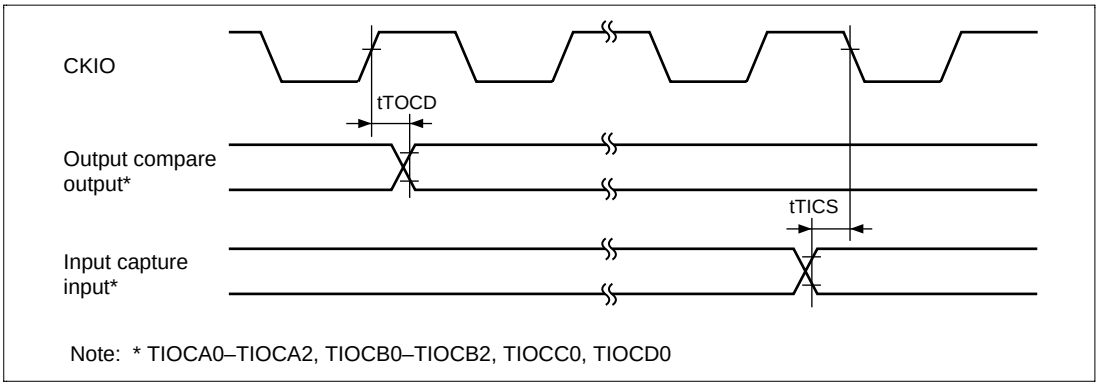



**Figure 23.55 TPU Input/Output Timing (tEcyc:tPcyc = 1:1)**

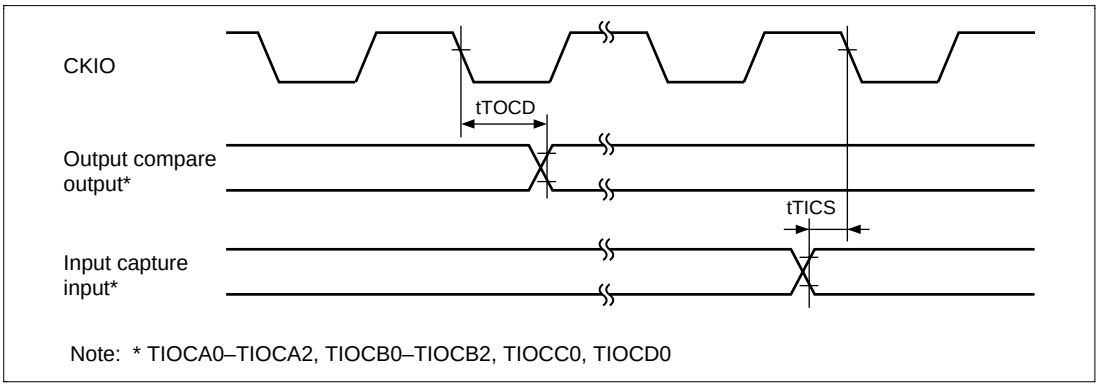



**Figure 23.56 TPU Input/Output Timing (tEcyc:tPcyc ≠ 1:1)**

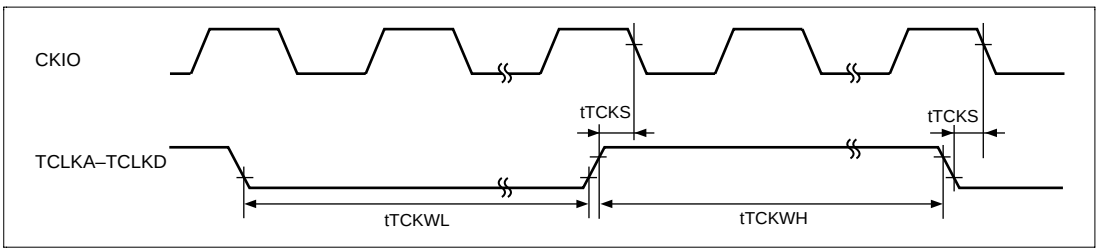



**Figure 23.57 TPU Clock Input Timing**

### 23.3.7 Watchdog Timer Timing

**Table 23.15 Watchdog Timer Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| $\overline{\text{WDTOVF}}$ delay time | tWOVD | — | 70 | ns | 23.58, 23.59 |

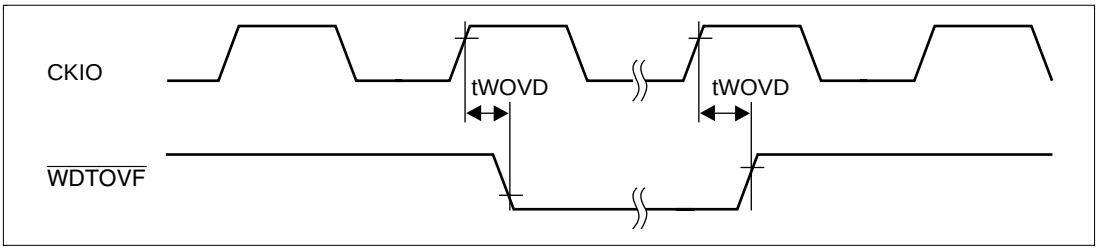



**Figure 23.58 Watchdog Timer Output Timing (tEcyc:tPcyc = 1:1)**

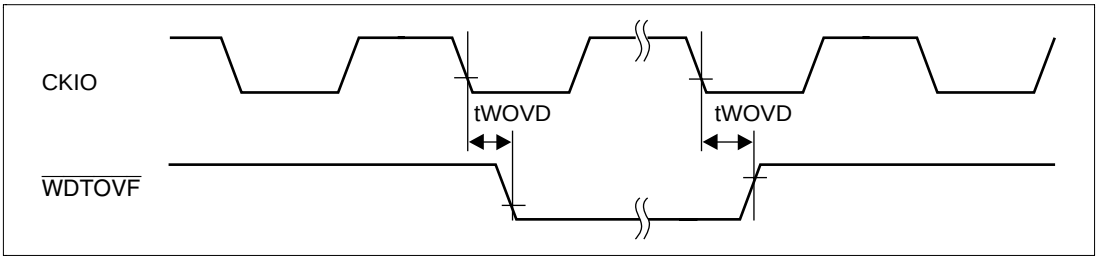



**Figure 23.59 Watchdog Timer Output Timing (tEcyc:tPcyc ≠ 1:1)**

### 23.3.8 Serial I/O Timing

**Table 23.16 Serial I/O Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| SRCK, STCK clock input cycle time | tIcyc | tpcyc or* 66.7 | — | ns | 23.60 |
| SRCK, STCK clock input low-level width | tWL | 0.4 × tsIcyc | — | ns | |
| SRCK, STCK clock input high-level width | tWH | 0.4 × tsIcyc | — | ns | |
| SRS input setup time | tRSS | 15 | — | ns | 23.61 |
| SRS input hold time | tRSH | 10 | — | ns | |
| SRXD input setup time | tSRDS | 15 | — | ns | |
| SRXD input hold time | tSRDH | 10 | — | ns | |
| STS input setup time | tTSS | 15 | — | ns | 23.62 |
| STS input hold time | tTSH | 10 | — | ns | |
| STS output delay time | tTSD | 0 | 20 | ns | 23.63 |
| STXD output delay time | tTDD | 0 | 20 | ns | 23.62, 23.63 |

Note: * Specified as tPcyc or 66.7, whichever is greater.

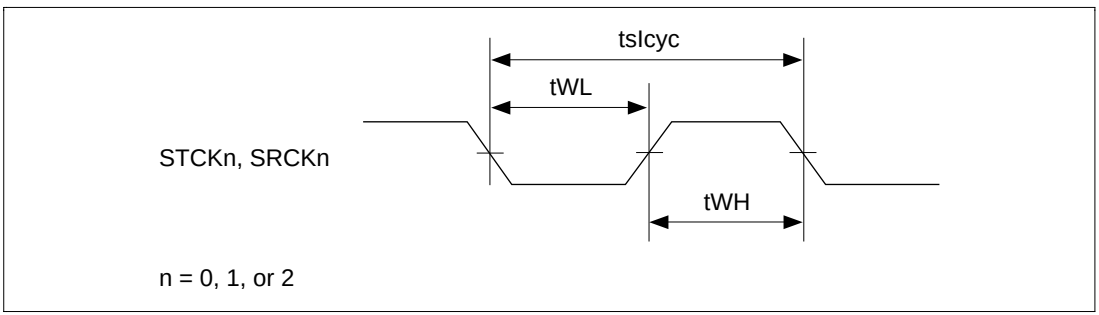



**Figure 23.60 SIO Input Clock Timing**

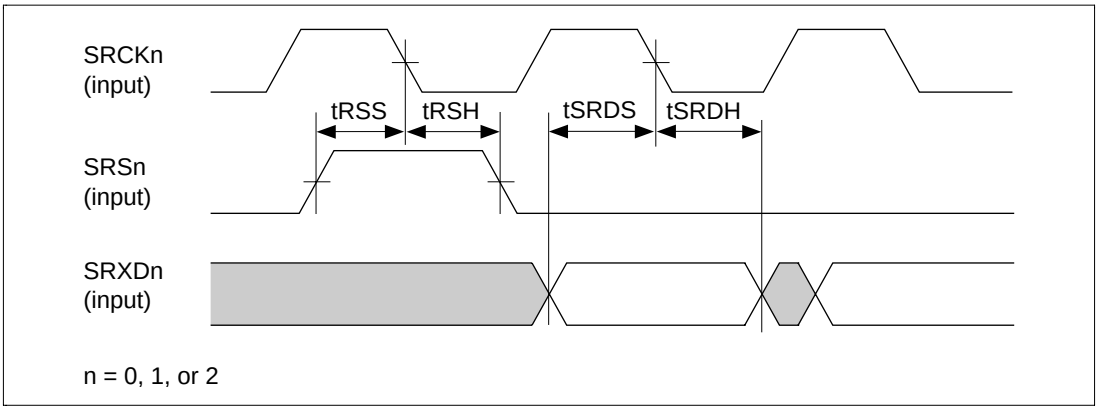



**Figure 23.61 SIO Receive Timing**

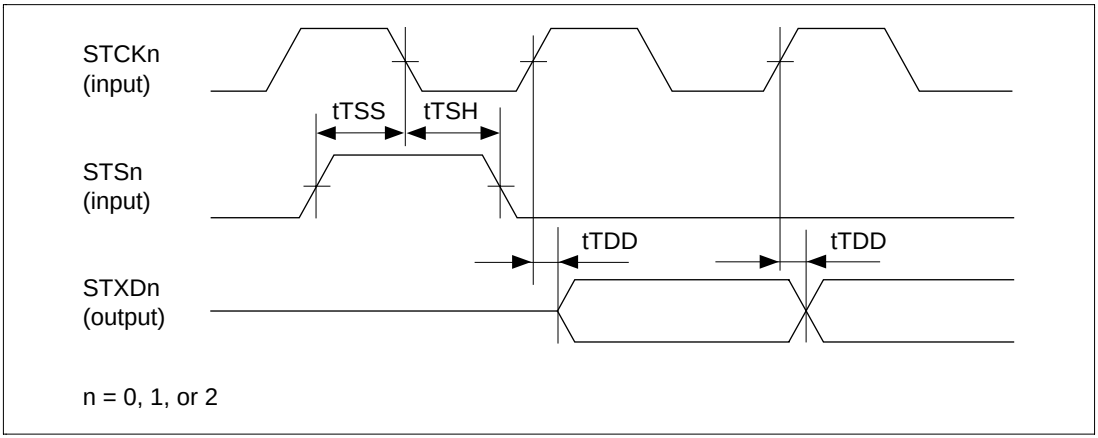



**Figure 23.62 SIO Transmit Timing (TMn = 0 Mode)**

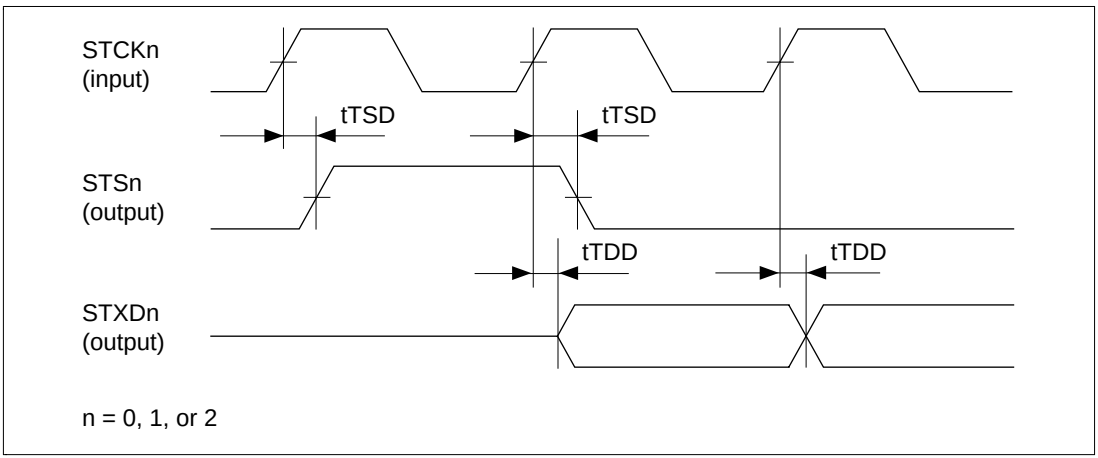



**Figure 23.63 SIO Transmit Timing (TMn = 1 Mode)**

### 23.3.9 Hitachi User Debug Interface Timing

**Table 23.17 Hitachi User Debug Interface Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| TCK clock input cycle time | tPcyc | tPcyc or* 66.7 ns | — | ns | 23.64 |
| TCK clock input high-level width | tTCKH | 0.4 | 0.6 | tIcyc | |
| TCK clock input low-level width | tTCKL | 0.4 | 0.6 | tIcyc | |
| $\overline{\text{TRST}}$ pulse width | tTRSW | 20 | — | tIcyc | 23.65 |
| $\overline{\text{TRST}}$ setup time | tTRSS | 40 | — | ns | |
| TMS setup time | tTMSS | 30 | — | ns | 23.66 |
| TMS hold time | tTMSH | 10 | — | ns | |
| TDI setup time | tTDIS | 30 | — | ns | |
| TDI hold time | tTDIH | 10 | — | ns | |
| TDO delay time | tTDOD | 0 | 30 | ns | |

Note: * Specified as tPcyc or 66.7, whichever is greater.

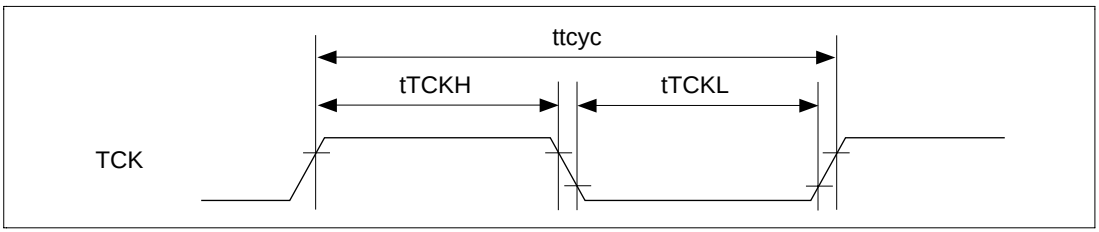



**Figure 23.64 H-UDI Clock Timing**

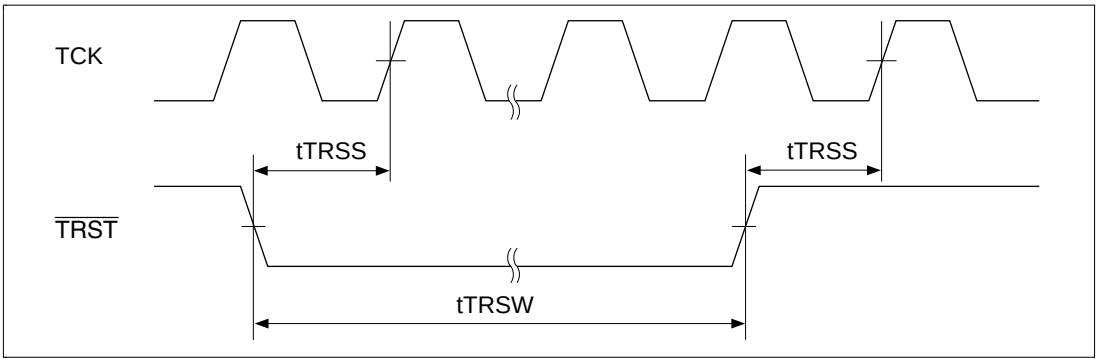



**Figure 23.65 H-UDI $\overline{\text{TRST}}$ Timing**

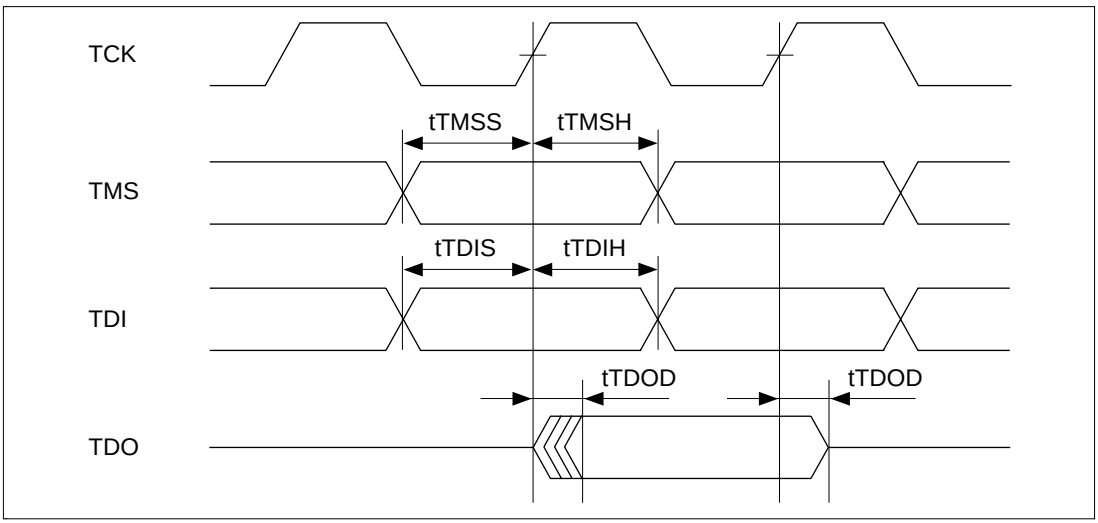



**Figure 23.66 H-UDI Input/Output Timing**

### 23.3.10 I/O Port Timing

**Table 23.18 I/O Port Timing**

Conditions: Vcc = 3.0 to 3.6 V, Ta = –20 to +75°C

| Item | Symbol | Min | Max | Unit | Figure |
|---|---|---|---|---|---|
| Port output data delay time | tPWD | — | 50 | ns | 23.67, 23.68 |
| Port input data setup time (tEcyc:tPcyc = 1:1) | tPRS | 50 | — | ns | 23.67 |
| Port input data setup time (tEcyc:tPcyc = 1:2) | tPRS | tcyc + 50 | — | ns | 23.68 |
| Port input data setup time (tEcyc:tPcyc = 1:4) | tPRS | 3tcyc + 50 | — | ns | |
| Port input data hold time | tPRH | 50 | — | ns | 23.67, 23.68 |

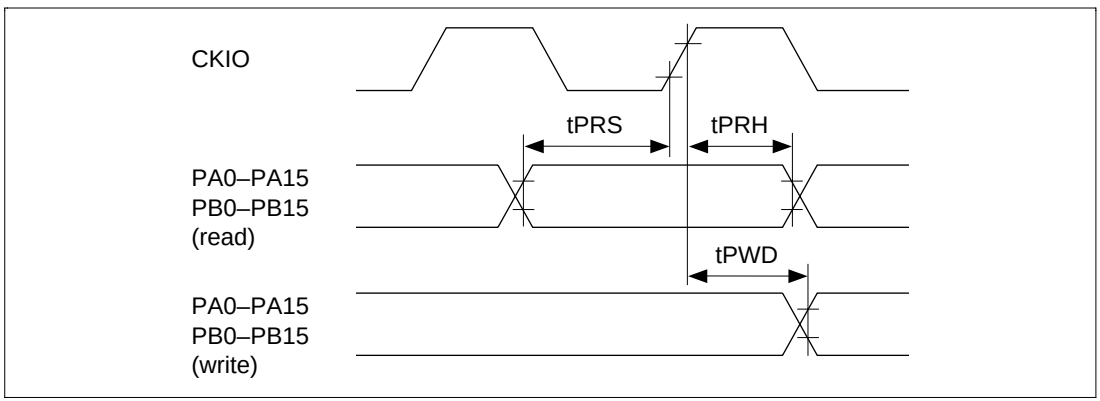



**Figure 23.67 I/O Port Input/Output Timing (tEcyc:tPcyc = 1:1)**

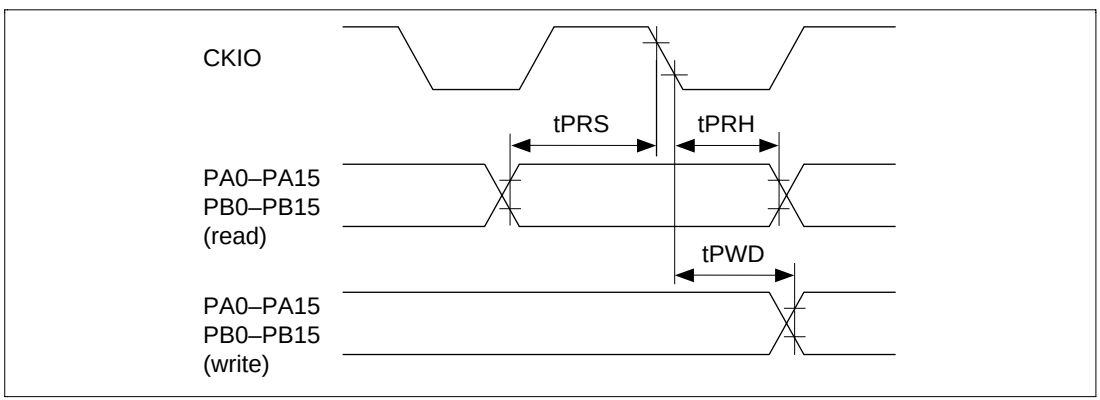


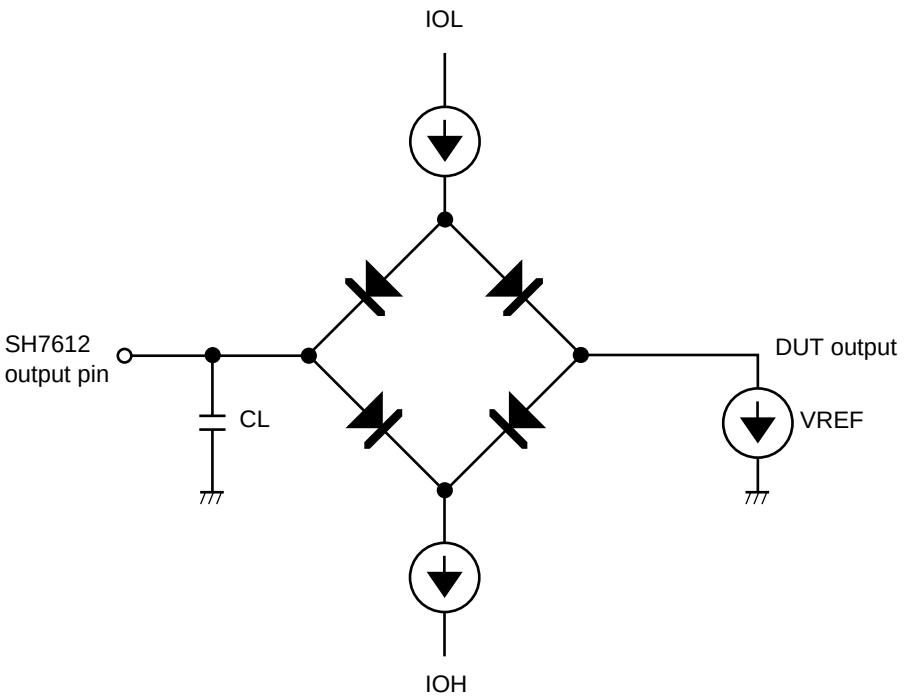

**Figure 23.68 I/O Port Input/Output Timing (tEcyc:tPcyc ≠ 1:1)**

## 23.4 AC Characteristic Test Conditions

The AC characteristic test conditions are as follows:

- Input/output signal reference level: 1.5 V (Vcc = 3.3 to 3.6 V)
- Input pulse level: Vss to 3.0 V (Vss to Vcc for $\overline{\text{RES}}$, $\overline{\text{TRST}}$, EXTAL, CKIO, MD0–MD4, and NMI)
- Input rise/fall time: 1 ns

The output load circuit is shown in figure 23.69.




CL is the total value, including the capacitance of the test jig, etc. The capacitance of each pin is as follows:

30 pF: CKIO, A24–A0, D31–D0, $\overline{\text{BS}}$, $\overline{\text{RD}}$, $\overline{\text{CS4}}$–$\overline{\text{CS0}}$, DQMUU/$\overline{\text{WE3}}$–DQMLL/$\overline{\text{WE0}}$, $\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$, $\overline{\text{RAS}}$, $\overline{\text{CAS}}$/$\overline{\text{OE}}$, DACK1, DACK0

50 pF: All other pins

IOL and IOH values are as shown in table 23.3, Permissible Output Currents.

**Figure 23.69 Output Load Circuit**

# Appendix A On-Chip Peripheral Module Registers

## A.1 Addresses

On-chip peripheral module register addresses and bit names are shown in the following table. 16-bit registers and 32-bit registers are shown, respectively, in two and four lines of 8 bits.

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFC00 | SIRDR0 | | | | | | | | | SIO |
| H'FFFFFC01 | | | | | | | | | | |
| H'FFFFFC02 | SITDR0 | | | | | | | | | |
| H'FFFFFC03 | | | | | | | | | | |
| H'FFFFFC04 | SICTR0 | — | — | — | — | — | — | — | — | |
| H'FFFFFC05 | | — | TM | SE | DL | TIE | RIE | TE | RE | |
| H'FFFFFC06 | SISTR0 | — | — | — | — | — | — | — | — | |
| H'FFFFFC07 | | — | — | — | — | TERR | RERR | TDRE | RDRF | |
| H'FFFFFC08 to H'FFFFFC0F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC10 | SIRDR1 | | | | | | | | | SIO |
| H'FFFFFC11 | | | | | | | | | | |
| H'FFFFFC12 | SITDR1 | | | | | | | | | |
| H'FFFFFC13 | | | | | | | | | | |
| H'FFFFFC14 | SICTR1 | — | — | — | — | — | — | — | — | |
| H'FFFFFC15 | | — | TM | SE | DL | TIE | RIE | TE | RE | |
| H'FFFFFC16 | SISTR1 | — | — | — | — | — | — | — | — | |
| H'FFFFFC17 | | — | — | — | — | TERR | RERR | TDRE | RDRF | |
| H'FFFFFC18 to H'FFFFFC1F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC20 | SIRDR2 | | | | | | | | | SIO |
| H'FFFFFC21 | | | | | | | | | | |
| H'FFFFFC22 | SITDR2 | | | | | | | | | |
| H'FFFFFC23 | | | | | | | | | | |
| H'FFFFFC24 | SICTR2 | — | — | — | — | — | — | — | — | |
| H'FFFFFC25 | | — | TM | SE | DE | TIE | RIE | TE | RE | |
| H'FFFFFC26 | SISTR2 | — | — | — | — | — | — | — | — | |
| H'FFFFFC27 | | — | — | — | — | TERR | RERR | TDRE | RDRF | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFC28 to H'FFFFFC3F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC40 | TSTR | — | — | — | — | — | CST2 | CST1 | CST0 | TPU |
| H'FFFFFC41 | TSYR | — | — | — | — | — | SYNC2 | SYNC1 | SYNC0 | |
| H'FFFFFC42 to H'FFFFFC4F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC50 | TCR0 | CCLR2 | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU |
| H'FFFFFC51 | TMDR0 | — | — | BFB | BFA | MD3 | MD2 | MD1 | MD0 | |
| H'FFFFFC52 | TIOR0H | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| H'FFFFFC53 | TIOR0L | IOD3 | IOD2 | IOD1 | IOD0 | IOC3 | IOC2 | IOC1 | IOC0 | |
| H'FFFFFC54 | TIER0 | — | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA | |
| H'FFFFFC55 | TSR0 | — | — | — | TCFV | TGFD | TGFC | TGFB | TGFA | |
| H'FFFFFC56 | TCNT0 | | | | | | | | | |
| H'FFFFFC57 | | | | | | | | | | |
| H'FFFFFC58 | TGR0A | | | | | | | | | |
| H'FFFFFC59 | | | | | | | | | | |
| H'FFFFFC5A | TGR0B | | | | | | | | | |
| H'FFFFFC5B | | | | | | | | | | |
| H'FFFFFC5C | TGR0C | | | | | | | | | |
| H'FFFFFC5D | | | | | | | | | | |
| H'FFFFFC5E | TGR0D | | | | | | | | | |
| H'FFFFFC5F | | | | | | | | | | |
| H'FFFFFC60 | TCR1 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | |
| H'FFFFFC61 | TMDR1 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| H'FFFFFC62 | TIOR1 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| H'FFFFFC63 | — | — | — | — | — | — | — | — | — | |
| H'FFFFFC64 | TIER1 | — | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| H'FFFFFC65 | TSR1 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| H'FFFFFC66 | TCNT1 | | | | | | | | | |
| H'FFFFFC67 | | | | | | | | | | |
| H'FFFFFC68 | TGR1A | | | | | | | | | |
| H'FFFFFC69 | | | | | | | | | | |
| H'FFFFFC6A | TGR1B | | | | | | | | | |
| H'FFFFFC6B | | | | | | | | | | |
| H'FFFFFC6C to H'FFFFFC6F | — | — | — | — | — | — | — | — | — | — |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFC70 | TCR2 | — | CCLR1 | CCLR0 | CKEG1 | CKEG0 | TPSC2 | TPSC1 | TPSC0 | TPU |
| H'FFFFFC71 | TMDR2 | — | — | — | — | MD3 | MD2 | MD1 | MD0 | |
| H'FFFFFC72 | TIOR2 | IOB3 | IOB2 | IOB1 | IOB0 | IOA3 | IOA2 | IOA1 | IOA0 | |
| H'FFFFFC73 | — | — | — | — | — | — | — | — | — | |
| H'FFFFFC74 | TIER2 | — | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA | |
| H'FFFFFC75 | TSR2 | TCFD | — | TCFU | TCFV | — | — | TGFB | TGFA | |
| H'FFFFFC76 | TCNT2 | | | | | | | | | |
| H'FFFFFC77 | | | | | | | | | | |
| H'FFFFFC78 | TGR2A | | | | | | | | | |
| H'FFFFFC79 | | | | | | | | | | |
| H'FFFFFC7A | TGR2B | | | | | | | | | |
| H'FFFFFC7B | | | | | | | | | | |
| H'FFFFFC7C to H'FFFFFC7F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC80 | PACR | — | — | PA13MD | PA12MD | PA11MD | PA10MD | PA9MD | PA8MD | PFC |
| H'FFFFFC81 | | PA7MD | PA6MD | PA5MD | PA4MD | PA3MD | PA2MD | PA1MD | PA0MD | |
| H'FFFFFC82 | PAIOR | — | — | PA13IOR | PA12IOR | PA11IOR | PA10IOR | PA9IOR | PA8IOR | |
| H'FFFFFC83 | | PA7IOR | PA6IOR | PA5IOR | PA4IOR | — | PA2IOR | PA1IOR | PA0IOR | |
| H'FFFFFC84 | PADR | — | — | PA13DR | PA12DR | PA11DR | PA10DR | PA9DR | PA8DR | I/O port |
| H'FFFFFC85 | | PA7DR | PA6DR | PA5DR | PA4DR | — | PA2DR | PA1DR | PA0DR | |
| H'FFFFFC86 to H'FFFFFC87 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFC88 | PBCR | PB15MD1 | PB15MD0 | PB14MD1 | PB14MD0 | PB13MD1 | PB13MD0 | PB12MD1 | PB12MD0 | PFC |
| H'FFFFFC89 | | PB11MD1 | PB11MD0 | PB10MD1 | PB10MD0 | PB9MD1 | PB9MD0 | PB8MD1 | PB8MD0 | |
| H'FFFFFC8A | PBIOR | PB15IOR | PB14IOR | PB13IOR | PB12IOR | PB11IOR | PB10IOR | PB9IOR | PB8IOR | |
| H'FFFFFC8B | | PB7IOR | PB6IOR | PB5IOR | PB4IOR | PB3IOR | PB2IOR | PB1IOR | PB0IOR | |
| H'FFFFFC8C | PBDR | PB15DR | PB14DR | PB13DR | PB12DR | PB11DR | PB10DR | PB9DR | PB8DR | I/O port |
| H'FFFFFC8D | | PB7DR | PB6DR | PB5DR | PB4DR | PB3DR | PB2DR | PB1DR | PB0DR | |
| H'FFFFFC8E | PBCR2 | PB7MD1 | PB7MD0 | PB6MD1 | PB6MD0 | PB5MD1 | PB5MD0 | PB4MD1 | PB4MD0 | PFC |
| H'FFFFFC8F | | PB3MD1 | PB3MD0 | PB2MD1 | PB2MD0 | PB1MD1 | PB1MD0 | PB0MD1 | PB0MD0 | |
| H'FFFFFC90 to H'FFFFFCBF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCC0 | SCSMR1 | — | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 | SCIF |
| | | IRMOD | ICK3 | ICK2 | ICK1 | ICK0 | PSEL | CKS1 | CKS0 | IrDA |
| H'FFFFFCC1 | — | — | — | — | — | — | — | — | — | — |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFCC2 | SCBRR1 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCC3 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCC4 | SCSCR1 | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 | SCIF |
| | | TIE | RIE | TE | RE | | | CKE1 | CKE0 | IrDA |
| H'FFFFFCC5 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCC6 | SCFTDR1 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCC7 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCC8 | SCSSR1 | PER3 | PER2 | PER1 | PER0 | FER3 | FER2 | FER1 | FER0 | SCIF |
| H'FFFFFCC9 | | ER | TEND | TDFE | BRK | FER | PER | RDF | DR | |
| H'FFFFFCC8 | | PER3 | PER2 | PER1 | PER0 | FER3 | FER2 | FER1 | FER0 | IrDA |
| H'FFFFFCC9 | | ER | TEND | TDFE | BRK | FER | PER | RDF | DR | |
| H'FFFFFCCA | SCFRDR1 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCCB | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCCC | SCFCR1 | RTRG1 | RTRG0 | TTRG1 | TTRG0 | MCE | TFRST | RFRST | LOOP | SCIF |
| | | RTRG1 | RTRG0 | TTRG1 | TTRG0 | MCE | TFRST | RFRST | LOOP | IrDA |
| H'FFFFFCCD | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCCE | SCFDR1 | — | — | — | T4 | T3 | T2 | T1 | T0 | SCIF |
| H'FFFFFCCF | | — | — | — | R4 | R3 | R2 | R1 | R0 | |
| H'FFFFFCCE | | — | — | — | T4 | T3 | T2 | T1 | T0 | IrDA |
| H'FFFFFCCF | | — | — | — | R4 | R3 | R2 | R1 | R0 | |
| H'FFFFFCD0 | SCSMR2 | — | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 | SCIF |
| | | IRMOD | ICK3 | ICK2 | ICK1 | ICK0 | PSEL | CKS1 | CKS0 | IrDA |
| H'FFFFFCD1 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCD2 | SCBRR2 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCD3 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCD4 | SCSCR2 | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 | SCIF |
| | | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 | IrDA |
| H'FFFFFCD5 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCD6 | SCFTDR2 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCD7 | — | — | — | — | — | — | — | — | — | — |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFCD8 | SCSSR2 | PER3 | PER2 | PER1 | PER0 | FER3 | FER2 | FER1 | FER0 | SCIF |
| H'FFFFFCD9 | | ER | TEND | TDFE | BRK | FER | PER | RDF | DR | |
| H'FFFFFCD8 | | PER3 | PER2 | PER1 | PER0 | FER3 | FER2 | FER1 | FER0 | IrDA |
| H'FFFFFCD9 | | ER | TEND | TDFE | BRK | FER | PER | RDF | DR | |
| H'FFFFFCDA | SCFRDR2 | | | | | | | | | SCIF |
| | | | | | | | | | | IrDA |
| H'FFFFFCDB | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCDC | SCFCR2 | RTRG1 | RTRG0 | TTRG1 | TTRG0 | MCE | TFRST | RFRST | LOOP | SCIF |
| | | RTRG1 | RTRG0 | TTRG1 | TTRG0 | MCE | TFRST | RFRST | LOOP | IrDA |
| H'FFFFFCDD | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFCDE | SCFDR2 | — | — | — | T4 | T3 | T2 | T1 | T0 | SCIF |
| H'FFFFFCDF | | — | — | — | R4 | R3 | R2 | R1 | R0 | |
| H'FFFFFCDE | | — | — | — | T4 | T3 | T2 | T1 | T0 | IrDA |
| H'FFFFFCDF | | — | — | — | R4 | R3 | R2 | R1 | R0 | |
| H'FFFFFCE0 | SDIR | TS3 | TS2 | TS1 | TS0 | — | — | — | — | H-UDI |
| H'FFFFFCE1 | | — | — | — | — | — | — | — | — | |
| H'FFFFFCE2 | SDSR | — | — | — | — | — | — | — | — | |
| H'FFFFFCE3 | | — | — | — | — | — | — | — | SDTRF | |
| H'FFFFFCE4 | SDDRH | | | | | | | | | |
| H'FFFFFCE5 | | | | | | | | | | |
| H'FFFFFCE6 | SDDRL | | | | | | | | | |
| H'FFFFFCE7 | | | | | | | | | | |
| H'FFFFFCE8 to H'FFFFFDFF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE00 | SMR | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | MP | CKS1 | CKS0 | SCI |
| | | GM | CHR | PE | O/$\overline{E}$ | STOP | MP | CKS1 | CKS0 | Smart card interface |
| H'FFFFFE01 | BRR | | | | | | | | | SCI |
| | | | | | | | | | | Smart card interface |
| H'FFFFFE02 | SCR | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 | SCI |
| | | TIE | RIE | TE | RE | MPIE | TEIE | CKE1 | CKE0 | Smart card interface |
| H'FFFFFE03 | TDR | | | | | | | | | SCI |
| | | | | | | | | | | Smart card interface |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFE04 | SSR | TDRE | RDRF | ORER | FER | PER | TEND | MPB | MPBT | SCI |
| | | TDRE | RDRF | ORER | ERS | PER | TEND | MPB | MPBT | Smart card interface |
| H'FFFFFE05 | RDR | | | | | | | | | SCI |
| | | | | | | | | | | Smart card interface |
| H'FFFFFE06 | SCMR | — | — | — | — | SDIR | SINV | — | SMIF | Smart card interface |
| H'FFFFFE07 to H'FFFFFE0F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE10 | TIER | ICIE | — | — | — | OCIAE | OCIBE | OVIE | — | FRT |
| H'FFFFFE11 | FTCSR | ICF | — | — | — | OCFA | OCFB | OVF | CCLRA | |
| H'FFFFFE12 | FRC H | | | | | | | | | |
| H'FFFFFE13 | FRC L | | | | | | | | | |
| H'FFFFFE14*[1] | OCRA H | | | | | | | | | |
| | OCRB H | | | | | | | | | |
| H'FFFFFE15*[1] | OCRA L | | | | | | | | | |
| | OCRB L | | | | | | | | | |
| H'FFFFFE16 | TCR | IEDG | — | — | — | — | — | CKS1 | CKS0 | |
| H'FFFFFE17 | TOCR | — | — | — | OCRS | — | — | OLVLA | OLVLB | |
| H'FFFFFE18 | FICR H | | | | | | | | | |
| H'FFFFFE19 | FICR L | | | | | | | | | |
| H'FFFFFE1A to H'FFFFFE3F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE40 | IPRD | TPU0IP3 | TPU0IP2 | TPU0IP1 | TPU0IP0 | TPU1IP3 | TPU1IP2 | TPU1IP1 | TPU1IP0 | INTC |
| H'FFFFFE41 | | TPU2IP3 | TPU2IP2 | TPU2IP1 | TPU2IP0 | SCF1IP3 | SCF1IP2 | SCF1IP1 | SCF1IP0 | |
| H'FFFFFE42 | VCRE | — | TG0AV6 | TG0AV5 | TG0AV4 | TG0AV3 | TG0AV2 | TG0AV1 | TG0AV0 | |
| H'FFFFFE43 | | — | TG0BV6 | TG0BV5 | TG0BV4 | TG0BV3 | TG0BV2 | TG0BV1 | TG0BV0 | |
| H'FFFFFE44 | VCRF | — | TG0CV6 | TG0CV5 | TG0CV4 | TG0CV3 | TG0CV2 | TG0CV1 | TG0CV0 | |
| H'FFFFFE45 | | — | TG0DV6 | TG0DV5 | TG0DV4 | TG0DV3 | TG0DV2 | TG0DV1 | TG0DV0 | |
| H'FFFFFE46 | VCRG | — | TC0VV6 | TC0VV5 | TC0VV4 | TC0VV3 | TC0VV2 | TC0VV1 | TC0VV0 | |
| H'FFFFFE47 | | — | — | — | — | — | — | — | — | |
| H'FFFFFE48 | VCRH | — | TG1AV6 | TG1AV5 | TG1AV4 | TG1AV3 | TG1AV2 | TG1AV1 | TG1AV0 | |
| H'FFFFFE49 | | — | TG1BV6 | TG1BV5 | TG1BV4 | TG1BV3 | TG1BV2 | TG1BV1 | TG1BV0 | |
| H'FFFFFE4A | VCRI | — | TC1VV6 | TC1VV5 | TC1VV4 | TC1VV3 | TC1VV2 | TC1VV1 | TC1VV0 | |
| H'FFFFFE4B | | — | TC1UV6 | TC1UV5 | TC1UV4 | TC1UV3 | TC1UV2 | TC1UV1 | TC1UV0 | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFE4C | VCRJ | — | TG2AV6 | TG2AV5 | TG2AV4 | TG2AV3 | TG2AV2 | TG2AV1 | TG2AV0 | INTC |
| H'FFFFFE4D | | — | TG2BV6 | TG2BV5 | TG2BV4 | TG2BV3 | TG2BV2 | TG2BV1 | TG2BV0 | |
| H'FFFFFE4E | VCRK | — | TC2VV6 | TC2VV5 | TC2VV4 | TC2VV3 | TC2VV2 | TC2VV1 | TC2VV0 | |
| H'FFFFFE4F | | — | TC2UV6 | TC2UV 5 | TC2UV4 | TC2UV3 | TC2UV2 | TC2UV1 | TC2UV0 | |
| H'FFFFFE50 | VCRL | — | SER1V6 | SER1V5 | SER1V4 | SER1V3 | SER1V2 | SER1V1 | SER1V0 | |
| H'FFFFFE51 | | — | SRX1V6 | SRX1V5 | SRX1V4 | SRX1V3 | SRX1V2 | SRX1V1 | SER1V0 | |
| H'FFFFFE52 | VCRM | — | SBR1V6 | SBR1V5 | SBR1V4 | SBR1V3 | SBR1V2 | SBR1V1 | SBR1V0 | |
| H'FFFFFE53 | | — | STX1V6 | STX1V5 | STX1V4 | STX1V3 | STX1V2 | STX1V1 | STX1V0 | |
| H'FFFFFE54 | VCRN | — | SER2V6 | SER2V5 | SER2V4 | SER2V3 | SER2V2 | SER2V1 | SER2V0 | |
| H'FFFFFE55 | | — | SRX2V6 | SRX2V5 | SRX2V4 | SRX2V3 | SRX2V2 | SRX2V1 | SRX2V0 | |
| H'FFFFFE56 | VCRO | — | SBR2V6 | SBR2V5 | SBR2V4 | SBR2V3 | SBR2V2 | SBR2V1 | SBR2V0 | |
| H'FFFFFE57 | | — | STX2V6 | STX2V5 | STX2V4 | STX2V3 | STX2V2 | STX2V1 | STX2V0 | |
| H'FFFFFE58 to H'FFFFFE5F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE60 | IPRB | SCIIP3 | SCIIP2 | SCIIP1 | SCIIP0 | FRTIP3 | FRTIP2 | FRTIP1 | FRTIP0 | INTC |
| H'FFFFFE61 | | — | — | — | — | — | — | — | — | |
| H'FFFFFE62 | VCRA | — | SERV6 | SERV5 | SERV4 | SERV3 | SERV2 | SERV1 | SERV0 | |
| H'FFFFFE63 | | — | SRXV6 | SRXV5 | SRXV4 | SRXV3 | SRXV2 | SRXV1 | SRXV0 | |
| H'FFFFFE64 | VCRB | — | STXV6 | STXV5 | STXV4 | STXV3 | STXV2 | STXV1 | STXV0 | |
| H'FFFFFE65 | | — | STEV6 | STEV5 | STEV4 | STEV3 | STEV2 | STEV1 | STEV0 | |
| H'FFFFFE66 | VCRC | — | FICV6 | FICV5 | FICV4 | FICV3 | FICV2 | FICV1 | FICV0 | |
| H'FFFFFE67 | | — | FOCV6 | FOCV5 | FOCV4 | FOCV3 | FOCV2 | FOCV1 | FOCV0 | |
| H'FFFFFE68 | VCRD | — | FOVV6 | FOVV5 | FOVV4 | FOVV3 | FOVV2 | FOVV1 | FOVV0 | |
| H'FFFFFE69 | | — | — | — | — | — | — | — | — | |
| H'FFFFFE6A to H'FFFFFE70 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE71 | DRCR0 | — | — | — | RS4 | RS3 | RS2 | RS1 | RS0 | DMAC |
| H'FFFFFE72 | DRCR1 | — | — | — | RS4 | RS3 | RS2 | RS1 | RS0 | |
| H'FFFFFE73 to H'FFFFFE7F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE80*2 | WTCSR | OVF | WT/$\overline{IT}$ | TME | — | — | CKS2 | CKS1 | CKS0 | WDT |
| H'FFFFFE81*2 | WTCNT | | | | | | | | | |
| H'FFFFFE82*2 | — | — | — | — | — | — | — | — | — | |
| H'FFFFFE83*2 | RSTCSR | WOVF | RSTE | RSTS | — | — | — | — | — | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFE84 to H'FFFFFE8F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFE90 | FMR | PLL2ST | PLL1ST | CKIOST | — | FR3 | FR2 | FR1 | FR0 | On-chip oscillation circuit |
| H'FFFFFE91 | SBYCR1 | SBY | HIZ | MSTP5 UBC | MSTP4 DMAC | MSTP3 DSP | MSTP2 DIVU | MSTP1 FRT | MSTP0 SCI | Power-down state |
| H'FFFFFE92 | CCR | W1 | W0 | WB | CP | TW | OD | ID | CE | CACHE |
| H'FFFFFE93 | SBYCR2 | — | — | MSTP11 TPU | MSTP10 SIO2 | MSTP9 SIO1 | MSTP8 SIO0 | MSTP7 SCIF2 | MSTP6 SCIF1 | Power-down state |
| H'FFFFFE94 to H'FFFFFEBF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFEC0 | IPRE | SCF2IP3 | SCF2IP2 | SCF2IP1 | SCF2IP0 | SIO0IP3 | SIO0IP2 | SIO0IP1 | SIO0IP0 | INTC |
| H'FFFFFEC1 | | SIO1IP3 | SIO1P2 | SIO2P1 | SIO1IP0 | SIO2IP3 | SIO2IP2 | SIO2IP1 | SIO2IP0 | |
| H'FFFFFEC2 | VCRP | — | RER0V6 | RER0V5 | RER0V4 | RER0V3 | RER0V2 | RER0V1 | RER0V0 | |
| H'FFFFFEC3 | | — | TER0V6 | TER0V5 | TER0V4 | TER0V3 | TER0V2 | TER0V1 | TER0V0 | |
| H'FFFFFEC4 | VCRQ | — | RDF0V6 | RDF0V5 | RDF0V4 | RDF0V3 | RDF0V2 | RDF0V1 | RDF0V0 | |
| H'FFFFFEC5 | | — | TDE0V6 | TDE0V5 | TDE0V4 | TDE0V3 | TDE0V2 | TDE0V1 | TDE0V0 | |
| H'FFFFFEC6 | VCRR | — | RER1V6 | RER1V5 | RER1V4 | RER1V3 | RER1V2 | RER1V1 | RER1V0 | |
| H'FFFFFEC7 | | — | TER1V6 | TER1V5 | TER1V4 | TER1V3 | TER1V2 | TER1V1 | TER1V0 | |
| H'FFFFFEC8 | VCRS | — | RDF1V6 | RDF1V5 | RDF1V4 | RDF1V3 | RDF1V2 | RDF1V1 | RDF1V0 | |
| H'FFFFFEC9 | | — | TDE1V6 | TDE1V5 | TDE1V4 | TDE1V3 | TDE1V2 | TDE1V1 | TDE1V0 | |
| H'FFFFFECA | VCRT | — | RER2V6 | RER2V5 | RER2V4 | RER2V3 | RER2V2 | RER2V1 | RER2V0 | |
| H'FFFFFECB | | — | TER2V6 | TER2V5 | TER2V4 | TER2V3 | TER2V2 | TER2V1 | TER2V0 | |
| H'FFFFFECC | VCRU | — | RDF2V6 | RDF2V5 | RDF2V4 | RDF2V3 | RDF2V2 | RDF2V1 | RDF2V0 | |
| H'FFFFFECD | | — | TDE2V6 | TDE2V5 | TDE2V4 | TDE2V3 | TDE2V2 | TDE2V1 | TDE2V0 | |
| H'FFFFFECE to H'FFFFFEDF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFEE0 | ICR | NMIL | — | — | — | — | — | — | NMIE | INTC |
| H'FFFFFEE1 | | — | — | — | — | — | — | EXIMD | VECMD | |
| H'FFFFFEE2 | IPRA | DIVUIP3 | DIVUIP2 | DIVUIP1 | DIVUIP0 | DMACIP3 | DMACIP2 | DMACIP1 | DMACIP0 | |
| H'FFFFFEE3 | | WDTIP3 | WDTIP2 | WDTIP1 | WDTIP0 | — | — | — | — | |
| H'FFFFFEE4 | VCRWDT | — | WITV6 | WITV5 | WITV4 | WITV3 | WITV2 | WITV1 | WITV0 | |
| H'FFFFFEE5 | | — | BCMV6 | BCMV5 | BCMV4 | BCMV3 | BCMV2 | BCMV1 | BCMV0 | |
| H'FFFFFEE6 | IPRC | IRQ0IP3 | IRQ0IP2 | IRQ0IP1 | IRQ0IP0 | IRQ1IP3 | IRQ1IP2 | IRQ1IP1 | IRQ1IP0 | |
| H'FFFFFEE7 | | IRQ2IP3 | IRQ2IP2 | IRQ2IP1 | IRQ2IP0 | IRQ3IP3 | IRQ3IP2 | IRQ3IP1 | IRQ3IP0 | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFEE8 | IRQCSR | IRQ31S | IRQ30S | IRQ21S | IRQ20S | IRQ11S | IRQ10S | IRQ01S | IRQ00S | INTC |
| H'FFFFFEE9 | | IRL3PS | IRL2PS | IRL1PS | IRL0PS | IRQ3F | IRQ2F | IRQ1F | IRQ0F | |
| H'FFFFFEEA to H'FFFFFEFF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF00 | DVSR | | | | | | | | | DIVU |
| H'FFFFFF01 | | | | | | | | | | |
| H'FFFFFF02 | | | | | | | | | | |
| H'FFFFFF03 | | | | | | | | | | |
| H'FFFFFF04 | DVDNT | | | | | | | | | |
| H'FFFFFF05 | | | | | | | | | | |
| H'FFFFFF06 | | | | | | | | | | |
| H'FFFFFF07 | | | | | | | | | | |
| H'FFFFFF08 | DVCR | — | — | — | — | — | — | — | — | |
| H'FFFFFF09 | | — | — | — | — | — | — | — | — | |
| H'FFFFFF0A | | — | — | — | — | — | — | — | — | |
| H'FFFFFF0B | | — | — | — | — | — | — | OVFIE | OVF | |
| H'FFFFFF0C | VCRDIV | — | — | — | — | — | — | — | — | |
| H'FFFFFF0D | | — | — | — | — | — | — | — | — | |
| H'FFFFFF0E | | | | | | | | | | |
| H'FFFFFF0F | | | | | | | | | | |
| H'FFFFFF10 | DVDNTH | | | | | | | | | |
| H'FFFFFF11 | | | | | | | | | | |
| H'FFFFFF12 | | | | | | | | | | |
| H'FFFFFF13 | | | | | | | | | | |
| H'FFFFFF14 | DVDNTL | | | | | | | | | |
| H'FFFFFF15 | | | | | | | | | | |
| H'FFFFFF16 | | | | | | | | | | |
| H'FFFFFF17 | | | | | | | | | | |
| H'FFFFFF18 to H'FFFFFF3F | | — | — | — | — | — | — | — | — | — |
| H'FFFFFF40 | BARAH | BAA31 | BAA30 | BAA29 | BAA28 | BAA27 | BAA26 | BAA25 | BAA24 | UBC |
| H'FFFFFF41 | | BAA23 | BAA22 | BAA21 | BAA20 | BAA19 | BAA18 | BAA17 | BAA16 | |
| H'FFFFFF42 | BARAL | BAA15 | BAA14 | BAA13 | BAA12 | BAA11 | BAA10 | BAA9 | BAA8 | |
| H'FFFFFF43 | | BAA7 | BAA6 | BAA5 | BAA4 | BAA3 | BAA2 | BAA1 | BAA0 | |
| H'FFFFFF44 | BAMRAH | BAMA31 | BAMA30 | BAMA29 | BAMA28 | BAMA27 | BAMA26 | BAMA25 | BAMA24 | |
| H'FFFFFF45 | | BAMA23 | BAMA22 | BAMA21 | BAMA20 | BAMA19 | BAMA18 | BAMA17 | BAMA16 | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFF46 | BAMRAL | BAMA15 | BAMA14 | BAMA13 | BAMA12 | BAMA11 | BAMA10 | BAMA9 | BAMA8 | UBC |
| H'FFFFFF47 | | BAMA7 | BAMA6 | BAMA5 | BAMA4 | BAMA3 | BAMA2 | BAMA1 | BAMA0 | |
| H'FFFFFF48 | BBRA | — | — | — | — | — | — | — | — | |
| H'FFFFFF49 | | CPA1 | CPA0 | IDA1 | IDA0 | RWA1 | RWA0 | SZA1 | SZA0 | |
| H'FFFFFF4A to H'FFFFFF4B | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF4C | BETR | — | — | — | — | | | | | UBC |
| H'FFFFFF4D | | | | | | | | | | |
| H'FFFFFF4E to H'FFFFFF4F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF50 | BRFR | SVF | PID2 | PID1 | PID0 | — | — | — | — | UBC |
| H'FFFFFF51 | | DVF | — | — | — | — | — | — | — | |
| H'FFFFFF52 to H'FFFFFF53 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF54 | BRSR | BSA31 | BSA30 | BSA29 | BSA28 | BSA27 | BSA26 | BSA25 | BSA24 | UBC |
| H'FFFFFF55 | | BSA23 | BSA22 | BSA21 | BSA20 | BSA19 | BSA18 | BSA17 | BSA16 | |
| H'FFFFFF56 | | BSA15 | BSA14 | BSA13 | BSA12 | BSA11 | BSA10 | BSA9 | BSA8 | |
| H'FFFFFF57 | | BSA7 | BSA6 | BSA5 | BSA4 | BSA3 | BSA2 | BSA1 | BSA0 | |
| H'FFFFFF58 | BRDR | BDA31 | BDA30 | BDA29 | BDA28 | BDA27 | BDA26 | BDA25 | BDA24 | |
| H'FFFFFF59 | | BDA23 | BDA22 | BDA21 | BDA20 | BDA19 | BDA18 | BDA17 | BDA16 | |
| H'FFFFFF5A | | BDA15 | BDA14 | BDA13 | BDA12 | BDA11 | BDA10 | BDA9 | BDA8 | |
| H'FFFFFF5B | | BDA7 | BDA6 | BDA5 | BDA4 | BDA3 | BDA2 | BDA1 | BDA0 | |
| H'FFFFFF5C to H'FFFFFF5F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF60 | BARBH | BAB31 | BAB30 | BAB29 | BAB28 | BAB27 | BAB26 | BAB25 | BAB24 | UBC |
| H'FFFFFF61 | | BAB23 | BAB22 | BAB21 | BAB20 | BAB19 | BAB18 | BAB17 | BAB16 | |
| H'FFFFFF62 | BARBL | BAB15 | BAB14 | BAB13 | BAB12 | BAB11 | BAB10 | BAB9 | BAB8 | |
| H'FFFFFF63 | | BAB7 | BAB6 | BAB5 | BAB4 | BAB3 | BAB2 | BAB1 | BAB0 | |
| H'FFFFFF64 | BAMRBH | BAMB31 | BAMB30 | BAMB29 | BAMB28 | BAMB27 | BAMB26 | BAMB25 | BAMB24 | |
| H'FFFFFF65 | | BAMB23 | BAMB22 | BAMB21 | BAMB20 | BAMB19 | BAMB18 | BAMB17 | BAMB16 | |
| H'FFFFFF66 | BAMRBL | BAMB15 | BAMB14 | BAMB13 | BAMB12 | BAMB11 | BAMB10 | BAMB9 | BAMB8 | |
| H'FFFFFF67 | | BAMB7 | BAMB6 | BAMB5 | BAMB4 | BAMB3 | BAMB2 | BAMB1 | BAMB0 | |
| H'FFFFFF68 | BBRB | — | — | — | — | — | — | XYE | XYS | |
| H'FFFFFF69 | | CPB1 | CPB0 | IDB1 | IDB0 | RWB1 | RWB0 | SZB1 | SZB0 | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFF6A to H'FFFFFF6F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF70 | BDRBH | BDB31 | BDB30 | BDB29 | BDB28 | BDB27 | BDB26 | BDB25 | BDB24 | UBC |
| H'FFFFFF71 | | BDB23 | BDB22 | BDB21 | BDB20 | BDB19 | BDB18 | BDB17 | BDB16 | |
| H'FFFFFF72 | BDRBL | BDB15 | BDB14 | BDB13 | BDB12 | BDB11 | BDB10 | BDB9 | BDB8 | |
| H'FFFFFF73 | | BDB7 | BDB6 | BDB5 | BDB4 | BDB3 | BDB2 | BDB1 | BDB0 | |
| H'FFFFFF74 | BDMRBH | BDMB31 | BDMB30 | BDMB29 | BDMB28 | BDMB27 | BDMB26 | BDMB25 | BDMB24 | |
| H'FFFFFF75 | | BDMB23 | BDMB22 | BDMB21 | BDMB20 | BDMB19 | BDMB18 | BDMB17 | BDMB16 | |
| H'FFFFFF76 | BDMRBL | BDMB15 | BDMB14 | BDMB13 | BDMB12 | BDMB11 | BDMB10 | BDMB9 | BDMB8 | |
| H'FFFFFF77 | | BDMB7 | BDMB6 | BDMB5 | BDMB4 | BDMB3 | BDMB2 | BDMB1 | BDMB0 | |
| H'FFFFFF78 | BRCR | CMFCA | CMFPA | EBBE | UMD | PCTE | PCBA | — | — | |
| H'FFFFFF79 | | CMFCB | CMFPB | ETBE | SEQ | DBEB | PCBB | — | — | |
| H'FFFFFF7A to H'FFFFFF7F | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFF80 | SAR0 | | | | | | | | | DMAC |
| H'FFFFFF81 | | | | | | | | | | |
| H'FFFFFF82 | | | | | | | | | | |
| H'FFFFFF83 | | | | | | | | | | |
| H'FFFFFF84 | DAR0 | | | | | | | | | |
| H'FFFFFF85 | | | | | | | | | | |
| H'FFFFFF86 | | | | | | | | | | |
| H'FFFFFF87 | | | | | | | | | | |
| H'FFFFFF88 | TCR0 | — | — | — | — | — | — | — | — | |
| H'FFFFFF89 | | | | | | | | | | |
| H'FFFFFF8A | | | | | | | | | | |
| H'FFFFFF8B | | | | | | | | | | |
| H'FFFFFF8C | CHCR0 | — | — | — | — | — | — | — | — | |
| H'FFFFFF8D | | — | — | — | — | — | — | — | — | |
| H'FFFFFF8E | | DM1 | DM0 | SM1 | SM0 | TS1 | TS0 | AR | AM | |
| H'FFFFFF8F | | AL | DS | DL | TB | TA | IE | TE | DE | |
| H'FFFFFF90 | SAR1 | | | | | | | | | |
| H'FFFFFF91 | | | | | | | | | | |
| H'FFFFFF92 | | | | | | | | | | |
| H'FFFFFF93 | | | | | | | | | | |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFF94 | DAR1 | | | | | | | | | DMAC |
| H'FFFFFF95 | | | | | | | | | | |
| H'FFFFFF96 | | | | | | | | | | |
| H'FFFFFF97 | | | | | | | | | | |
| H'FFFFFF98 | TCR1 | — | — | — | — | — | — | — | — | |
| H'FFFFFF99 | | | | | | | | | | |
| H'FFFFFF9A | | | | | | | | | | |
| H'FFFFFF9B | | | | | | | | | | |
| H'FFFFFF9C | CHCR1 | — | — | — | — | — | — | — | — | |
| H'FFFFFF9D | | — | — | — | — | — | — | — | — | |
| H'FFFFFF9E | | DM1 | DM0 | SM1 | SM0 | TS1 | TS0 | AR | AM | |
| H'FFFFFF9F | | AL | DS | DL | TB | TA | IE | TE | DE | |
| H'FFFFFFA0 | VCRDMA0 | — | — | — | — | — | — | — | — | |
| H'FFFFFFA1 | | — | — | — | — | — | — | — | — | |
| H'FFFFFFA2 | | — | — | — | — | — | — | — | — | |
| H'FFFFFFA3 | | VC7 | VC6 | VC5 | VC4 | VC3 | VC2 | VC1 | VC0 | |
| H'FFFFFFA4 to H'FFFFFFA7 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFA8 | VCRDMA1 | — | — | — | — | — | — | — | — | DMAC |
| H'FFFFFFA9 | | — | — | — | — | — | — | — | — | |
| H'FFFFFFAA | | — | — | — | — | — | — | — | — | |
| H'FFFFFFAB | | VC7 | VC6 | VC5 | VC4 | VC3 | VC2 | VC1 | VC0 | |
| H'FFFFFFAC to H'FFFFFFAF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFB0 | DMAOR | — | — | — | — | — | — | — | — | DMAC |
| H'FFFFFFB1 | | — | — | — | — | — | — | — | — | |
| H'FFFFFFB2 | | — | — | — | — | — | — | — | — | |
| H'FFFFFFB3 | | — | — | — | — | PR | AE | NMIF | DME | |
| H'FFFFFFB4 to H'FFFFFFBF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFC0 | WCR2 | A4WD1 | A4WD0 | — | A4WM | A3WM | A2WM | A1WM | A0WM | BSC |
| H'FFFFFFC1 | | — | — | — | — | IW41 | IW40 | W41 | W40 | |
| H'FFFFFFC2 to H'FFFFFFC3 | — | — | — | — | — | — | — | — | — | — |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFFC4 | WCR3 | — | — | A4SW2 | A4SW1 | A4SW0 | — | A4HW1 | A4HW0 | BSC |
| H'FFFFFFC5 | | A3SHW1 | A3SHW0 | A2SHW1 | A2SHW0 | A1SHW1 | A1SHW0 | A0SHW1 | A0SHW0 | |
| H'FFFFFFC6 to H'FFFFFFDF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFE0 | BCR1 | — | A4LW1 | A4LW0 | A2ENDIAN | BSTROM | — | AHLW1 | AHLW0 | BSC |
| H'FFFFFFE1 | | A1LW1 | A1LW0 | A0LW1 | A0LW0 | A4ENDIAN | DRAM2 | DRAM1 | DRAM0 | |
| H'FFFFFFE2 to H'FFFFFFE3 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFE4 | BCR2 | — | — | — | — | — | — | A4SZ1 | A4SZ0 | BSC |
| H'FFFFFFE5 | | A3SZ1 | A3SZ0 | A2SZ1 | A2SZ0 | A1SZ1 | A1SZ0 | — | — | |
| H'FFFFFFE6 to H'FFFFFFE7 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFE8 | WCR1 | IW31 | IW30 | IW21 | IW20 | IW11 | IW10 | IW01 | IW00 | BSC |
| H'FFFFFFE9 | | W31 | W30 | W21 | W20 | W11 | W10 | W01 | W00 | |
| H'FFFFFFEA to H'FFFFFFEB | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFEC | MCR | TRP0 | RCD0 | TRWL0 | TRAS1 | TRAS0 | BE | RASD | TRWL1 | BSC |
| H'FFFFFFED | | AMX2 | SZ | AMX1 | AMX0 | RFSH | RMD | TRP1 | RCD1 | |
| H'FFFFFFEE to H'FFFFFFEF | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFF0 | RTCSR | — | — | — | — | — | — | — | — | BSC |
| H'FFFFFFF1 | | CMF | CMIE | CKS2 | CKS1 | CKS0 | RRC2 | RRC1 | RRC0 | |
| H'FFFFFFF2 to H'FFFFFFF3 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFF4 | RTCNT | — | — | — | — | — | — | — | — | BSC |
| H'FFFFFFF5 | | | | | | | | | | |
| H'FFFFFFF6 to H'FFFFFFF7 | — | — | — | — | — | — | — | — | — | — |
| H'FFFFFFF8 | RTCOR | — | — | — | — | — | — | — | — | BSC |
| H'FFFFFFF9 | | | | | | | | | | |
| H'FFFFFFFA to H'FFFFFFFB | — | — | — | — | — | — | — | — | — | — |

| Address | Register Name | Bit Names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'FFFFFFFC | BCR3 | — | — | — | — | A4LW2 | AHLW2 | A1LW2 | A0LW2 | BSC |
| H'FFFFFFFD | | DSWW1 | DSWW0 | — | — | — | BASEL | EDO | BWE | |
| H'FFFFFFFE to H'FFFFFFFF | — | — | — | — | — | — | — | — | — | — |

Notes: 1. OCRA and OCRB have the same address. Switching between the two is performed by means of the OCRS bit in TOCR. For details, see section 11, 16-Bit Free-Running Timer (FRT).

2. This is the read address. The write address is H'FFFFFE80 for WTCSR and WTCNT, and H'FFFFFE82 for RSTCSR. For details, see section 12, Watchdog Timer (WDT).

# Appendix B Pin States

## B.1 Pin States in Reset, Power-Down State, and Bus-Released State

| | | Pin State | | | | | |
|---|---|---|---|---|---|---|---|
| | | Power-On Reset | Manual Reset | | Power-Down State | | Bus-Released State |
| Pin Type | Pin Name | | Bus Acquired | Bus Released | Standby Mode | Sleep Mode | |
| Clock | CKIO | I/O*1 | I/O*1 | I/O*1 | I/O*1 | I/O*1 | I/O*1 |
| | EXTAL | I*1 | I*1 | I*1 | I*1 | I*1 | I*1 |
| | XTAL | O*1 | O*1 | O*1 | O*1 | O*1 | O*1 |
| | $\overline{\text{CKPREQ}}$/CKM | I | I | I | I | I | I |
| | $\overline{\text{CKPACK}}$ | H | H | H | H*1, *2 | H | H |
| System control | $\overline{\text{RES}}$ | I | I | I | I | I | I |
| | $\overline{\text{WDTOVF}}$/PA7 | H | H / I/O | H / I/O | O/K*3 | O / I/O | O / I/O |
| | $\overline{\text{BGR}}$ | H | O | O | H | O | O |
| | $\overline{\text{BRLS}}$ | Z | I | I | Z | I | I |
| | MD4–MD0 | I | I | I | I | I | I |
| Interrupt | NMI | I | I | I | I | I | I |
| | $\overline{\text{IRL3}}$–$\overline{\text{IRL0}}$ | Z | Z | Z | I | I | I |
| | $\overline{\text{IVECF}}$ | H | H | H | H*3 | H | H |
| Address bus | A24–A0 | O | O | Z | Z | O | Z |
| Data bus | D31–D0 | Z | I/O | Z | Z | I/O | Z |
| Bus control | $\overline{\text{CS4}}$–$\overline{\text{CS0}}$ | H | O | Z | H | H | Z |
| | $\overline{\text{BS}}$ | H | O | Z | H | H | Z |
| | RD/$\overline{\text{WR}}$ | H | O | Z | H | H | Z |
| | $\overline{\text{RAS}}$ | H | O | Z | H | H | Z |
| | $\overline{\text{CAS}}$/$\overline{\text{OE}}$ | H | O | Z | H | H | Z |
| | DQMUU/$\overline{\text{WE3}}$ | H | O | Z | H | H | Z |
| | DQMUL/$\overline{\text{WE2}}$ | H | O | Z | H | H | Z |
| | DQMLU/$\overline{\text{WE1}}$ | H | O | Z | H | H | Z |
| | DQMLL/$\overline{\text{WE0}}$ | H | O | Z | H | H | Z |
| | $\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$ | H | O | Z | H | H | Z |
| | $\overline{\text{RD}}$ | H | O | Z | H | H | Z |
| | CKE | H | O | H | O | O | H |
| | $\overline{\text{WAIT}}$ | Z | I | Z | Z | I | Ignored |
| | REFOUT | L | O | O | L*3 | O | O |

| | | Pin State | | | | | |
|---|---|---|---|---|---|---|---|
| | | Power-On Reset | Manual Reset | | Power-Down State | | Bus-Released State |
| Pin Type | Pin Name | | Bus Acquired | Bus Released | Standby Mode | Sleep Mode | |
| Direct memory access controller (DMAC) | DACK1–DACK0 | H | H | H | K*[3] | O | O |
| | DREQ1–DREQ0 | Z | Z | Z | Z | I | I |
| 16-bit free-running timer (FRT) | FTCI/PA6 | Z | Z / I/O | Z / I/O | K*[3]/K*[3] | I / I/O | I / I/O |
| | FTCI/PA5 | Z | Z / I/O | Z / I/O | K*[3]/K*[3] | I / I/O | I / I/O |
| | FTOA/PA4 | L | L / I/O | L / I/O | K*[3]/K*[3] | O / I/O | O / I/O |
| | FTOB/CKPO | L | L / I/O | L / I/O | K*[3]/K*[3] | O / I/O | O / I/O |
| Serial communication interface(SCI) | SCK/PA2 | Z | Z / I/O | Z / I/O | K*[3]/K*[3] | I/O / I/O | I/O / I/O |
| | RxD/PA1 | Z | Z / I/O | Z / I/O | K*[3]/K*[3] | I / I/O | I / I/O |
| | TxD/PA0 | Z | Z / I/O | Z / I/O | K*[3]/K*[3] | O / I/O | O / I/O |
| Serial I/O (SIO)/ serial communication interface (SCI)/ serial communication interface with FIFO (SCIF)/ 16-bit timer pulse unit (TPU) | PB15/SRCK0/ SCK2 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I/O | I/O / I / I/O |
| | PB14/SRS0/ RxD2 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I | I/O / I / I |
| | PB13/SRXD0/ TxD2 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / O | I/O / I / O |
| | PB12/STCK0/ SCK1 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I/O | I/O / I / I/O |
| | PB11/STS0/ RxD1 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I/O / I | I/O / I/O / I |
| | PB10/STXD0/ TxD1 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / O / O | I/O / O / O |
| | PB9/SRCK1/ $\overline{\text{RTS}}$ | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / O | I/O / I / O |
| | PB8/SRS1/ $\overline{\text{CTS}}$ | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I | I/O / I / I |
| | PB7/SRXD1/ TIOCA2 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I/O | I/O / I / I/O |
| | PB6/STCK1/ TIOCB2/ TCLKD | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I / I/O | I/O / I / I/O |
| | PB5/STS1/ TIOCA1 | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / I/O / I/O | I/O / I/O / I/O |
| | PB4/STXD1/ TIOCB1/ TCLKC | Z | I/O / Z / Z | I/O / Z / Z | K*[3]/K*[3]/K*[3] | I/O / O / I/O | I/O / O / I/O |

| | | Pin State | | | | | |
|---|---|---|---|---|---|---|---|
| | | Power-On Reset | Manual Reset | | Power-Down State | | Bus-Released State |
| Pin Type | Pin Name | | Bus Acquired | Bus Released | Standby Mode | Sleep Mode | |
| Serial I/O (SIO)/ serial communication interface (SCI)/ serial communication interface with FIFO (SCIF)/ 16-bit timer pulse unit (TPU) | PB3/SRCK2/ TIOCA0 | Z | I/O / Z / Z | I/O / Z / Z | K*3/K*3/K*3 | I/O / I / I/O | I/O / I / I/O |
| | PB2/SRS2/ TIOCB0 | Z | I/O / Z / Z | I/O / Z / Z | K*3/K*3/K*3 | I/O / I / I/O | I/O / I / I/O |
| | PB1/SRXD2/ TIOCC0/ TCLKA | Z | I/O / Z / Z | I/O / Z / Z | K*3/K*3/K*3 | I/O / I / I/O | I/O / I / I/O |
| | PB0/STCK2/ TIOCD0/ TCLKB | Z | I/O / Z / Z | I/O / Z / Z | K*3/K*3/K*3 | I/O / I / I/O | I/O / I / I/O |
| | PA9/STS2 | Z | I/O / Z | I/O / Z | K*3/K*3 | I/O / I/O | I/O / I/O |
| | PA8/STXD2 | Z | I/O / Z | I/O / Z | K*3/K*3 | I/O / O | I/O / O |
| Hitachi user debug interface (H-UDI) | $\overline{\text{TRST}}$ | I | I | I | I | I | I |
| | TCK/PA13 | I | I / I/O | I / I/O | I / K*3 | I / I/O | I / I/O |
| | TMS/PA12 | I | I / I/O | I / I/O | I / K*3 | I / I/O | I / I/O |
| | TDI/PA11 | I | I / I/O | I / I/O | I / K*3 | I / I/O | I / I/O |
| | TDO/PA10 | O | O / I/O | O / I/O | O / K*3 | O / I/O | O / I/O |

I: Input

O: Output

H: High-level output

L: Low-level output

Z: High-impedance state

K: Input pins are in the high-impedance state; output pins maintain their previous state.

Notes: In sleep mode, if the DMAC is operating the address/data bus and bus control signals vary according to the operation of the DMAC. (The same applies when refreshing is performed.)

1. Depends on the clock mode (MD2–MD0 setting).
2. Low-level output in clock pause standby mode.
3. When the high-impedance bit (HIZ) is set to 1 in the standby control register (SBYCR), output pins go to the high-impedance state.

# Appendix C Product Lineup

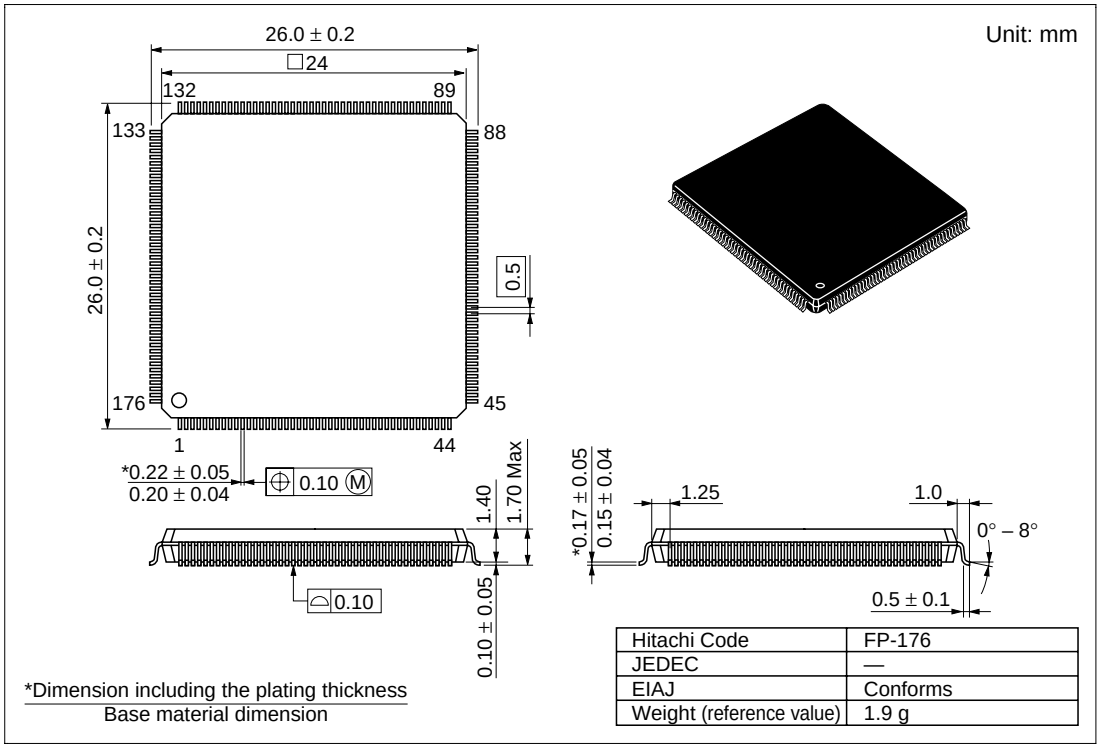
## C.1 Product Lineup

**Table C.1 SH7612 Product Lineup**

| Abbreviation | Voltage | Operating Frequency | Mark Code | Package |
|---|---|---|---|---|
| SH7612 | 3.3V | 60MHz | HD6417612RF60 | FP-176C |
| | | | HD6417612RBP60 | TBP-176 |

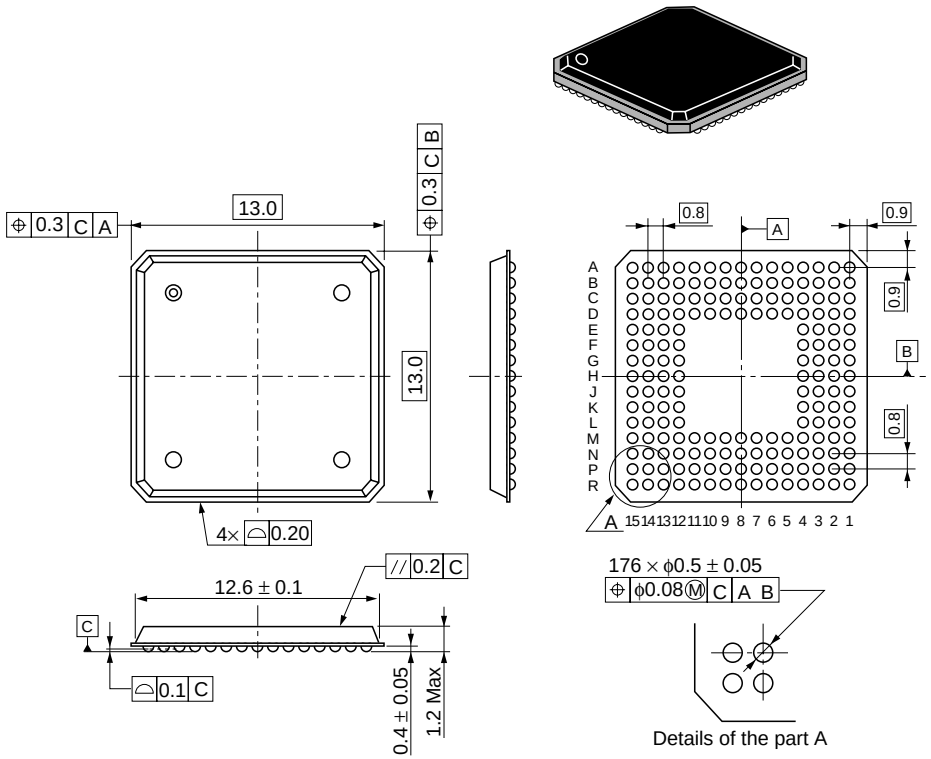
# Appendix D Package Dimensions

Figures D.1 and D.2 show the FP-176 and TBP-176 package dimensions.




| | |
|---|---|
| Hitachi Code | FP-176 |
| JEDEC | — |
| EIAJ | Conforms |
| Weight (reference value) | 1.9 g |

**Figure D.1 Package Dimensions (FP-176)**




| | |
|---|---|
| Hitachi Code | TBP-176 |
| JEDEC | — |
| EIAJ | — |
| Weight (reference value) | 0.32 g |

**Figure D.2 Package Dimensions (TBP-176)**

**SH7612 Hardware Manual**

Publication Date: 1st Edition, September 1999
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
UL Media Co., Ltd.
Copyright © Hitachi, Ltd., 1999. All rights reserved. Printed in Japan.