User's Manual

32

# RX610 Group

## User's Manual: Hardware

RENESAS 32-Bit MCU
RX Family / RX600 Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.20 Feb 2013

Downloaded from Arrow.com.

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(2012 4)

Downloaded from Arrow.com.

## General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Downloaded from Arrow.com.

# How to Use This Manual

## 1. Objective and Target Users

This manual was written to explain the hardware functions and electrical characteristics of this LSI to the target users, i.e. those who will be using this LSI in the design of application systems. Target users are expected to understand the fundamentals of electrical circuits, logic circuits, and microcomputers.
This manual is organized in the following items: an overview of the product, descriptions of the CPU, system control functions, and peripheral functions, electrical characteristics of the device, and usage notes.

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

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

The following documents have been prepared for the RX610 Group. Before using any of the documents, please visit our web site to verify that you have the most up-to-date available version of the document.

| Document Type | Contents | Document Title | Document No. |
|---|---|---|---|
| Short Sheet | Overview of hardware | — | — |
| Data Sheet | Overview of hardware and electrical characteristics | RX610 Group Data Sheet | — |
| User's manual: Hardware | Hardware specifications (pin assignments, memory maps, peripheral specifications, electrical characteristics, and timing charts) and descriptions of operation | RX610 Group User's manual: Hardware | This User's manual |
| User's manual: Software | Detailed descriptions of the CPU and instruction set | RX Family Series User's manual: Software | REJ09B0435 |
| Application Note | Examples of applications and sample programs | — | — |
| Renesas Technical Update | Preliminary report on the specifications of a product, document, etc. | — | — |

Downloaded from Arrow.com.

## 2. Description of Registers

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

### X.X.X ... Register

Address xxxx xxxxxh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | ... | ... | ... | — | — | — | ... |
| Value after reset | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ... 0 | ... Bit | 0: .....<br>1: Setting prohibited (3) | R/W (1) |
| b3 to b1 | — | Reserved (2) | The read value is 0. The write value should always be 0. | R/W |
| b4 | ... 4 | ... Bit | 0: .....<br>1: ..... | R |
| b6, b5 | ... [1:0] | ... Bi | 0 0: .....<br>0 1: .....<br>Settings other than above are prohibited. (3) | R/(W)* |
| b7 | — | Reserved | The read value is undefined. Writing to this bit has no effect. | R |

(1) R/W: The bit or field is readable and writable.
R/(W): The bit or field is readable and writable. However, writing to this bit or field has some limitations. For details on the limitations, see the description or notes of respective registers.
R: The bit or field is readable. Writing to this bit or field has no effect.

(2) Reserved. Make sure to use the specified value when writing to this bit or field; otherwise, the correct operation is not guaranteed.

(3) Setting prohibited. The correct operation is not guaranteed if such a setting is performed.

Downloaded from Arrow.com.

# 3. List of Abbreviations and Acronyms

| Abbreviation | Full Form |
|---|---|
| ACIA | Asynchronous Communication Interface Adapter |
| bps | bits per second |
| CRC | Cyclic Redundancy Check |
| DMA | Direct Memory Access |
| DMAC | Direct Memory Access Controller |
| GSM | Global System for Mobile Communications |
| Hi-Z | High Impedance |
| IEBus | Inter Equipment Bus |
| I/O | Input / Output |
| IrDA | Infrared Data Association |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| NC | Non-Connect |
| PLL | Phase Locked Loop |
| PWM | Pulse Width Modulation |
| SIM | Subscriber Identity Module |
| UART | Universal Asynchronous Receiver / Transmitter |
| VCO | Voltage Controlled Oscillator |

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

Downloaded from Arrow.com.

# Contents

Contents ..... 7

1. Overview ..... 28
1.1 Features ..... 28
1.1.1 Applications ..... 28
1.1.2 Outline of Specifications ..... 29
1.2 List of Products ..... 32
1.3 Block Diagram ..... 34
1.4 Pin Assignments ..... 35
1.5 Pin Functions ..... 49

2. CPU ..... 54
2.1 Features ..... 54
2.2 Register Set of the CPU ..... 55
2.2.1 General-Purpose Registers (R0 to R15) ..... 56
2.2.2 Control Registers ..... 56
2.2.2.1 Interrupt Stack Pointer (ISP)/User Stack Pointer (USP) ..... 57
2.2.2.2 Interrupt Table Register (INTB) ..... 57
2.2.2.3 Program Counter (PC) ..... 57
2.2.2.4 Processor Status Word (PSW) ..... 58
2.2.2.5 Backup PC (BPC) ..... 60
2.2.2.6 Backup PSW (BPSW) ..... 60
2.2.2.7 Fast Interrupt Vector Register (FINTV) ..... 60
2.2.2.8 Floating-Point Status Word (FPSW) ..... 61
2.2.2.9 Accumulator (ACC) ..... 64
2.3 Processor Mode ..... 65
2.3.1 Supervisor Mode ..... 65
2.3.2 User Mode ..... 65
2.3.3 Privileged Instruction ..... 65
2.3.4 Switching Between Processor Modes ..... 65
2.4 Data Types ..... 66
2.4.1 Integer ..... 66
2.4.2 Floating-Point ..... 66
2.4.3 Bitwise Operations ..... 67
2.4.4 Strings ..... 67
2.5 Endian ..... 68
2.5.1 Switching the Endian ..... 68
2.5.2 Access to I/O Registers ..... 72
2.5.3 Notes on Access to I/O Registers ..... 72
2.5.4 Data Arrangement ..... 73

Downloaded from Arrow.com.

2.5.4.1 Data Arrangement in Registers ... 73
2.5.4.2 Data Arrangement in Memory ... 73
2.5.5 Notes on Arrangement of Instruction Code ... 73
2.6 Vector Table ... 74
2.6.1 Fixed Vector Table ... 74
2.6.2 Relocatable Vector Table ... 75
2.7 Operation of Instructions ... 76
2.7.1 Data Prefetching by the RMPA Instruction and the String-Manipulation Instructions ... 76
2.8 Pipeline ... 77
2.8.1 Overview ... 77
2.8.2 Instructions and Pipeline Processing ... 79
2.8.2.1 Instructions Converted into Single Micro-Operation and Pipeline Processing ... 79
2.8.2.2 Instructions Converted into Multiple Micro-Operations and Pipeline Processing ... 81
2.8.2.3 Pipeline Basic Operation ... 84
2.8.3 Calculation of the Instruction Processing Time ... 86
2.8.4 Numbers of Cycles for Response to Interrupts ... 87

3. Operating Modes ... 88
3.1 Operating Mode Types and Selection ... 88
3.2 Register Descriptions ... 89
3.2.1 Mode Monitor Register (MDMONR) ... 89
3.2.2 Mode Status Register (MDSR) ... 90
3.2.3 System Control Register 0 (SYSCR0) ... 91
3.2.4 System Control Register 1 (SYSCR1) ... 93
3.3 Details of Operating Modes ... 94
3.3.1 Single-Chip Mode ... 94
3.3.2 On-Chip ROM Enabled Extended Mode ... 94
3.3.3 On-Chip ROM Disabled Extended Mode ... 94
3.3.4 Boot Mode ... 94
3.3.5 User Boot Mode ... 94
3.4 Transitions of Operating Modes ... 95
3.4.1 Operating Mode Transitions According to Mode Pin Setting ... 95
3.4.2 Operating Mode Transitions According to Register Setting ... 96

4. Address Space ... 97
4.1 Address Space ... 97
4.2 External Address Space ... 101

5. I/O Registers ... 102
5.1 I/O Register Addresses (Address Order) ... 104
5.2 I/O Register Bits ... 125

Downloaded from Arrow.com.

6. Resets ..... 154
6.1 Overview ..... 154
6.2 Register Descriptions ..... 156
6.2.1 Reset Status Register (RSTSR) ..... 156
6.2.2 Reset Control/Status Register (RSTCSR) ..... 157
6.3 Operation ..... 158
6.3.1 Pin Reset ..... 158
6.3.2 Deep Software Standby Reset ..... 158
6.3.3 Watchdog Timer Reset ..... 158
6.4 Determining Reset Generation Source ..... 159
6.5 Usage Notes ..... 159
6.5.1 Notes on Design of Board ..... 159

7. Clock Generation Circuit ..... 160
7.1 Overview ..... 160
7.2 Register Descriptions ..... 161
7.2.1 System Clock Control Register (SCKCR) ..... 162
7.3 Main Clock Oscillator ..... 164
7.3.1 Connecting a Crystal Resonator ..... 164
7.3.2 External Clock Input ..... 165
7.4 PLL Circuit ..... 165
7.5 Frequency Divider ..... 165
7.6 Internal Clock ..... 166
7.6.1 System Clock (ICLK) ..... 166
7.6.2 Peripheral Module Clock (PCLK) ..... 166
7.6.3 External Bus Clock (BCLK) ..... 166
7.7 Usage Notes ..... 167
7.7.1 Notes on the Clock Generation Circuit ..... 167
7.7.2 Notes on Resonator ..... 168
7.7.3 Notes on Board Design ..... 168

8. Low Power Consumption ..... 169
8.1 Overview ..... 169
8.2 Register Descriptions ..... 172
8.2.1 Standby Control Register (SBYCR) ..... 174
8.2.2 Module Stop Control Register A (MSTPCRA) ..... 176
8.2.3 Module Stop Control Register B (MSTPCRB) ..... 178
8.2.4 Module Stop Control Register C (MSTPCRC) ..... 179
8.2.5 Deep Standby Control Register (DPSBYCR) ..... 180
8.2.6 Deep Standby Wait Control Register (DPSWCR) ..... 182
8.2.7 Deep Standby Interrupt Enable Register (DPSIER) ..... 183

Downloaded from Arrow.com.

8.2.8 Deep Standby Interrupt Flag Register (DPSIFR) ... 184
8.2.9 Deep Standby Interrupt Edge Register (DPSIEGR) ... 185
8.2.10 Reset Status Register (RSTSR) ... 186
8.2.11 Deep Standby Backup Register (DPSBKRy) (y = 0 to 31) ... 186
8.3 Multi-Clock Function ... 187
8.4 Module Stop Function ... 187
8.5 Low Power Consumption Modes ... 188
8.5.1 Sleep Mode ... 188
8.5.1.1 Transition to Sleep Mode ... 188
8.5.1.2 Canceling Sleep Mode ... 188
8.5.2 All-Module Clock Stop Mode ... 189
8.5.2.1 Transitions to All-Module Clock Stop Mode ... 189
8.5.2.2 Release from All-Module Clock Stop Mode ... 190
8.5.3 Software Standby Mode ... 191
8.5.3.1 Transition to Software Standby Mode ... 191
8.5.3.2 Canceling Software Standby Mode ... 192
8.5.3.3 Setting Oscillation Settling Time after Software Standby Mode is Canceled ... 193
8.5.3.4 Example of Software Standby Mode Application ... 194
8.5.4 Deep Software Standby Mode ... 195
8.5.4.1 Transition to Deep Software Standby Mode ... 195
8.5.4.2 Canceling Deep Software Standby Mode ... 196
8.5.4.3 Pin States when Deep Software Standby Mode is Canceled ... 196
8.5.4.4 Setting Oscillation Settling Time after Deep Software Standby Mode is Canceled ... 197
8.5.4.5 Example of Deep Software Standby Mode Application ... 198
8.5.4.6 Flowchart to Use Deep Software Standby Mode ... 199
8.6 BCLK Output Control ... 200
8.7 Usage Notes ... 201
8.7.1 I/O Port States ... 201
8.7.2 Module Stop State of the DMAC and DTC ... 201
8.7.3 On-Chip Peripheral Module Interrupts ... 201
8.7.4 Write-Access to MSTPCRA, MSTPCRB, and MSTPCRC ... 201
8.7.5 Input Buffer Control by DIRQnE Bit (n = 3 to 0) ... 201
8.7.6 Conflict between Transition to Deep Software Standby Mode and Interrupt ... 201
8.7.7 Timing of Wait Instructions ... 201

9. Exceptions ... 203

9.1 Types of Exceptions ... 203
9.1.1 Undefined Instruction Exception ... 204
9.1.2 Privileged Instruction Exception ... 204
9.1.3 Floating-Point Exceptions ... 204
9.1.4 Reset ... 204
9.1.5 Non-Maskable Interrupt ... 204

Downloaded from Arrow.com.

9.1.6 Interrupts ..... 204
9.1.7 Unconditional Trap ..... 204
9.2 Exception Handling Procedure ..... 205
9.3 Acceptance of Exceptions ..... 207
9.3.1 Timing of Acceptance and Saved PC Values ..... 207
9.3.2 Vector and Site for Saving the Values in the PC and PSW ..... 208
9.4 Hardware Processing for Accepting and Returning from Exceptions ..... 209
9.5 Hardware Pre-Processing ..... 210
9.5.1 Undefined Instruction Exception ..... 210
9.5.2 Privileged Instruction Exception ..... 210
9.5.3 Floating-Point Exceptions ..... 210
9.5.4 Reset ..... 210
9.5.5 Non-Maskable Interrupt ..... 211
9.5.6 Interrupts ..... 211
9.5.7 Unconditional Trap ..... 211
9.6 Return from Exception Handling Routines ..... 212
9.7 Order of Priority for Exceptions ..... 212

10. Interrupt Control Unit (ICU) ..... 213
10.1 Overview ..... 213
10.2 Register Descriptions ..... 215
10.2.1 Interrupt Request Register i (IRi) (i = interrupt vector number) ..... 223
10.2.2 Interrupt Request Destination Setting Register i (ISELRi) (i = interrupt vector number) ..... 225
10.2.3 Interrupt Request Enable Register m (IERi) (i = 02h to 1Fh) ..... 226
10.2.4 Interrupt Priority Register i (IPRi) (i = 00h to 8Fh) ..... 227
10.2.5 Fast Interrupt Register (FIR) ..... 228
10.2.6 IRQ Detection Enable Register n (IRQERn) (n = 0 to 15) ..... 229
10.2.7 IRQ Control Register n (IRQCRn) (n = 0 to 15) ..... 230
10.2.8 Non-maskable Interrupt Enable Register (NMIER) ..... 231
10.2.9 NMI Pin Interrupt Control Register (NMICR) ..... 232
10.2.10 Non-maskable Interrupt Status Register (NMISR) ..... 233
10.2.11 Non-maskable Interrupt Clear Register (NMICLR) ..... 234
10.2.12 Software Standby Release IRQ Enable Register (SSIER) ..... 235
10.3 Vector Table ..... 236
10.3.1 Interrupt Vector Table ..... 236
10.3.2 Fast Interrupt Vector Address ..... 241
10.3.3 Non-maskable Interrupt Vector Address ..... 241
10.4 Operation ..... 242
10.4.1 Enabling and Disabling Interrupts ..... 242
10.4.2 Interrupt Status Flag ..... 243

Downloaded from Arrow.com.

10.4.2.1 Interrupt Status Flag in Edge Detection ..... 243
10.4.2.2 Interrupt Status Flag in Level Detection ..... 244
10.4.3 Selecting Interrupt Request Destinations ..... 246
10.4.4 Determining Priority ..... 248
10.4.5 Fast Interrupt ..... 248
10.4.6 External Interrupts ..... 249
10.5 Non-maskable Interrupt Operation ..... 250
10.6 Returning from Low Power Consumption Modes ..... 251
10.6.1 Returning from Sleep Mode and All-Module Clock Stop Mode ..... 251
10.6.2 Returning from Software Standby Mode ..... 252
10.7 Usage Notes ..... 253
10.7.1 Notes when writing to the Register of the Interrupt Control Unit ..... 253
10.7.2 Notes on the WAIT Instruction when the NMI Pin Interrupt is Used ..... 253
10.7.3 Notes on Transferring DMAC/DTC Using Communication Function (SCI, RIIC) ..... 253

11. Buses ..... 256
11.1 Overview ..... 256
11.2 Description of Buses ..... 257
11.2.1 CPU Buses ..... 257
11.2.2 Internal Main Buses ..... 257
11.2.3 Internal Peripheral Buses ..... 258
11.2.4 External Bus ..... 258
11.2.5 Parallel Operation ..... 260
11.3 Register Descriptions ..... 261
11.3.1 CSi Control Register (CSiCNT) (i = 0 to 7) ..... 263
11.3.2 CSi Recovery Cycle Register (CSiREC) (i = 0 to 7) ..... 265
11.3.3 CSi Mode Register (CSiMOD) (i = 0 to 7) ..... 267
11.3.4 CSi Wait Control Register 1 (CSiWCNT1) (i = 0 to 7) ..... 269
11.3.5 CSi Wait Control Register 2 (CSiWCNT2) (i = 0 to 7) ..... 273
11.3.6 Bus Error Source Clear Register (BERCLR) ..... 277
11.3.7 Bus Error Monitoring Enable Register (BEREN) ..... 277
11.3.8 Bus Error Interrupt Enable Register (BERIE) ..... 278
11.4 Endian and Data Alignment ..... 279
11.4.1 16-Bit Bus Space ..... 279
11.4.2 8-Bit Bus Space ..... 280
11.5 Operation ..... 283
11.5.1 Timing of External Bus Access ..... 283
11.5.1.1 Normal Access ..... 285
11.5.1.2 Page Access ..... 290
11.5.2 External Wait Function ..... 293
11.5.2.1 Normal Access ..... 293

Downloaded from Arrow.com.

11.5.2.2 Page Access ... 293
11.5.3 Insertion of Recovery Cycles ... 295
11.5.4 Write Buffer Function ... 296
11.5.5 Notes on Usage ... 297
11.5.5.1 Limitations at the Time of Normal and Page Access ... 297
11.5.5.2 Prohibition of Access that Spans Areas of Address Space ... 297
11.5.5.3 Restrictions in Relation to RMPA and String-Manipulation Instructions ... 297
11.5.5.4 Point for Caution Regarding Register Settings ... 297
11.5.5.5 Restriction on Instruction Code ... 298
11.6 Bus Error Monitoring Section ... 299
11.6.1 Types of Bus Error ... 299
11.6.1.1 Illegal Address Access ... 299
11.6.1.2 Time-out ... 299
11.6.2 Operations When a Bus Error Occurs ... 299
11.6.3 Conditions Leading to Bus Errors ... 300

12. DMA Controller (DMAC) ... 301
12.1 Overview ... 301
12.2 Register Descriptions ... 303
12.2.1 DMA Mode Register (DMMOD) ... 305
12.2.2 DMA Control Register A (DMCRA) ... 307
12.2.3 DMA Control Register B (DMCRB) ... 310
12.2.4 DMA Control Register C (DMCRC) ... 311
12.2.5 DMA Control Register D (DMCRD) ... 312
12.2.6 DMA Control Register E (DMCRE) ... 313
12.2.7 DMA Current Transfer Source Address Register (DMCSA) ... 314
12.2.8 DMA Current Transfer Destination Address Register (DMCDA) ... 315
12.2.9 DMA Current Transfer Byte Count Register (DMCBC) ... 316
12.2.10 DMA Reload Transfer Source Address Register (DMRSA) ... 317
12.2.11 DMA Reload Transfer Destination Address Register (DMRDA) ... 317
12.2.12 DMA Reload Transfer Byte Count Register (DMRBC) ... 318
12.2.13 DMA Interrupt Control Register (DMICNT) ... 319
12.2.14 DMA Start Register (DMSCNT) ... 320
12.2.15 DMA Arbitration Status Register (DMASTS) ... 321
12.2.16 DMA Transfer End Detect Register (DMEDET) ... 322
12.3 Operation ... 323
12.3.1 Bus Mastership Release Timing ... 323
12.3.2 Transfer System ... 324
12.3.3 Activating the DMAC ... 326
12.3.4 Starting DMA Transfer ... 327
12.3.5 Ending DMA Transfer ... 327

Downloaded from Arrow.com.

12.3.6 Suspending, Restarting, and Canceling DMA Transfer ..... 328
12.3.7 DMA Activation Source ..... 329
12.3.7.1 Software Trigger ..... 329
12.3.7.2 Interrupt Signals on External Pins and Peripheral Function Interrupts ..... 329
12.3.8 Channel Arbitration ..... 330
12.3.9 Reload Function ..... 330
12.3.10 Rotate ..... 331
12.4 Interrupts ..... 331
12.5 Low-Power Consumption ..... 332
12.6 Usage Notes ..... 333
12.6.1 Register Settings ..... 333
12.6.2 DMA Transfer to External Devices ..... 333

13. Data Transfer Controller (DTC) ..... 334
13.1 Overview ..... 334
13.2 Register Descriptions ..... 336
13.2.1 DTC Mode Register A (MRA) ..... 337
13.2.2 DTC Mode Register B (MRB) ..... 338
13.2.3 DTC Source Address register (SAR) ..... 339
13.2.4 DTC Destination Address Register (DAR) ..... 339
13.2.5 DTC Transfer Count Register A (CRA) ..... 340
13.2.6 DTC Transfer Count Register B (CRB) ..... 341
13.2.7 DTC Control Register (DTCCR) ..... 341
13.2.8 DTC Vector Base Register (DTCVBR) ..... 342
13.2.9 DTC Address Mode Register (DTCADMOD) ..... 343
13.2.10 DTC Module Start Register (DTCST) ..... 343
13.3 Sources of Activation ..... 344
13.3.1 Allocating Transfer Data and DTC Vector Table ..... 344
13.3.2 Startup source and Vector Address ..... 346
13.4 Operation ..... 349
13.4.1 Transfer Data Read Skip Function ..... 352
13.4.2 Transfer Data Write-Back Skip Function ..... 353
13.4.3 Normal Transfer Mode ..... 354
13.4.4 Repeat Transfer Mode ..... 355
13.4.5 Block Transfer Mode ..... 357
13.4.6 Chain Transfer ..... 358
13.4.7 Operation Timing ..... 359
13.4.8 Execution Cycle of the DTC ..... 361
13.4.9 DTC Bus Mastership Release Timing ..... 361
13.5 DTC Setting Procedure ..... 362
13.6 Examples of DTC Usage ..... 363

Downloaded from Arrow.com.

13.6.1 Normal Transfer ..... 363
13.6.2 Chain Transfer ..... 364
13.6.3 Chain Transfer when Counter = 0 ..... 365
13.7 Interrupt Source ..... 367
13.8 Low-Power Consumption ..... 367
13.8.1 Setting the DTC Module Start Register ..... 367
13.9 Usage Notes ..... 367
13.9.1 Transfer Data Start Address/Transfer Source Address/Transfer Destination Address ..... 367
13.9.2 Allocating Transfer Data ..... 368

14. I/O Ports ..... 369
14.1 Overview ..... 369
14.2 Register Descriptions ..... 375
14.2.1 Data Direction Register (DDR) ..... 378
14.2.2 Data Register (DR) ..... 379
14.2.3 Port Register (PORT) ..... 380
14.2.4 Input Buffer Control Register (ICR) ..... 381
14.2.5 Pull-Up Resistor Control Register (PCR) ..... 382
14.2.6 Open Drain Control Register (ODR) ..... 383
14.2.7 Port Function Control Register0 (PFCR0) ..... 383
14.2.8 Port Function Control Register 1 (PFCR1) ..... 384
14.2.9 Port Function Control Register 2 (PFCR2) ..... 386
14.2.10 Port Function Control Register 3 (PFCR3) ..... 387
14.2.11 Port Function Control Register 4 (PFCR4) ..... 388
14.2.12 Port Function Control Register 5 (PFCR5) ..... 389
14.2.13 Port Function Control Register 6 (PFCR6) ..... 390
14.2.14 Port Function Control Register 7 (PFCR7) ..... 393
14.2.15 Port Function Control Register 8 (PFCR8) ..... 396
14.2.16 Port Function Control Register 9 (PFCR9) ..... 397
14.3 Settings of Ports ..... 398
14.3.1 Port 0 (P0) ..... 398
14.3.2 Port 1 (P1) ..... 400
14.3.3 Port 2 (P2) ..... 402
14.3.4 Port 3 (P3) ..... 405
14.3.5 Port 4 (P4) ..... 408
14.3.6 Port 5 (P5) ..... 410
14.3.7 Port 6 (P6) ..... 412
14.3.8 Port 7 (P7) ..... 415
14.3.9 Port 8 (P8) ..... 417
14.3.10 Port 9 (P9) ..... 419

Downloaded from Arrow.com.

14.3.11 Port A (PA) ..... 421
14.3.12 Port B (PB) ..... 425
14.3.13 Port C (PC) ..... 428
14.3.14 Port D (PD) ..... 431
14.3.15 Port E (PE) ..... 431
14.3.16 Port F (PF) ..... 432
14.3.17 Port G (PG) ..... 434
14.3.18 Port H (PH) ..... 436
14.4 Settings to Enable Output of the Signals ..... 438
14.5 Treatment of Unused Pins ..... 446
14.6 I/O Port Configuration ..... 447
14.7 Usage Notes ..... 451
14.7.1 Setting the Input Buffer Control Register (Pm.ICR) ..... 451
14.7.2 Setting the Port Function Control Register (PFCRn) ..... 451
14.7.3 Port Setting when A/D Converter Input is Used ..... 451

15. 16-Bit Timer Pulse Unit (TPU) ..... 452
15.1 Overview ..... 452
15.2 Register Descriptions ..... 460
15.2.1 Timer Control Register (TCR) ..... 463
15.2.2 Timer Mode Register (TMDR) ..... 467
15.2.3 Timer I/O Control Register (TIORH, TIORL, TIOR) ..... 469
15.2.4 Timer Interrupt Enable Register (TIER) ..... 479
15.2.5 Timer Status Register (TSR) ..... 481
15.2.6 Timer Counter (TCNT) ..... 482
15.2.7 Timer General Register A (TGRA) Timer General Register B (TGRB) Timer General Register C (TGRC) Timer General Register D (TGRD) ..... 482
15.2.8 Timer Start Register (TSTRA, TSTRB) ..... 483
15.2.9 Timer Synchronous Register (TSYRA, TSYRB) ..... 484
15.3 Operation ..... 485
15.3.1 Basic Functions ..... 485
15.3.2 Synchronous Operation ..... 490
15.3.3 Buffer Operation ..... 492
15.3.4 Cascaded Operation ..... 496
15.3.5 PWM Modes ..... 498
15.3.6 Phase Counting Mode ..... 503
15.3.6.1 Phase Counting Mode Application Example ..... 508
15.4 Interrupt Sources ..... 509
15.5 DTC Activation ..... 512
15.6 DMAC Activation ..... 512
15.7 A/D Converter Activation ..... 512

Downloaded from Arrow.com.

15.8 Operation Timing ..... 513
15.8.1 Input/Output Timing ..... 513
15.8.2 Interrupt Signal Timing ..... 517
15.9 Usage Notes ..... 519
15.9.1 Module Stop Function Setting ..... 519
15.9.2 Input Clock Restrictions ..... 519
15.9.3 Caution on Cycle Setting ..... 520
15.9.4 Conflict between TPUm.TCNT Write and Clear Operations ..... 520
15.9.5 Conflict between TPUm.TCNT Write and Increment Operations ..... 520
15.9.6 Conflict between TPUm.TGRy Write and Compare Match ..... 521
15.9.7 Conflict between Buffer Register Write and Compare Match ..... 521
15.9.8 Conflict between TPUm.TGRy Read and Input Capture ..... 522
15.9.9 Conflict between TPUm.TGRy Write and Input Capture ..... 522
15.9.10 Conflict between Buffer Register Write and Input Capture ..... 523
15.9.11 Conflict between Overflow/Underflow and Counter Clearing ..... 523
15.9.12 Conflict between TPUm.TCNT Write and Overflow/Underflow ..... 524
15.9.13 Multiplexing of I/O Pins ..... 524

16. Programmable Pulse Generator (PPG) ..... 525
16.1 Overview ..... 525
16.2 Register Descriptions ..... 529
16.2.1 PPG Trigger Select Register (PTRSLR) ..... 530
16.2.2 Next Data Enable Registers H and L (NDERH, NDERL) ..... 531
16.2.3 Output Data Registers H and L (PODRH, PODRL) ..... 535
16.2.4 Next Data Registers H and L (NDRH, NDRL) ..... 537
16.2.5 PPG Output Control Register (PCR) ..... 542
16.2.6 PPG Output Mode Register (PMR) ..... 544
16.3 Operation ..... 547
16.3.1 Output Timing ..... 548
16.3.2 Sample Setup Procedure for Normal Pulse Output ..... 549
16.3.3 Example of Normal Pulse Output (Example of Five-Phase Pulse Output) ..... 551
16.3.4 Non-Overlapping Pulse Output ..... 552
16.3.5 Sample Setup Procedure for Non-Overlapping Pulse Output ..... 554
16.3.6 Example of Non-Overlapping Pulse Output (Example of Four-Phase Complementary Non-Overlapping Output) ..... 556
16.3.7 Inverted Pulse Output ..... 558
16.3.8 Pulse Output Triggered by Input Capture ..... 559
16.4 Usage Note ..... 560
16.4.1 Module Stop Function Setting ..... 560

Downloaded from Arrow.com.

17. 8-Bit Timer (TMR) ..... 561
17.1 Overview ..... 561
17.2 Register Descriptions ..... 565
17.2.1 Timer Counter (TCNT) ..... 566
17.2.2 Time Constant Register A (TCORA) ..... 566
17.2.3 Time Constant Register B (TCORB) ..... 567
17.2.4 Timer Control Register (TCR) ..... 568
17.2.5 Timer Counter Control Register (TCCR) ..... 569
17.2.6 Timer Control/Status Register (TCSR) ..... 571
17.3 Operation ..... 573
17.3.1 Pulse Output ..... 573
17.3.2 Reset Input ..... 574
17.4 Operation Timing ..... 575
17.4.1 TCNT Count Timing ..... 575
17.4.2 Timing of Interrupt Flag Setting to 1 at Compare Match ..... 576
17.4.3 Timing of Timer Output at Compare Match ..... 577
17.4.4 Timing of Counter Clear by Compare Match ..... 577
17.4.5 Timing of the External Reset for TCNT ..... 578
17.4.6 Timing of Overflow Interrupt Flag Setting to 1 ..... 579
17.5 Operation with Cascaded Connection ..... 580
17.5.1 16-Bit Count Mode ..... 580
17.5.2 Compare Match Count Mode ..... 580
17.6 Interrupt Sources ..... 581
17.6.1 Interrupt Sources and DTC Activation ..... 581
17.6.2 A/D Converter Activation ..... 581
17.7 Usage Notes ..... 582
17.7.1 Module Stop State Setting ..... 582
17.7.2 Notes on Setting Cycle ..... 582
17.7.3 Conflict between TCNT Write and Counter Clear ..... 582
17.7.4 Conflict between TCNT Write and Increment ..... 583
17.7.5 Conflict between TCORA or TCORB Write and Compare Match ..... 584
17.7.6 Conflict between Compare Matches A and B ..... 584
17.7.7 Switching of Internal Clocks and TCNT Operation ..... 585
17.7.8 Clock Source Setting with Cascaded Connection ..... 586

18. Compare Match Timer (CMT) ..... 587
18.1 Overview ..... 587
18.2 Register Descriptions ..... 588
18.2.1 Compare Match Timer Start Register 0 (CMSTR0) ..... 589
18.2.2 Compare Match Timer Start Register 1 (CMSTR1) ..... 590

Downloaded from Arrow.com.

18.2.3 Compare Match Timer Control Register (CMCR) ... 591
18.2.4 Compare Match Counter (CMCNT) ... 592
18.2.5 Compare Match Constant Register (CMCOR) ... 592
18.3 Operation ... 593
18.3.1 Periodic Count Operation ... 593
18.3.2 CMCNT Count Timing ... 593
18.4 Interrupts ... 594
18.4.1 Interrupt Sources ... 594
18.4.2 Timing of Compare Match Interrupt Generation ... 594
18.5 Usage Notes ... 595
18.5.1 Setting the Module Stop Function ... 595
18.5.2 Conflict between Write and Compare-Match Processes of CMCNT ... 595
18.5.3 Conflict between Write and Count-Up Processes of CMCNT ... 595
18.5.4 Notes on Rewriting Compare Match Timer Control Register (CMCR) ... 596
18.5.5 Notes on Compare Match Timer Counter (CMCNT) and Compare Match Constant Register (CMCOR) ... 596

19. Watchdog Timer (WDT) ... 597
19.1 Overview ... 597
19.2 Register Descriptions ... 599
19.2.1 Timer Counter (TCNT) ... 599
19.2.2 Timer Control/Status Register (TCSR) ... 600
19.2.3 Reset Control/Status Register (RSTCSR) ... 601
19.2.4 Write Window A Register (WINA) ... 602
19.2.5 Write Window B Register (WINB) ... 602
19.3 Operation ... 603
19.3.1 Watchdog Timer Mode ... 603
19.3.2 Interval Timer Mode ... 604
19.4 Interrupt Source ... 604
19.5 Usage Notes ... 605
19.5.1 Notes on Register Access ... 605
19.5.2 Conflict between Timer Counter (TCNT) Write and Increment ... 606
19.5.3 Changing Values of Bits CKS[2:0] ... 606
19.5.4 Switching between Watchdog Timer Mode and Interval Timer Mode ... 607
19.5.5 Internal Reset in Watchdog Timer Mode ... 607
19.5.6 System Reset by WDTOVF# Signal ... 607
19.5.7 Transition to Watchdog Timer Mode or Software Standby Mode ... 607

20. Serial Communications Interface (SCI) ... 608
20.1 Overview ... 608
20.2 Register Descriptions ... 612

Downloaded from Arrow.com.

20.2.1 Receive Shift Register (RSR) ..... 614
20.2.2 Receive Data Register (RDR) ..... 614
20.2.3 Transmit Data Register (TDR) ..... 614
20.2.4 Transmit Shift Register (TSR) ..... 615
20.2.5 Serial Mode Register (SMR) ..... 615
20.2.6 Serial Control Register (SCR) ..... 619
20.2.7 Serial Status Register (SSR) ..... 624
20.2.8 Smart Card Mode Register (SCMR) ..... 630
20.2.9 Bit Rate Register (BRR) ..... 632
20.2.10 Serial Extended Mode Register (SEMR) ..... 639
20.3 Operation in Asynchronous Mode ..... 641
20.3.1 Serial Data Transfer Format ..... 642
20.3.2 Receive Data Sampling Timing and Reception Margin in Asynchronous Mode ..... 643
20.3.3 Clock ..... 644
20.3.4 SCI Initialization (Asynchronous Mode) ..... 645
20.3.5 Serial Data Transmission (Asynchronous Mode) ..... 646
20.3.6 Serial Data Reception (Asynchronous Mode) ..... 648
20.4 Operation in Clock Synchronous Mode ..... 652
20.4.1 Clock ..... 652
20.4.2 SCI Initialization (Clock Synchronous Mode) ..... 653
20.4.3 Serial Data Transmission (Clock Synchronous Mode) ..... 654
20.4.4 Serial Data Reception (Clock Synchronous Mode) ..... 656
20.4.5 Simultaneous Serial Data (Clock Synchronous Mode) ..... 658
20.5 Operation in Smart Card Interface Mode ..... 660
20.5.1 Sample Connection ..... 660
20.5.2 Data Format (Except in Block Transfer Mode) ..... 661
20.5.3 Block Transfer Mode ..... 662
20.5.4 Receive Data Sampling Timing and Reception Margin ..... 663
20.5.5 Initialization of the SCI ..... 665
20.5.6 Serial Data Transmission (Except in Block Transfer Mode) ..... 666
20.5.7 Serial Data Reception (Except in Block Transfer Mode) ..... 669
20.5.8 Clock Output Control ..... 670
20.6 Interrupt Sources ..... 672
20.6.1 Interrupts in Serial Communications Interface Mode ..... 672
20.6.2 Interrupts in Smart Card Interface Mode ..... 673
20.7 Usage Notes ..... 674
20.7.1 Setting the Module Stop Function ..... 674
20.7.2 Break Detection and Processing ..... 674
20.7.3 Mark State and Break Detection ..... 674
20.7.4 Receive Error Flags and Transmit Operations (Clock Synchronous Mode Only) ..... 674

Downloaded from Arrow.com.

20.7.5 Writing Data to TDR ... 674
20.7.6 Restrictions on Clock Synchronous Transmission ... 675
20.7.7 Restrictions on Using DTC or DMAC ... 675
20.7.8 SCI Operations during Power-Down State ... 675
20.7.9 External Clock Input in Clock Synchronous Mode ... 678

21. CRC Calculator (CRC) ... 679
21.1 Overview ... 679
21.2 Register Descriptions ... 680
21.2.1 CRC Control Register (CRCCR) ... 680
21.2.2 CRC Data Input Register (CRCDIR) ... 681
21.2.3 CRC Data Output Register (CRCDOR) ... 681
21.3 Operation ... 682
21.4 Usage Notes ... 685
21.4.1 Module Stop Function Setting ... 685
21.5 Note on Transmission ... 685

22. I²C Bus Interface (RIIC) ... 686
22.1 Overview ... 686
22.2 Register Descriptions ... 690
22.2.1 I²C Bus Control Register 1 (ICCR1) ... 692
22.2.2 I²C Bus Control Register 2 (ICCR2) ... 695
22.2.3 I²C Bus Mode Register 1 (ICMR1) ... 699
22.2.4 I²C Bus Mode Register 2 (ICMR2) ... 701
22.2.5 I²C Bus Mode Register 3 (ICMR3) ... 703
22.2.6 I²C Bus Function Enable Register (ICFER) ... 706
22.2.7 I²C Bus Status Enable Register (ICSER) ... 708
22.2.8 I²C Bus Interrupt Enable Register (ICIER) ... 710
22.2.9 I²C Bus Status Register 1 (ICSR1) ... 712
22.2.10 I²C Bus Status Register 2 (ICSR2) ... 716
22.2.11 Slave Address Register Ly (SARLy) (y = 0 to 2) ... 720
22.2.12 Slave Address Register Uy (SARUy) (y = 0 to 2) ... 721
22.2.13 I²C Bus Bit Rate Low-Level Register (ICBRL) ... 722
22.2.14 I²C Bus Bit Rate High-Level Register (ICBRH) ... 723
22.2.15 I²C Bus Transmit Data Register (ICDRT) ... 725
22.2.16 I²C Bus Receive Data Register (ICDRR) ... 725
22.2.17 I²C Bus Shift Register (ICDRS) ... 725
22.2.18 Internal Counter for Timeout (TMOCNT) ... 726
22.3 Operation ... 727
22.3.1 Communication Data Format ... 727
22.3.2 Initial Settings ... 728

Downloaded from Arrow.com.

22.3.3 Master Transmitter Operation ..... 729
22.3.4 Master Receiver Operation ..... 733
22.3.5 Slave Transmitter Operation ..... 738
22.3.6 Slave Receiver Operation ..... 741
22.4 SCL Synchronization Circuit ..... 744
22.5 Facility for Delaying SDA Output ..... 745
22.6 Digital Noise-Filter Circuits ..... 746
22.7 Address Match Detection ..... 747
22.7.1 Slave-Address Match Detection ..... 747
22.7.2 Detection of the General Call Address ..... 749
22.7.3 Device-ID Address Detection ..... 750
22.7.4 Host Address Detection ..... 752
22.8 Function to Automatically Hold SCLn Clock Low ..... 753
22.8.1 Function to Prevent Wrong Transmission of Transmit Data ..... 753
22.8.2 NACK Reception Transfer Suspension Function ..... 754
22.8.3 Function to Prevent Failure to Receive Data ..... 755
22.9 Arbitration-Lost Detection Functions ..... 757
22.9.1 Master Arbitration Lost Detection (MALE Bit) ..... 757
22.9.2 Function to Detect Loss of Arbitration during NACK Transmission (NALE Bit) ..... 759
22.9.3 Slave Arbitration Lost Detection (SALE Bit) ..... 760
22.10 Start Condition/Restart Condition/Stop Condition Issuing Function ..... 761
22.10.1 Issuing a Start Condition ..... 761
22.10.2 Issuing a Restart Condition ..... 761
22.10.3 Issuing a Stop Condition ..... 762
22.11 Bus Hanging ..... 763
22.11.1 Timeout Detection Function ..... 763
22.11.2 Extra SCL Clock Cycle Output Function ..... 765
22.11.3 RIIC Reset and Internal Reset ..... 766
22.12 SMBus Operation ..... 767
22.12.1 SMBus Timeout Measurement ..... 767
22.12.2 Packet Error Code (PEC) ..... 768
22.12.3 SMBus Host Notification Protocol/Notify ARP Master ..... 769
22.13 Interrupt Request ..... 770
22.14 Reset States ..... 771
22.15 Usage Notes ..... 772
22.15.1 Setting Module Stop Function ..... 772
22.15.2 Setting Input Buffer Control Register ..... 772
22.15.3 Timings for Writing and Outputting of Transmit Acknowledge Bit ..... 772
22.15.4 Restrictions on Timings for Stop Condition Issuance Request and Transmit Data Writing in Master Transmitter Mode ..... 772

Downloaded from Arrow.com.

22.15.5 Notes when Communication is Restarted with the NACK Reception in Master Mode ..... 773
22.15.6 Notes on the RDRF Flag Set Timing Selection Bit (RDRFS) ..... 773

23. A/D Converter ..... 774
23.1 Overview ..... 774
23.2 Register Descriptions ..... 781
23.2.1 A/D Data Register y (ADDRy) (y = A to D) ..... 782
23.2.2 A/D Control/Status Register (ADCSR) ..... 783
23.2.3 A/D Control Register (ADCR) ..... 785
23.2.4 ADDRy Format Select Register (ADDPR) ..... 787
23.2.5 A/D Sampling State Register (ADSSTR) ..... 787
23.3 Operation ..... 788
23.3.1 Single Mode ..... 788
23.3.2 Scan Mode ..... 790
23.3.2.1 Continuous Scan Mode ..... 790
23.3.2.2 Single Scan Mode ..... 792
23.3.3 Input Sampling and A/D Conversion Time ..... 793
23.3.4 Activation by External Triggers ..... 795
23.3.5 Activation by the Compare-Match/Input-Capture A to D Signals from TPU0 ..... 796
23.3.6 Activation by the Compare-Match/Input-Capture A Signals from TPU0 to TPU5 ..... 797
23.3.7 Activation on Compare-Match of TMR Units ..... 798
23.4 Interrupt Source ..... 799
23.5 A/D Conversion Accuracy Definitions ..... 799
23.6 Usage Notes ..... 801
23.6.1 Module Stop Function Setting ..... 801
23.6.2 Notes on Disabling A/D Conversion ..... 801
23.6.3 Notes on Restarting A/D Conversion ..... 801
23.6.4 Notes on Entering Power-Down States ..... 801
23.6.5 Permissible Impedance of Signal Sources ..... 802
23.6.6 Factors Affecting Absolute Accuracy ..... 802
23.6.7 Ranges of Settings for Analog Power Supply and Other Pins ..... 803
23.6.8 Point for Caution Regarding Board Design ..... 803
23.6.9 Point for Caution Regarding Countermeasures for Noise ..... 804
23.6.10 Realizing High-Speed Conversion ..... 805
23.6.11 Notes when Using Multiple Units of A/D Converter ..... 806

24. D/A Converter ..... 808
24.1 Overview ..... 808
24.2 Register Descriptions ..... 810
24.2.1 D/A Data Register y (DADRy) (y = 0, 1) ..... 810
24.2.2 D/A Control Register (DACR) ..... 811

Downloaded from Arrow.com.

24.2.3 DADRy Format Select Register (DADPR) ... 813
24.3 Operation ... 814
24.4 Usage Notes ... 815
24.4.1 Module Stop Function Setting ... 815
24.4.2 Operation of the D/A Converter in Module Stop State ... 815
24.4.3 Operation of the D/A Converter in Software Standby Mode ... 815
24.4.4 Note on Entering Deep Software Standby Mode ... 815
24.4.5 Note when Using the A/D Converter and D/A Converter Simultaneously ... 815

25. RAM ... 817
25.1 Overview ... 817
25.2 Operation ... 817
25.2.1 Data Retention ... 817
25.2.2 Power-Down Function ... 817

26. ROM (Flash Memory for Code Storage) ... 818
26.1 Overview ... 818
26.2 Register Descriptions ... 821
26.2.1 Flash Mode Register (FMODR) ... 822
26.2.2 Flash Access Status Register (FASTAT) ... 823
26.2.3 Flash Access Error Interrupt Enable Register (FAEINT) ... 825
26.2.4 FCU RAM Enable Register (FCURAME) ... 826
26.2.5 Flash Status Register 0 (FSTATR0) ... 827
26.2.6 Flash Status Register 1 (FSTATR1) ... 830
26.2.7 Flash Ready Interrupt Enable Register (FRDYIE) ... 831
26.2.8 Flash P/E Mode Entry Register (FENTRYR) ... 832
26.2.9 Flash Protection Register (FPROTR) ... 835
26.2.10 Flash Reset Register (FRESETR) ... 836
26.2.11 FCU Command Register (FCMDR) ... 837
26.2.12 FCU Processing Switching Register (FCPSR) ... 838
26.2.13 Flash P/E Status Register (FPESTAT) ... 839
26.2.14 Peripheral Clock Notification Register (PCKAR) ... 840
26.2.15 Flash Write Erase Protection Register (FWEPROR) ... 841
26.3 Configuration of Memory Mats for the ROM ... 842
26.4 Block Configuration ... 842
26.5 Operating Modes Associated with the ROM ... 843
26.6 Programming and Erasing the ROM ... 846
26.6.1 FCU Modes ... 846
26.6.1.1 ROM Read Modes ... 847
26.6.1.2 ROM P/E Modes ... 847
26.6.2 FCU Commands ... 848

Downloaded from Arrow.com.

26.6.3 Connections between FCU Modes and Commands ..... 850
26.6.4 FCU Command Usage ..... 851
26.6.4.1 Mode Transitions ..... 851
26.6.4.2 Programming and Erasure Procedures ..... 855
26.6.4.3 Error Processing ..... 864
26.6.4.4 Suspension and Resumption ..... 865
26.7 Suspending Operation ..... 868
26.7.1 Suspension during Programming ..... 868
26.7.2 Suspension during Erasure (Suspension Priority Mode) ..... 869
26.7.3 Suspension during Erasure (Erasure Priority Mode) ..... 870
26.8 Protection ..... 871
26.8.1 Software Protection ..... 871
26.8.2 Error Protection ..... 871
26.9 User Boot Mode ..... 873
26.10 Boot Mode ..... 873
26.10.1 System Configuration ..... 873
26.10.2 ID Code Protection ..... 874
26.10.3 State Transitions in Boot Mode ..... 876
26.10.4 Automatic Adjustment of the Bit Rate ..... 878
26.10.5 Inquiry/Selection Host Command Wait State ..... 879
26.10.6 ID Code Wait State ..... 893
26.10.7 Programming/Erasure Host Command Wait State ..... 894
26.11 ID Code Protection on Connection of the On-Chip Debugger ..... 903
26.12 ROM Code Protection ..... 904
26.13 Usage Notes ..... 905

27. Data Flash (Flash Memory for Data Storage) ..... 907
27.1 Overview ..... 907
27.2 Register Descriptions ..... 909
27.2.1 Flash Mode Register (FMODR) ..... 910
27.2.2 Flash Access Status Register (FASTAT) ..... 911
27.2.3 Flash Access Error Interrupt Enable Register (FAEINT) ..... 913
27.2.4 Data Flash Read Enable Register (DFLRE) ..... 915
27.2.5 Data Flash Programming/Erasure Enable Register (DFLWE) ..... 916
27.2.6 Flash P/E Mode Entry Register (FENTRYR) ..... 917
27.2.7 Data Flash Blank Check Control Register (DFLBCCNT) ..... 919
27.2.8 Data Flash Blank Check Status Register (DFLBCSTAT) ..... 920
27.3 Configuration of Memory Mat for the Data Flash Memory ..... 921
27.4 Block Configuration ..... 921
27.5 Operating Modes Associated with the Data Flash ..... 922
27.6 Programming and Erasing the Data Flash Memory ..... 923

Downloaded from Arrow.com.

27.6.1 FCU Modes ..... 923
27.6.1.1 ROM P/E Modes ..... 924
27.6.1.2 ROM/Data Flash Read Mode ..... 924
27.6.1.3 Data Flash P/E Modes ..... 924
27.6.2 FCU Commands ..... 925
27.6.3 Connections between FCU Modes and Commands ..... 926
27.6.4 FCU Command Usage ..... 927
27.7 Protection ..... 931
27.7.1 Software Protection ..... 931
27.7.2 Error Protection ..... 932
27.8 Boot Mode ..... 933
27.8.1 Inquiry/Selection Host Commands ..... 933
27.8.2 Programming/Erasing Host Commands ..... 935
27.9 Usage Notes ..... 936

28. Boundary Scan ..... 937
28.1 Features ..... 937
28.2 Register Descriptions ..... 938
28.2.1 Instruction Register (JTIR) ..... 939
28.2.2 Bypass Register (JTBPR) ..... 939
28.2.3 Boundary Scan Register (JTBSR) ..... 939
28.2.4 IDCODE Register (JTID) ..... 948
28.3 Operations ..... 949
28.3.1 TAP Controller ..... 949
28.3.2 List of Commands ..... 950
28.4 Usage Notes ..... 952

29. Electrical Characteristics ..... 954
29.1 Absolute Maximum Ratings ..... 954
29.2 DC Characteristics ..... 955
29.3 AC Characteristics ..... 958
29.3.1 Clock Timing ..... 958
29.3.2 Control Signal Timing ..... 962
29.3.3 Bus Timing ..... 963
29.3.4 Timing of On-Chip Peripheral Modules ..... 968
29.4 A/D Conversion Characteristics ..... 976
29.5 D/A Conversion Characteristics ..... 976
29.6 ROM (Flash Memory for Code Storage) Characteristics ..... 977
29.7 Data Flash (Flash Memory for Data Storage) Characteristics ..... 978

Downloaded from Arrow.com.

Appendix 1. Port States in Each Processing Mode ..... 980

Appendix 2. Package Dimensions ..... 985

REVISION HISTORY ..... 987

Downloaded from Arrow.com.

# 1. Overview

## 1.1 Features

The RX610 Group is an MCU with the high-speed, high-performance RX CPU as its core.

One basic instruction is executable in one cycle of the system clock. Calculation functionality is further enhanced, with the inclusion of a single-precision floating-point calculation unit as well as a 32-bit multiplier and divider. Additionally, code efficiency is improved by instructions with lengths that are variable in byte units and by an enhanced range of addressing modes.

Timers, serial communication interfaces, $I^2C$ bus interfaces, an A/D converter, and a D/A converter are incorporated as peripheral functions which are essential to embedded devices.

Facilities for connecting external memory are also included, enabling direct connection to memory and peripheral LSI circuits. The on-chip memory is flash memory capable of large-capacity, high-speed operation, and this significantly reduces the cost of configuring systems.

### 1.1.1 Applications

Office automation equipment and digital industrial equipment



Downloaded from Arrow.com.

### 1.1.2 Outline of Specifications

Table 1.1 lists the specifications of the RX610 Group in outline.

**Table 1.1 Outline of Specifications**

| Classification | Module/Function | Description |
|---|---|---|
| CPU | CPU | • Maximum operating frequency: 100 MHz<br>• 32-bit RX CPU<br>• Minimum instruction execution time: One instruction in one state (in one system clock cycle)<br>• Address space: 4-Gbyte linear address<br>• Register set of the CPU<br>General purpose: Sixteen 32-bit registers<br>Control: Nine 32-bit registers<br>Accumulator: One 64-bit register<br>• Basic instructions: 73<br>• Floating-point operation instructions: 8<br>• DSP instructions: 9<br>• Addressing modes: 10<br>• Data arrangement<br>Instructions: Little endian<br>Data: Selectable as little endian or big endian<br>• On-chip 32-bit multiplier: 32 x 32 → 64 bits<br>• On-chip divider: 32 / 32 → 32 bits<br>• Barrel shifter: 32 bits |
| | FPU | • Single precision (32-bit) floating point<br>• Data types and floating-point exceptions conforming to the IEEE754 standard |
| Memory | Flash | • Flash capacity: 2 Mbytes (max.)<br>• Three types of on-board programming modes<br>SCI boot mode, user program mode, and user boot mode |
| | RAM | RAM capacity: 128 Kbytes |
| | Data flash | Data flash capacity: 32 Kbytes |
| MCU operating modes | | Single-chip mode, on-chip ROM enabled extended mode, and on-chip ROM disabled extended mode |
| Clock | Clock generation circuit | • One main clock oscillation circuit<br>• Includes a PLL circuit and frequency divider, so the operating frequency is selectable<br>• System clock, peripheral module clock, and external bus clock are independently specifiable.<br>The CPU, DMAC, DTC, ROM, and RAM run in synchronization with the system clock (ICLK): 8 to 100 MHz<br>Peripheral modules run in synchronization with the peripheral module clock (PCLK): 8 to 50 MHz<br>Devices connected to the external bus run in synchronization with the external bus clock (BCLK): 8 to 25 MHz |
| Power down | Power-down function | • Module stop function<br>• Four power-down modes<br>Sleep mode, all-module clock stop mode, software standby mode, and deep software standby mode |



Downloaded from Arrow.com.

| Classification | Module/Function | Description |
|---|---|---|
| Interrupt | Interrupt control unit | • Peripheral function interrupts: 116<br>• External interrupts: 16 (pins IRQ15 to IRQ0)<br>• Non-maskable interrupt: 1 (the NMI pin)<br>• Eight priority orders specifiable |
| External bus extension | | • The external address space can be divided into eight areas (CS0 to CS7), each of which is independently controllable.<br>Capacity of each area: 16 Mbytes<br>Chip-select signals (CS0# to CS7#) can be output for each area.<br>8-bit or 16-bit bus space can be specified for each area.<br>The data arrangement is selectable as little endian or big endian for each area. (only for data)<br>• Separate bus system<br>• Wait control<br>• Write buffer programming |
| DMA | DMA controller | • 4-channel DMA transfer available<br>• Activation sources: Software trigger, external interrupts, and interrupt requests from peripheral functions |
| | Data transfer controller | • Three transfer modes: Normal transfer, repeat transfer, and block transfer<br>• Activated by interrupt requests (chain transfer enabled) |
| I/O ports | Programmable I/O ports | • I/O pins: 117 (144-pin LQFP), 140 (176-pin LFBGA)<br>• Pull-up resistors: 40<br>• Open-drain outputs: 16<br>• 5-V tolerance: 10 |
| Timer | 16-bit timer pulse unit | • (16 bits x 6 channels) x 2 units<br>• Up to 16 pulse inputs and outputs<br>• Select from among 7 or 8 counter-input clocks for each channel<br>• Input capture/output compare function<br>• Maximum of 15-phase PWM output poss ble in PWM mode<br>• Buffered operation, phase counting mode (two-phase encoder input), and cascaded operation (32 bits x 2 channels) settable for each channel<br>• PPG output trigger can be generated<br>• Conversion start trigger for the A/D converter can be generated |
| | Programmable pulse generator | • (4 bits x 4 groups) x 2 units<br>• Provides pulse outputs by using the TPU output as a trigger<br>• Maximum of 32-bit pulse output possible |
| | 8-bit timer | • (8 bits x 2 channels) x 2 units<br>• Select from among 8 clock sources (7 internal clocks and 1 external clock)<br>• Allows the output of pulse trains with a desired duty cycle or PWM signals<br>• Cascading of 2 channels enables it to be used as a 16-bit timer<br>• Generation of trigger to start A/D converter conversion<br>• Capable of generating baud rate clock for SCI5 and SCI6 |
| | Compare match timer | • (16 bits x 2 channels) x 2 units<br>• Select from among 4 counter-input clocks |



Downloaded from Arrow.com.

| | | |
|---|---|---|
| Watchdog timer | | • 8 bits x 1 channel<br>• Select from among 8 counter-input clocks<br>• Switchable between watchdog timer mode and interval timer mode |
| Communication function | Serial communication interface | • 7 channels<br>• Serial communication mode:<br>Asynchronous, clock synchronous, and smart card interface<br>• On-chip baud rate generator allows any bit rate to be selected<br>• Choice of LSB-first or MSB-first transfer<br>• Enables average transfer rate clock input from TMR (SCI5, SCI6) |
| | $I^2C$ bus interface | • 2 channels<br>• Communication format<br>$I^2C$ bus format/SMBus format<br>Master/slave selectable (For multi-master operation)<br>• Maximum transfer rate: 1 Mbps |
| A/D converter | | • 4 units (1 unit x 4 channels)<br>• 10-bit resolution<br>• Conversion time: 1.0 μs per channel (at 50-MHz (PCLK) operation)<br>• Two kinds of operating modes<br>Single mode and scan mode (single scan mode or continuous scan mode)<br>• Sample-and-hold function<br>• Three types of A/D conversion start<br>Conversion can be started by software, a conversion start trigger by the timer (TPU or TMR), or an external trigger signal. |
| D/A converter | | • 2 channels<br>• 10-bit resolution<br>• Output voltage: 0 V to VREFH |
| CRC calculator | | • CRC code generation for arbitrary data lengths in 8-bit units<br>• One of three generating polynomials selectable<br>$X^8 + X^2 + X + 1$, $X^{16} + X^{15} + X^2 + 1$, $X^{16} + X^{12} + X^5 + 1$<br>• CRC code generation for LSB-first or MSB-first communication selectable |
| Operating frequency | | 8 to 100 MHz |
| Power supply voltage | | $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6V, VREFH = 3.0 to $AV_{CC}$ |
| Supply current | | 50 mA (typ.) (regular specifications) |
| Operating temperature | | −20 to +85 C (regular specifications), −40 to +85 C (wide-range specifications) |
| Package | | 176-pin LFBGA (PLBG0176GA-A)<br>144-pin LQFP (PLQP0144KA-A) |



Downloaded from Arrow.com.

## 1.2 List of Products

Table 1.2 is the list of products, and figure 1.1 shows how to read the product part no.

**Table 1.2 List of Products**

| Part No. | Package | ROM Capacity | RAM Capacity | Data Flash | Operating Frequency (Max.) |
|---|---|---|---|---|---|
| R5F56108VNFP | PLQP0144KA-A | 2 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56108VDFP | PLQP0144KA-A | 2 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56108WNBG | PLBG0176GA-A | 2 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56108WDBG | PLBG0176GA-A | 2 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56107VNFP | PLQP0144KA-A | 1.5 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56107VDFP | PLQP0144KA-A | 1.5 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56107WNBG | PLBG0176GA-A | 1.5 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56107WDBG | PLBG0176GA-A | 1.5 Mbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56106VNFP | PLQP0144KA-A | 1 Mbyte | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56106VDFP | PLQP0144KA-A | 1 Mbyte | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56106WNBG | PLBG0176GA-A | 1 Mbyte | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56106WDBG | PLBG0176GA-A | 1 Mbyte | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56104VNFP | PLQP0144KA-A | 768 Kbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56104VDFP | PLQP0144KA-A | 768 Kbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56104WNBG | PLBG0176GA-A | 768 Kbytes | 128 Kbytes | 32 Kbytes | 100 MHz |
| R5F56104WDBG | PLBG0176GA-A | 768 Kbytes | 128 Kbytes | 32 Kbytes | 100 MHz |



Downloaded from Arrow.com.

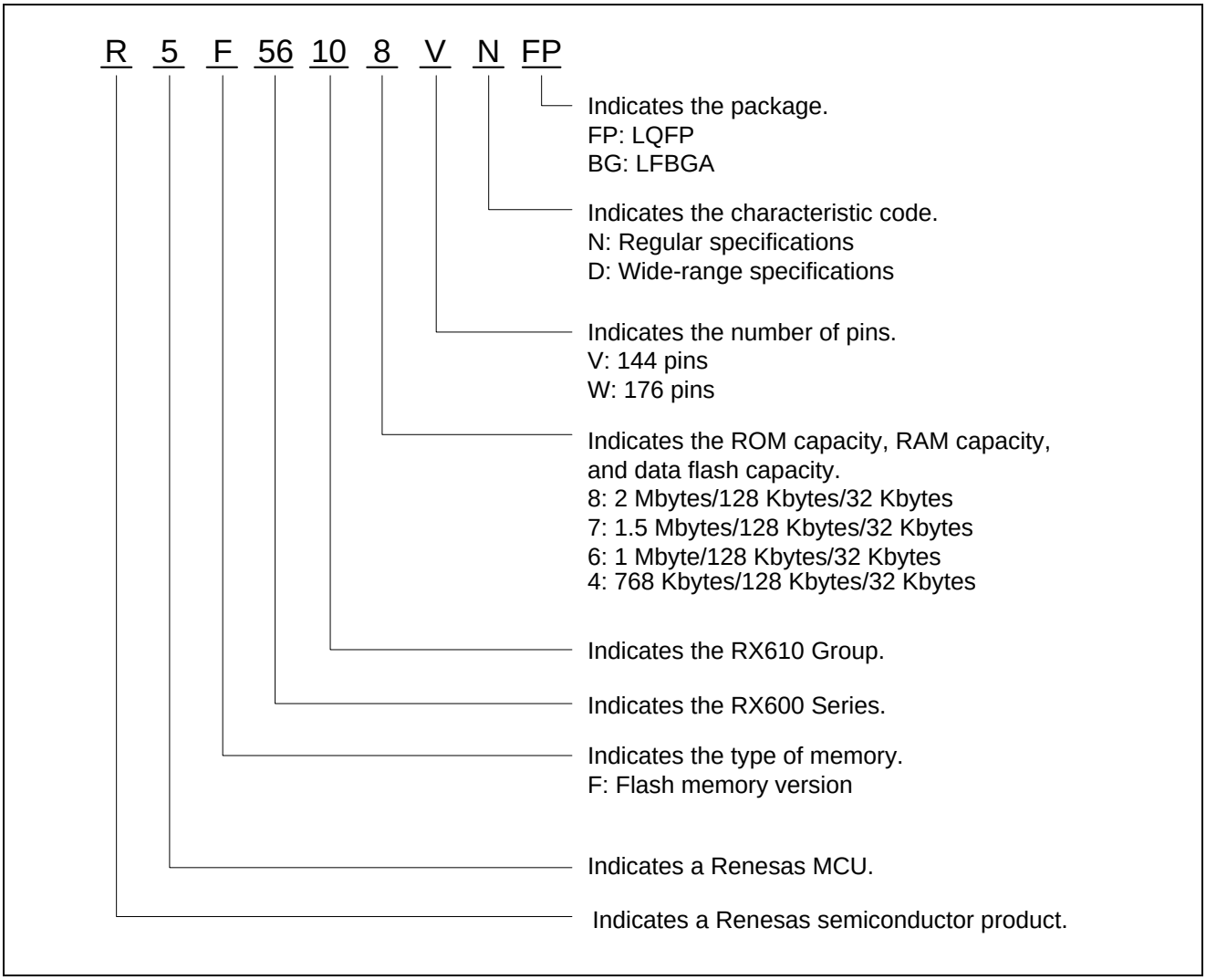


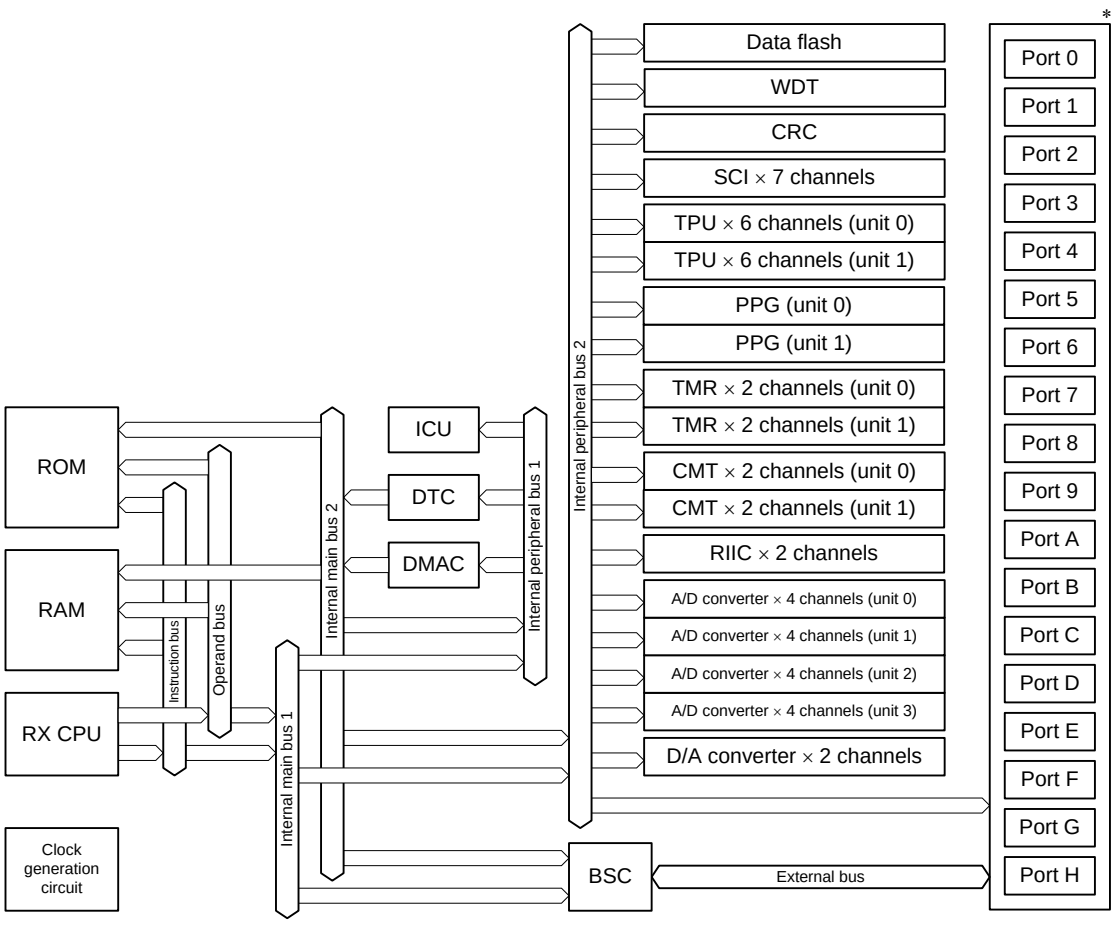

**Figure 1.1 How to Read the Product Part No.**



Downloaded from Arrow.com.

## 1.3 Block Diagram

Figure 1.2 shows a block diagram of the RX610 Group.




[Legend]

| | | | |
|---|---|---|---|
| ICU: | Interrupt control unit | SCI: | Serial communications interfaces |
| DTC: | Data transfer controller | TPU: | 16-bit timer pulse unit |
| DMAC: | DMA controller | PPG: | Programmable pulse generator |
| BSC: | Bus controller | TMR: | 8-bit timer |
| WDT: | Watchdog timer | CMT: | Compare match timer |
| CRC: | CRC (Cyclic Redundancy Check) calculator | RIIC: | $I^2C$ bus interface |

Note: * Ports F and H are not included in the 144-pin LQFP package.

**Figure 1.2 Block Diagram**



Downloaded from Arrow.com.

## 1.4 Pin Assignments

Figures 1.3 and 1.4 show the pin assignments of the 176-pin LFBGA and the 144-pin LQFP, respectively. Figure 1.5 (assistance diagram) shows the pin assignment the 144-pin LQFP. Tables 1.3 and 1.4 show the lists of pins and pin functions of the 176-pin LFBGA and the 144-pin LQFP, respectively.

| | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | PE0 | PE2 | PE5 | PG5 | VSS | PA1 | PA5 | PH1 | P70 | P74 | PB3 | PB6 | PC1 | VCC | PC3 | 15 |
| 14 | PD6 | PE1 | PE3 | PE7 | PG6 | PA0 | PA4 | PH0 | VCC | P73 | PB4 | PC0 | PC2 | PC4 | PC5 | 14 |
| 13 | PD4 | PD5 | PD7 | PE6 | PG7 | PA2 | PA6 | VSS | P71 | PB1 | PB5 | VSS | PH2 | PC6 | P75 | 13 |
| 12 | P63 | VCC | VSS | PE4 | VCC | PA3 | PA7 | PB0 | P72 | PB2 | PB7 | PC7 | P76 | P77 | PH3 | 12 |
| 11 | P60 | P61 | P62 | P64 | | | | | | | | PH4 | VSS | VCC | PH5 | 11 |
| 10 | PD1 | PD0 | PD2 | PD3 | | | | | | | | P51 | P50 | PH6 | PH7 | 10 |
| 9 | PG2 | PG1 | PG3 | PG4 | | | | | | | | P81 | P80 | P52 | P53 | 9 |
| 8 | P97 | P96 | BSCANP | PG0 | | | | | | | | P83 | VSS | VCC | P82 | 8 |
| 7 | P93 | P92 | P94 | P95 | | | | | | | | P57 | P56 | P54 | P55 | 7 |
| 6 | P90 | VCC | VSS | P91 | | | | | | | | P37 | P36 | P84 | P35 | 6 |
| 5 | P46 | P45 | P47 | P44 | | | | | | | | P14 | P12 | P11 | P10 | 5 |
| 4 | P43 | P42 | P41 | P40 | P00 | MDE | P86 | VSS | P34 | P33 | PF0 | VSS | P16 | P15 | P13 | 4 |
| 3 | VREFL | VREFH | P03 | AVSS | EMLE | VCL | P85 | EXTAL | PF6 | P32 | PF3 | VCC | P20 | PLLVCC | PLLVSS | 3 |
| 2 | AVCC | P05 | P66 | P01 | WDTOVF# | MD0 | XTAL | NMI | PF4 | P30 | PF1 | P26 | P24 | P22 | P17 | 2 |
| 1 | P04 | P67 | P02 | P65 | VSS | MD1 | RES# | VCC | PF5 | P31 | PF2 | P27 | P25 | P23 | P21 | 1 |
| | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | |

RX610Group
PLBG0176GA-A
(176-pin LFBGA)
(Upper perspective view)

**Figure 1.3 Pin Assignment of the 176-pin LFBGA**



Downloaded from Arrow.com.

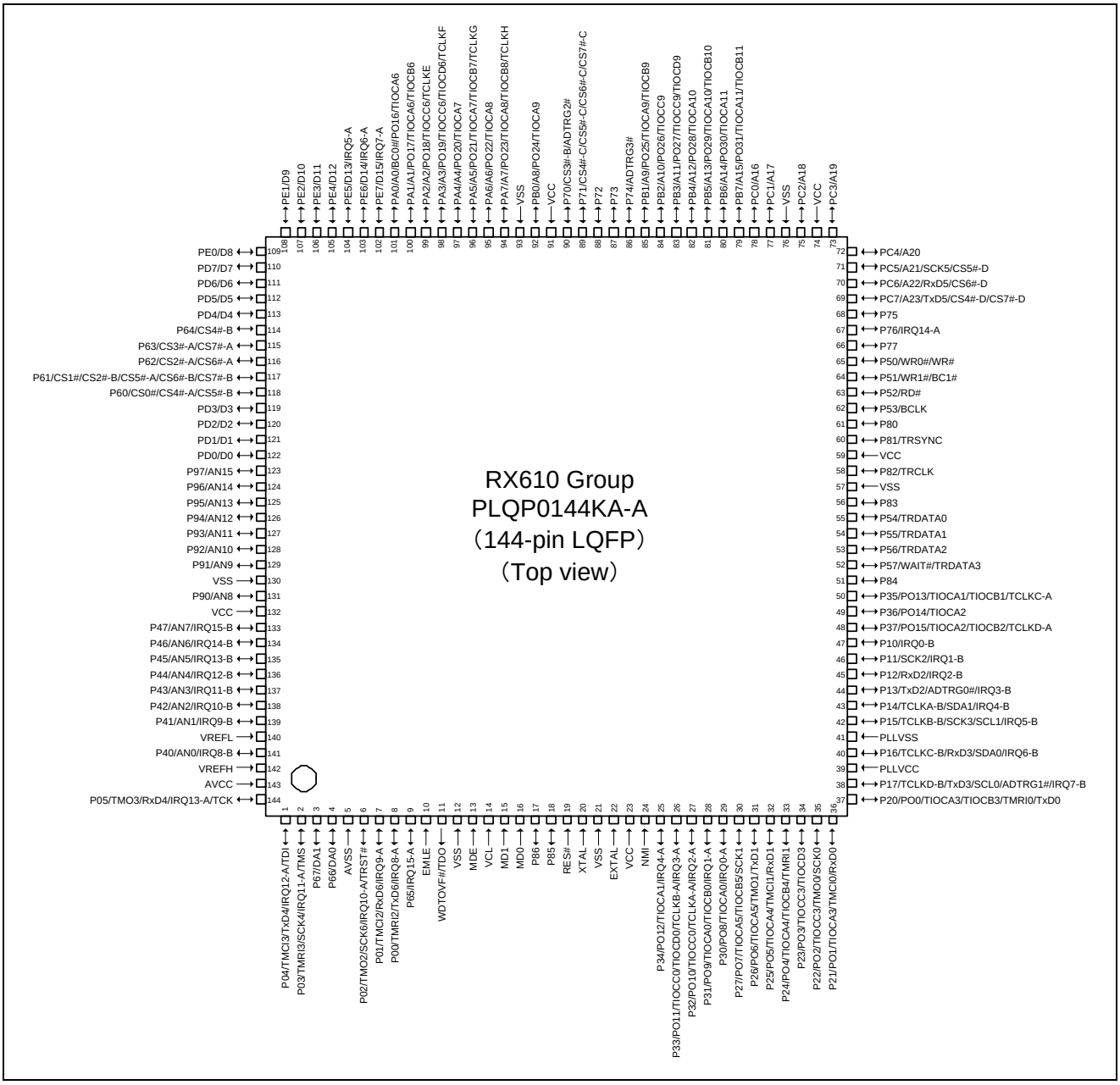



**Figure 1.4 Pin Assignment of the 144-Pin LQFP**



Downloaded from Arrow.com.

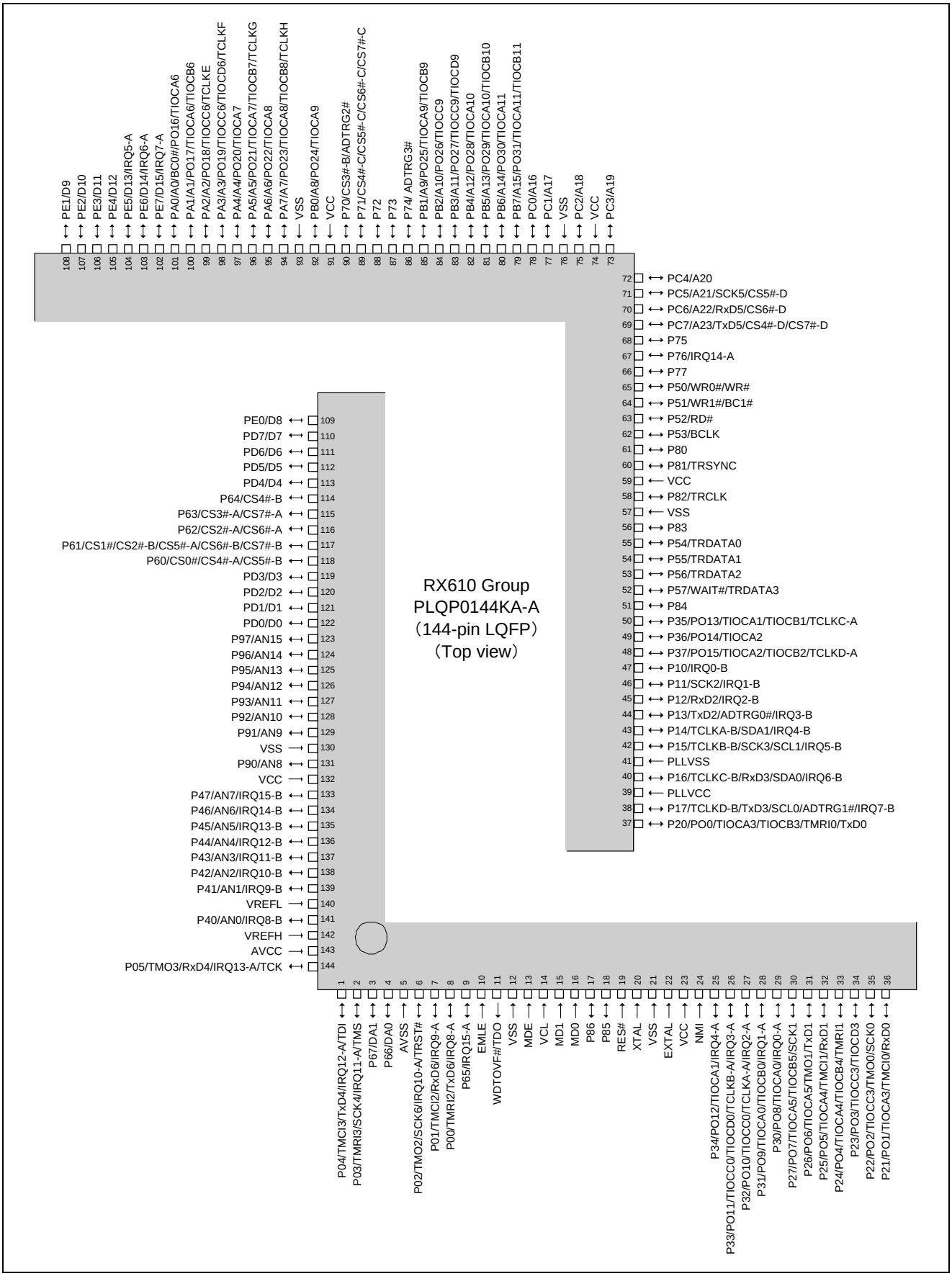



**Figure 1.5 Pin Assignment (Assistance Diagram) of the 144-Pin LQFP**



Downloaded from Arrow.com.

**Table 1.3 List of Pins and Pin Functions (176-Pin LFBGA)**

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| A1 | | P04 | IRQ12-A | | TMCI3 | TxD4 | | TDI |
| A2 | AVCC | | | | | | | |
| A3 | VREFL | | | | | | | |
| A4 | | P43 | IRQ11-B | | | | AN3 | |
| A5 | | P46 | IRQ14-B | | | | AN6 | |
| A6 | | P90 | | | | | AN8 | |
| A7 | | P93 | | | | | AN11 | |
| A8 | | P97 | | | | | AN15 | |
| A9 | | PG2 | | | | | | |
| A10 | | PD1 | | D1 | | | | |
| A11 | | P60 | | CS0#/ CS4#-A/ CS5#-B | | | | |
| A12 | | P63 | | CS3#-A/ CS7#-A | | | | |
| A13 | | PD4 | | D4 | | | | |
| A14 | | PD6 | | D6 | | | | |
| A15 | | PE0 | | D8 | | | | |
| B1 | | P67 | | | | | DA1 | |
| B2 | | P05 | IRQ13-A | | TMO3 | RxD4 | | TCK |
| B3 | VREFH | | | | | | | |
| B4 | | P42 | IRQ10-B | | | | AN2 | |
| B5 | | P45 | IRQ13-B | | | | AN5 | |
| B6 | VCC | | | | | | | |
| B7 | | P92 | | | | | AN10 | |
| B8 | | P96 | | | | | AN14 | |
| B9 | | PG1 | | | | | | |
| B10 | | PD0 | | D0 | | | | |
| B11 | | P61 | | CS1#/ CS2#-B/ CS5#-A/ CS6#-B/ CS7#-B | | | | |
| B12 | VCC | | | | | | | |
| B13 | | PD5 | | D5 | | | | |
| B14 | | PE1 | | D9 | | | | |
| B15 | | PE2 | | D10 | | | | |
| C1 | | P02 | IRQ10-A | | TMO2 | SCK6 | | TRST# |
| C2 | | P66 | | | | | DA0 | |
| C3 | | P03 | IRQ11-A | | TMRI3 | SCK4 | | TMS |
| C4 | | P41 | IRQ9-B | | | | AN1 | |
| C5 | | P47 | IRQ15-B | | | | AN7 | |



Downloaded from Arrow.com.

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| C6 | VSS | | | | | | | |
| C7 | | P94 | | | | | AN12 | |
| C8 | BSCANP | | | | | | | |
| C9 | | PG3 | | | | | | |
| C10 | | PD2 | | D2 | | | | |
| C11 | | P62 | | CS2#-A/ CS6#-A | | | | |
| C12 | VSS | | | | | | | |
| C13 | | PD7 | | D7 | | | | |
| C14 | | PE3 | | D11 | | | | |
| C15 | | PE5 | IRQ5-A | D13 | | | | |
| D1 | | P65 | IRQ15-A | | | | | |
| D2 | | P01 | IRQ9-A | | TMCI2 | RxD6 | | |
| D3 | AVSS | | | | | | | |
| D4 | | P40 | IRQ8-B | | | | AN0 | |
| D5 | | P44 | IRQ12-B | | | | AN4 | |
| D6 | | P91 | | | | | AN9 | |
| D7 | | P95 | | | | | AN13 | |
| D8 | | PG0 | | | | | | |
| D9 | | PG4 | | | | | | |
| D10 | | PD3 | | D3 | | | | |
| D11 | | P64 | | CS4#-B | | | | |
| D12 | | PE4 | | D12 | | | | |
| D13 | | PE6 | IRQ6-A | D14 | | | | |
| D14 | | PE7 | IRQ7-A | D15 | | | | |
| D15 | | PG5 | | | | | | |
| E1 | VSS | | | | | | | |
| E2 | WDTOVF# | | | | | | | TDO |
| E3 | EMLE | | | | | | | |
| E4 | | P00 | IRQ8-A | | TMRI2 | TxD6 | | |
| E12 | VCC | | | | | | | |
| E13 | | PG7 | | | | | | |
| E14 | | PG6 | | | | | | |
| E15 | VSS | | | | | | | |
| F1 | MD1 | | | | | | | |
| F2 | MD0 | | | | | | | |
| F3 | VCL | | | | | | | |
| F4 | MDE | | | | | | | |
| F12 | | PA3 | | A3 | PO19/ TIOCC6/ TIOCD6/ TCLKF | | | |



Downloaded from Arrow.com.

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| F13 | | PA2 | | A2 | PO18/ TIOCC6/ TCLKE | | | |
| F14 | | PA0 | | A0/BC0# | PO16/ TIOCA6 | | | |
| F15 | | PA1 | | A1 | PO17/ TIOCA6/ TIOCB6 | | | |
| G1 | RES# | | | | | | | |
| G2 | XTAL | | | | | | | |
| G3 | | P85 | | | | | | |
| G4 | | P86 | | | | | | |
| G12 | | PA7 | | A7 | PO23/ TIOCA8/ TIOCB8/ TCLKH | | | |
| G13 | | PA6 | | A6 | PO22/ TIOCA8 | | | |
| G14 | | PA4 | | A4 | PO20/ TIOCA7 | | | |
| G15 | | PA5 | | A5 | PO21/ TIOCA7/ TIOCB7/ TCLKG | | | |
| H1 | VCC | | | | | | | |
| H2 | | | NMI | | | | | |
| H3 | EXTAL | | | | | | | |
| H4 | VSS | | | | | | | |
| H12 | | PB0 | | A8 | PO24/ TIOCA9 | | | |
| H13 | VSS | | | | | | | |
| H14 | | PH0 | | | | | | |
| H15 | | PH1 | | | | | | |
| J1 | | PF5 | | | | | | |
| J2 | | PF4 | | | | | | |
| J3 | | PF6 | | | | | | |
| J4 | | P34 | IRQ4-A | | PO12/ TIOCA1 | | | |
| J12 | | P72 | | | | | | |
| J13 | | P71 | | CS4#-C/ CS5#-C/ CS6#-C/ CS7#-C | | | | |
| J14 | VCC | | | | | | | |



Downloaded from Arrow.com.

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| J15 | | P70 | | CS3#-B | | | ADTRG2# | |
| K1 | | P31 | IRQ1-A | | PO9/ TIOCA0/ TIOCB0 | | | |
| K2 | | P30 | IRQ0-A | | PO8/ TIOCA0 | | | |
| K3 | | P32 | IRQ2-A | | PO10/ TIOCC0/ TCLKA-A | | | |
| K4 | | P33 | IRQ3-A | | PO11/ TIOCC0/ TIOCD0/ TCLKB-A | | | |
| K12 | | PB2 | | A10 | PO26/ TIOCC9 | | | |
| K13 | | PB1 | | A9 | PO25/ TIOCA9/ TIOCB9 | | | |
| K14 | | P73 | | | | | | |
| K15 | | P74 | | | | | ADTRG3# | |
| L1 | | PF2 | | | | | | |
| L2 | | PF1 | | | | | | |
| L3 | | PF3 | | | | | | |
| L4 | | PF0 | | | | | | |
| L12 | | PB7 | | A15 | PO31/ TIOCA11/ TIOCB11 | | | |
| L13 | | PB5 | | A13 | PO29/ TIOCA10/ TIOCB10 | | | |
| L14 | | PB4 | | A12 | PO28/ TIOCA10 | | | |
| L15 | | PB3 | | A11 | PO27/ TIOCC9/ TIOCD9 | | | |
| M1 | | P27 | | | PO7/ TIOCA5/ TIOCB5 | SCK1 | | |
| M2 | | P26 | | | PO6/ TIOCA5/ TMO1 | TxD1 | | |
| M3 | VCC | | | | | | | |
| M4 | VSS | | | | | | | |
| M5 | | P14 | IRQ4-B | | TCLKA-B | SDA1 | | |



Downloaded from Arrow.com.

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| M6 | | P37 | | | PO15/ TIOCA2/ TIOCB2/ TCLKD-A | | | |
| M7 | | P57 | | WAIT# | | | | TRDATA3 |
| M8 | | P83 | | | | | | |
| M9 | | P81 | | | | | | TRSYNC |
| M10 | | P51 | | WR1#/BC1# | | | | |
| M11 | | PH4 | | | | | | |
| M12 | | PC7 | | A23/ CS4#-D/ CS7#-D | | TxD5 | | |
| M13 | VSS | | | | | | | |
| M14 | | PC0 | | A16 | | | | |
| M15 | | PB6 | | A14 | PO30/ TIOCA11 | | | |
| N1 | | P25 | | | PO5/ TIOCA4/ TMCI1 | RxD1 | | |
| N2 | | P24 | | | PO4/ TIOCA4/ TIOCB4/ TMRI1 | | | |
| N3 | | P20 | | | PO0/ TIOCA3/ TIOCB3/ TMRI0 | TxD0 | | |
| N4 | | P16 | IRQ6-B | | TCLKC-B | RxD3/SDA0 | | |
| N5 | | P12 | IRQ2-B | | | RxD2 | | |
| N6 | | P36 | | | PO14/ TIOCA2 | | | |
| N7 | | P56 | | | | | | TRDATA2 |
| N8 | VSS | | | | | | | |
| N9 | | P80 | | | | | | |
| N10 | | P50 | | WR0#/WR# | | | | |
| N11 | VSS | | | | | | | |
| N12 | | P76 | IRQ14-A | | | | | |
| N13 | | PH2 | | | | | | |
| N14 | | PC2 | | A18 | | | | |
| N15 | | PC1 | | A17 | | | | |
| P1 | | P23 | | | PO3/ TIOCC3/ TIOCD3 | | | |



Downloaded from Arrow.com.

| Pin No. 176-Pin LFBGA | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| P2 | | P22 | | | PO2/ TIOCC3/ TMO0 | SCK0 | | |
| P3 | PLLVCC | | | | | | | |
| P4 | | P15 | IRQ5-B | | TCLKB-B | SCK3/SCL1 | | |
| P5 | | P11 | IRQ1-B | | | SCK2 | | |
| P6 | | P84 | | | | | | |
| P7 | | P54 | | | | | | TRDATA0 |
| P8 | VCC | | | | | | | |
| P9 | | P52 | | RD# | | | | |
| P10 | | PH6 | | | | | | |
| P11 | VCC | | | | | | | |
| P12 | | P77 | | | | | | |
| P13 | | PC6 | | A22/ CS6#-D | | RxD5 | | |
| P14 | | PC4 | | A20 | | | | |
| P15 | VCC | | | | | | | |
| R1 | | P21 | | | PO1/ TIOCA3/ TMCI0 | RxD0 | | |
| R2 | | P17 | IRQ7-B | | TCLKD-B | TxD3/SCL0 | ADTRG1# | |
| R3 | PLLVSS | | | | | | | |
| R4 | | P13 | IRQ3-B | | | TxD2 | ADTRG0# | |
| R5 | | P10 | IRQ0-B | | | | | |
| R6 | | P35 | | | PO13/ TIOCA1/ TIOCB1/ TCLKC-A | | | |
| R7 | | P55 | | | | | | TRDATA1 |
| R8 | | P82 | | | | | | TRCLK |
| R9 | BCLK | P53 | | | | | | |
| R10 | | PH7 | | | | | | |
| R11 | | PH5 | | | | | | |
| R12 | | PH3 | | | | | | |
| R13 | | P75 | | | | | | |
| R14 | | PC5 | | A21/ CS5#-D | | SCK5 | | |
| R15 | | PC3 | | A19 | | | | |



Downloaded from Arrow.com.

**Table 1.4 List of Pins and Pin Functions (144-Pin LQFP)**

| Pin No. 144-Pin LQFP | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| 1 | | P04 | IRQ12-A | | TMCI3 | TxD4 | | TDI |
| 2 | | P03 | IRQ11-A | | TMRI3 | SCK4 | | TMS |
| 3 | | P67 | | | | | DA1 | |
| 4 | | P66 | | | | | DA0 | |
| 5 | AVSS | | | | | | | |
| 6 | | P02 | IRQ10-A | | TMO2 | SCK6 | | TRST# |
| 7 | | P01 | IRQ9-A | | TMCI2 | RxD6 | | |
| 8 | | P00 | IRQ8-A | | TMRI2 | TxD6 | | |
| 9 | | P65 | IRQ15-A | | | | | |
| 10 | EMLE | | | | | | | |
| 11 | WDTOVF# | | | | | | | TDO |
| 12 | VSS | | | | | | | |
| 13 | MDE | | | | | | | |
| 14 | VCL | | | | | | | |
| 15 | MD1 | | | | | | | |
| 16 | MD0 | | | | | | | |
| 17 | | P86 | | | | | | |
| 18 | | P85 | | | | | | |
| 19 | RES# | | | | | | | |
| 20 | XTAL | | | | | | | |
| 21 | VSS | | | | | | | |
| 22 | EXTAL | | | | | | | |
| 23 | VCC | | | | | | | |
| 24 | | | NMI | | | | | |
| 25 | | P34 | IRQ4-A | | PO12/ TIOCA1 | | | |
| 26 | | P33 | IRQ3-A | | PO11/ TIOCC0/ TIOCD0/ TCLKB-A | | | |
| 27 | | P32 | IRQ2-A | | PO10/ TIOCC0/ TCLKA-A | | | |
| 28 | | P31 | IRQ1-A | | PO9/ TIOCA0/ TIOCB0 | | | |
| 29 | | P30 | IRQ0-A | | PO8/ TIOCA0 | | | |
| 30 | | P27 | | | PO7/ TIOCA5/ TIOCB5 | SCK1 | | |
| 31 | | P26 | | | PO6/ TIOCA5/ TMO1 | TxD1 | | |



Downloaded from Arrow.com.

| Pin No. 144-Pin LQFP | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| 32 | | P25 | | | PO5/ TIOCA4/ TMCI1 | RxD1 | | |
| 33 | | P24 | | | PO4/ TIOCA4/ TIOCB4/ TMRI1 | | | |
| 34 | | P23 | | | PO3/ TIOCC3/ TIOCD3 | | | |
| 35 | | P22 | | | PO2/ TIOCC3/ TMO0 | SCK0 | | |
| 36 | | P21 | | | PO1/ TIOCA3/ TMCI0 | RxD0 | | |
| 37 | | P20 | | | PO0/ TIOCA3/ TIOCB3/ TMRI0 | TxD0 | | |
| 38 | | P17 | IRQ7-B | | TCLKD-B | TxD3/SCL0 | ADTRG1# | |
| 39 | PLLVCC | | | | | | | |
| 40 | | P16 | IRQ6-B | | TCLKC-B | RxD3/SDA0 | | |
| 41 | PLLVSS | | | | | | | |
| 42 | | P15 | IRQ5-B | | TCLKB-B | SCK3/SCL1 | | |
| 43 | | P14 | IRQ4-B | | TCLKA-B | SDA1 | | |
| 44 | | P13 | IRQ3-B | | | TxD2 | ADTRG0# | |
| 45 | | P12 | IRQ2-B | | | RxD2 | | |
| 46 | | P11 | IRQ1-B | | | SCK2 | | |
| 47 | | P10 | IRQ0-B | | | | | |
| 48 | | P37 | | | PO15/ TIOCA2/ TIOCB2/ TCLKD-A | | | |
| 49 | | P36 | | | PO14/ TIOCA2 | | | |
| 50 | | P35 | | | PO13/ TIOCA1/ TIOCB1/ TCLKC-A | | | |
| 51 | | P84 | | | | | | |
| 52 | | P57 | | WAIT# | | | | TRDATA3 |
| 53 | | P56 | | | | | | TRDATA2 |
| 54 | | P55 | | | | | | TRDATA1 |



Downloaded from Arrow.com.

| Pin No. 144-Pin LQFP | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| 55 | | P54 | | | | | | TRDATA0 |
| 56 | | P83 | | | | | | |
| 57 | VSS | | | | | | | |
| 58 | | P82 | | | | | | TRCLK |
| 59 | VCC | | | | | | | |
| 60 | | P81 | | | | | | TRSYNC# |
| 61 | | P80 | | | | | | |
| 62 | BCLK | P53 | | | | | | |
| 63 | | P52 | | RD# | | | | |
| 64 | | P51 | | WR1#/BC1# | | | | |
| 65 | | P50 | | WR0#/WR# | | | | |
| 66 | | P77 | | | | | | |
| 67 | | P76 | IRQ14-A | | | | | |
| 68 | | P75 | | | | | | |
| 69 | | PC7 | | A23/ CS4#-D/ CS7#-D | | TxD5 | | |
| 70 | | PC6 | | A22/ CS6#-D | | RxD5 | | |
| 71 | | PC5 | | A21/ CS5#-D | | SCK5 | | |
| 72 | | PC4 | | A20 | | | | |
| 73 | | PC3 | | A19 | | | | |
| 74 | VCC | | | | | | | |
| 75 | | PC2 | | A18 | | | | |
| 76 | VSS | | | | | | | |
| 77 | | PC1 | | A17 | | | | |
| 78 | | PC0 | | A16 | | | | |
| 79 | | PB7 | | A15 | PO31/ TIOCA11/ TIOCB11 | | | |
| 80 | | PB6 | | A14 | PO30/ TIOCA11 | | | |
| 81 | | PB5 | | A13 | PO29/ TIOCA10/ TIOCB10 | | | |
| 82 | | PB4 | | A12 | PO28/ TIOCA10 | | | |
| 83 | | PB3 | | A11 | PO27/ TIOCC9/ TIOCD9 | | | |
| 84 | | PB2 | | A10 | PO26/ TIOCC9 | | | |



Downloaded from Arrow.com.

| Pin No. 144-Pin LQFP | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| 85 | | PB1 | | A9 | PO25/ TIOCA9/ TIOCB9 | | | |
| 86 | | P74 | | | | | ADTRG3# | |
| 87 | | P73 | | | | | | |
| 88 | | P72 | | | | | | |
| 89 | | P71 | | CS4#-C/ CS5#-C/ CS6#-C/ CS7#-C | | | | |
| 90 | | P70 | | CS3#-B | | | ADTRG2# | |
| 91 | VCC | | | | | | | |
| 92 | | PB0 | | A8 | PO24/ TIOCA9 | | | |
| 93 | VSS | | | | | | | |
| 94 | | PA7 | | A7 | PO23/ TIOCA8/ TIOCB8/ TCLKH | | | |
| 95 | | PA6 | | A6 | PO22/ TIOCA8 | | | |
| 96 | | PA5 | | A5 | PO21/ TIOCA7/ TIOCB7/ TCLKG | | | |
| 97 | | PA4 | | A4 | PO20/ TIOCA7 | | | |
| 98 | | PA3 | | A3 | PO19/ TIOCC6/ TIOCD6/ TCLKF | | | |
| 99 | | PA2 | | A2 | PO18/ TIOCC6/ TCLKE | | | |
| 100 | | PA1 | | A1 | PO17/ TIOCA6/ TIOCB6 | | | |
| 101 | | PA0 | | A0/BC0# | PO16/ TIOCA6 | | | |
| 102 | | PE7 | IRQ7-A | D15 | | | | |
| 103 | | PE6 | IRQ6-A | D14 | | | | |
| 104 | | PE5 | IRQ5-A | D13 | | | | |
| 105 | | PE4 | | D12 | | | | |
| 106 | | PE3 | | D11 | | | | |
| 107 | | PE2 | | D10 | | | | |



Downloaded from Arrow.com.

| Pin No. 144-Pin LQFP | Power Supply Clock System Control | I/O Port | Interrupt | External Bus | Timer | Communi-cation | Analog | On-Chip Emulator |
|---|---|---|---|---|---|---|---|---|
| 108 | | PE1 | | D9 | | | | |
| 109 | | PE0 | | D8 | | | | |
| 110 | | PD7 | | D7 | | | | |
| 111 | | PD6 | | D6 | | | | |
| 112 | | PD5 | | D5 | | | | |
| 113 | | PD4 | | D4 | | | | |
| 114 | | P64 | | CS4#-B | | | | |
| 115 | | P63 | | CS3#-A/ CS7#-A | | | | |
| 116 | | P62 | | CS2#-A/ CS6#-A | | | | |
| 117 | | P61 | | CS1#/ CS2#-B/ CS5#-A/ CS6#-B/ CS7#-B | | | | |
| 118 | | P60 | | CS0#/ CS4#-A/ CS5#-B | | | | |
| 119 | | PD3 | | D3 | | | | |
| 120 | | PD2 | | D2 | | | | |
| 121 | | PD1 | | D1 | | | | |
| 122 | | PD0 | | D0 | | | | |
| 123 | | P97 | | | | | AN15 | |
| 124 | | P96 | | | | | AN14 | |
| 125 | | P95 | | | | | AN13 | |
| 126 | | P94 | | | | | AN12 | |
| 127 | | P93 | | | | | AN11 | |
| 128 | | P92 | | | | | AN10 | |
| 129 | | P91 | | | | | AN9 | |
| 130 | VSS | | | | | | | |
| 131 | | P90 | | | | | AN8 | |
| 132 | VCC | | | | | | | |
| 133 | | P47 | IRQ15-B | | | | AN7 | |
| 134 | | P46 | IRQ14-B | | | | AN6 | |
| 135 | | P45 | IRQ13-B | | | | AN5 | |
| 136 | | P44 | IRQ12-B | | | | AN4 | |
| 137 | | P43 | IRQ11-B | | | | AN3 | |
| 138 | | P42 | IRQ10-B | | | | AN2 | |
| 139 | | P41 | IRQ9-B | | | | AN1 | |
| 140 | VREFL | | | | | | | |
| 141 | | P40 | IRQ8-B | | | | AN0 | |
| 142 | VREFH | | | | | | | |
| 143 | AVCC | | | | | | | |
| 144 | | P05 | IRQ13-A | | TMO3 | RxD4 | | TCK |



Downloaded from Arrow.com.

## 1.5 Pin Functions

Table 1.5 lists the pin functions.

**Table 1.5 Pin Functions**

| Classifications | Pin Name | I/O | Description |
|---|---|---|---|
| Power supply | VCC | Input | Power supply pin. Connect it to the system power supply. |
| | VCL | Input | Connect this pin to VSS via a 0.1-μF capacitor. The capacitor should be placed close to the pin. |
| | VSS | Input | Ground pin. Connect it to the system power supply (0 V). |
| | PLLVCC | Input | Power supply pin for the PLL circuit. Connect it to the system power supply. |
| | PLLVSS | Input | Ground pin for the PLL circuit |
| Clock | XTAL | Input | Pins for a crystal resonator. An external clock signal can be input through the EXTAL pin. |
| | EXTAL | Input | |
| | BCLK | Output | Outputs the system clock for external devices. |
| Operating mode control | MD0, MD1, MDE | Input | Pins for setting the operating mode. The signal levels on these pins must not be changed during operation. |
| System control | RES# | Input | Reset signal input pin. This LSI enters the reset state when this signal goes low. |
| | EMLE | Input | Input pin to enable on-chip emulator signal. When the on-chip emulator is used, this pin should be driven high. When not used, it should be driven low. |
| | BSCANP | Input | Input pin to enable boundary-scan signal. When this pin is driven high, the boundary scan is enabled. When the boundary scan is not used, this pin should be driven low. |
| On-chip emulator | TRST# | Input | On-chip emulator pins.<br>When the EMLE pin is driven high, these pins are dedicated for the on-chip emulator. |
| | TMS | Input | |
| | TDI | Input | |
| | TCK | Input | |
| | TDO | Output | |
| | TRCLK | Output | This pin outputs the clock for synchronization with the trace data. |
| | TRSYNC | Output | This pin indicates that output from the TRDATA0 to TRDATA3 pins is valid. |
| | TRDATA0 to TRDATA3 | Output | These pins output the trace information. |
| Address bus | A0 to A23*[1] | Output | Output pins for the address |
| Data bus | D0 to D15 | I/O | Input and output pins for the bidirectional data bus |



Downloaded from Arrow.com.

| Classifications | Pin Name | I/O | Description |
|---|---|---|---|
| Bus control | RD# | Output | Strobe signal which indicates that reading from the external address space is in progress. |
| | WR0# | Output | Strobe signal which indicates that the lower-order byte (D0 to D7) is valid in writing to the external address space, in byte strobe mode. |
| | WR1# | Output | Strobe signal which indicates that the higher-order byte (D8 to D15) is valid in writing to the external address space, in byte strobe mode. |
| | WR# | Output | Strobe signal which indicates that writing to the external address space is in progress, in 1-write strobe mode. |
| | BC0# *1, *2 | Output | Strobe signal which indicates that the lower-order byte (D0 to D7) is valid in access to the external address space, in 1-write strobe mode. |
| | BC1# *2 | Output | Strobe signal which indicates that the higher-order byte (D8 to D15) is valid in access to the external address space, in 1-write strobe mode. |
| | CS0#, CS1#<br>CS2#-A/CS2#-B<br>CS3#-A/CS3#-B<br>CS4#-A/CS4#-B/<br>CS4#-C/CS4#-D<br>CS5#-A/CS5#-B/<br>CS5#-C/CS5#-D<br>CS6#-A/CS6#-B/<br>CS6#-C/CS6#-D<br>CS7#-A/CS7#-B/<br>CS7#-C/CS7#-D | Output | Select signals for areas 0 to 7 |
| | WAIT# | Input | Requests wait cycles in access to the external address space |



Downloaded from Arrow.com.

| Classifications | Pin Name | I/O | Description |
|---|---|---|---|
| Interrupt | NMI | Input | Non-maskable interrupt request signal |
| | IRQ0-A/IRQ0-B<br>IRQ1-A/IRQ1-B<br>IRQ2-A/IRQ2-B<br>IRQ3-A/IRQ3-B<br>IRQ4-A/IRQ4-B<br>IRQ5-A/IRQ5-B<br>IRQ6-A/IRQ6-B<br>IRQ7-A/IRQ7-B<br>IRQ8-A/IRQ8-B<br>IRQ9-A/IRQ9-B<br>IRQ10-A/IRQ10-B<br>IRQ11-A/IRQ11-B<br>IRQ12-A/IRQ12-B<br>IRQ13-A/IRQ13-B<br>IRQ14-A/IRQ14-B<br>IRQ15-A/IRQ15-B | Input | Maskable request signals |
| 16-bit timer pulse unit | TIOCA0, TIOCB0<br>TIOCC0, TIOCD0 | I/O | Signals for TGRA0 to TGRD0. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA1, TIOCB1 | I/O | Signals for TGRA1 and TGRB1. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA2, TIOCB2 | I/O | Signals for TGRA2 and TGRB2. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA3, TIOCB3<br>TIOCC3, TIOCD3 | I/O | Signals for TGRA3 to TGRD3. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA4, TIOCB4 | I/O | Signals for TGRA4 and TGRB4. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA5, TIOCB5 | I/O | Signals for TGRA5 and TGRB5. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA6, TIOCB6<br>TIOCC6, TIOCD6 | I/O | Signals for TGRA6 to TGRD6. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA7, TIOCB7 | I/O | Signals for TGRA7 and TGRB7. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA8, TIOCB8 | I/O | Signals for TGRA8 and TGRB8. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA9, TIOCB9<br>TIOCC9, TIOCD9 | I/O | Signals for TGRA9 to TGRD9. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA10, TIOCB10 | I/O | Signals for TGRA10 and TGRB10. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |
| | TIOCA11, TIOCB11 | I/O | Signals for TGRA11 and TGRB11. These pins are used as input capture inputs, output compare outputs, or PWM outputs. |



Downloaded from Arrow.com.

| Classifications | Pin Name | I/O | Description |
|---|---|---|---|
| 16-bit timer pulse unit | TCLKA-A/TCLKA-B<br>TCLKB-A/TCLKB-B<br>TCLKC-A/TCLKC-B<br>TCLKD-A/TCLKD-B<br>TCLKE, TCLKF<br>TCLKG, TCLKH | Input | Input pins for external clock signals |
| Programmable pulse generator | PO0 to PO31 | Output | Output pins for the pulse signals |
| 8-bit timer | TMO0 to TMO3 | Output | Output pins for the compare match signals |
| | TMCI0 to TMCI3 | Input | Input pins for the external clock signals that drive for the counters |
| | TMRI0 to TMRI3 | Input | Input pins for the counter-reset signals |
| Watchdog timer | WDTOVF# | Output | Output pin for the counter-overflow signal in watchdog-timer mode |
| Serial communication interface | TxD0, TxD1, TxD2, TxD3, TxD4, TxD5, TxD6 | Output | Output pins for data transmission |
| | RxD0, RxD1, RxD2, RxD3, RxD4, RxD5, RxD6 | Input | Input pins for data reception |
| | SCK0, SCK1, SCK2, SCK3, SCK4, SCK5, SCK6 | I/O | Input/output pins for clock signals |
| $I^2C$ bus interface | SCL0, SCL1 | I/O | Input/output pins for RIIC clocks. Bus can be directly driven by the NMOS open drain output. |
| | SDA0, SDA1 | I/O | Input/output pins for RIIC data. Bus can be directly driven by the NMOS open drain output. |
| A/D converter | AN0 to AN15 | Input | Input pins for the analog signals to be processed by the A/D converter |
| | ADTRG0# to ADTRG3# | Input | Input pins for the external trigger signals that start the A/D conversion |
| D/A converter | DA0, DA1 | Output | Output pins for the analog signals from the D/A converter |



Downloaded from Arrow.com.

| Classifications | Pin Name | I/O | Description |
|---|---|---|---|
| Analog power supply | AVCC | Input | Analog power supply pin for the A/D and D/A converters. When the A/D and D/A converters are not in use, connect this pin to the system power supply. |
| | AVSS | Input | Ground pin for the A/D and D/A converters. Connect this pin to the system power supply (0 V). |
| | VREFH | Input | Reference power supply pin for the A/D and D/A converters. When the A/D and D/A converters are not in use, connect this pin to the system power supply. |
| | VREFL | Input | Reference ground pin for the A/D and D/A converters. Make sure to connect this pin to the analog reference power supply (0 V). When the A/D and D/A converters are not in use, connect this pin to the system power supply (0 V). For details, see section 23.6.7, Ranges of Settings for Analog Power Supply and Other Pins. |
| I/O ports | P00 to P05 | I/O | 6-bit input/output pins |
| | P10 to P17 | I/O | 8-bit input/output pins |
| | P20 to P27 | I/O | 8-bit input/output pins |
| | P30 to P37 | I/O | 8-bit input/output pins |
| | P40 to P47 | I/O | 8-bit input/output pins |
| | P50 to P57 | I/O | 8-bit input/output pins. (P53 is an input-only pin.) |
| | P60 to P67 | I/O | 8-bit input/output pins |
| | P70 to P77 | I/O | 8-bit input/output pins |
| | P80 to P86 | I/O | 7-bit input/output pins |
| | P90 to P97 | I/O | 8-bit input/output pins |
| | PA0 to PA7 | I/O | 8-bit input/output pins |
| | PB0 to PB7 | I/O | 8-bit input/output pins |
| | PC0 to PC7 | I/O | 8-bit input/output pins |
| | PD0 to PD7 | I/O | 8-bit input/output pins |
| | PE0 to PE7 | I/O | 8-bit input/output pins |
| | PF0 to PF6 | I/O | 7-bit input/output pins |
| | PG0 to PG7 | I/O | 8-bit input/output pins |
| | PH0 to PH7 | I/O | 8-bit input/output pins |

Note 1: The A0 and BC0# pin functions are multiplexed on the same pin: the A0 pin is valid in byte-write mode and the BC0# pin becomes valid in single write-strobe mode. The setting for an eight-bit external bus width is prohibited in single write-strobe mode. For other multiplexed pin functions, refer to section 14, I/O Ports.

Note 2: The BC0# and BC1# signals are valid in both reading and writing.



Downloaded from Arrow.com.

# 2. CPU

The RX610 Group is an MCU with the high-speed, high-performance RX CPU as its core.

A variable-length instruction format has been adopted for the RX CPU. Allocating the more frequently used instructions to the shorter instruction lengths facilitates the development of efficient programs that take up less memory.

The CPU has 73 basic instructions and 8 floating-point operation instructions, and 9 DSP instructions, for a total of 90 instructions. It has 10 addressing modes and caters to register–register operations, register–memory operations, immediate–register operations, immediate–memory operations, memory–memory transfer, and bitwise operations. High-speed operation was realized by achieving execution in a single cycle not only for register–register operations, but also for other types of multiple instructions. The CPU includes an internal multiplier and an internal divider for high-speed multiplication and division.

The RX CPU has a five-stage pipeline for processing instructions. The stages are instruction fetching, instruction decoding, execution, memory access, and write-back. In cases where pipeline processing is drawn-out by memory access, subsequent operations may in fact be executed earlier. By adopting "out-of-order completion" of this kind, the execution of instructions is controlled to optimize numbers of clock cycles.

## 2.1 Features

- High instruction execution rate: One instruction in one clock cycle
- Address space: 4-Gbyte linear
- Register set of the CPU
  General purpose: Sixteen 32-bit registers
  Control: Nine 32-bit registers
  Accumulator: One 64-bit register
- Basic instructions: 73 (arithmetic/logic instructions, data-transfer instructions, branch instructions, bit-manipulation instructions, string-manipulation instructions, and system-manipulation instructions)
  Relative branch instructions to suit branch distances
  Variable-length instruction format (lengths from one to eight bytes)
  Short formats for frequently used instructions
- Floating-point operation instructions: 8
- DSP instructions (as an optional function): 9
  Supports 16-bit x 16-bit multiplication and multiply-and-accumulate operations.
  Rounds the data in the accumulator.
- Addressing modes: 10
- Five-stage pipeline
  Adoption of out-of-order completion
- Processor modes
  A supervisor mode and a user mode are supported.
- Floating-point operation unit
  Supports single-precision (32-bit) floating point
  Supports data types and exceptions in conformance with the IEEE754 standard
- Data arrangement
  Selectable as little endian or big endian



Downloaded from Arrow.com.

## 2.2 Register Set of the CPU

The RX CPU has sixteen general-purpose registers, nine control registers, and one accumulator used for DSP instructions.

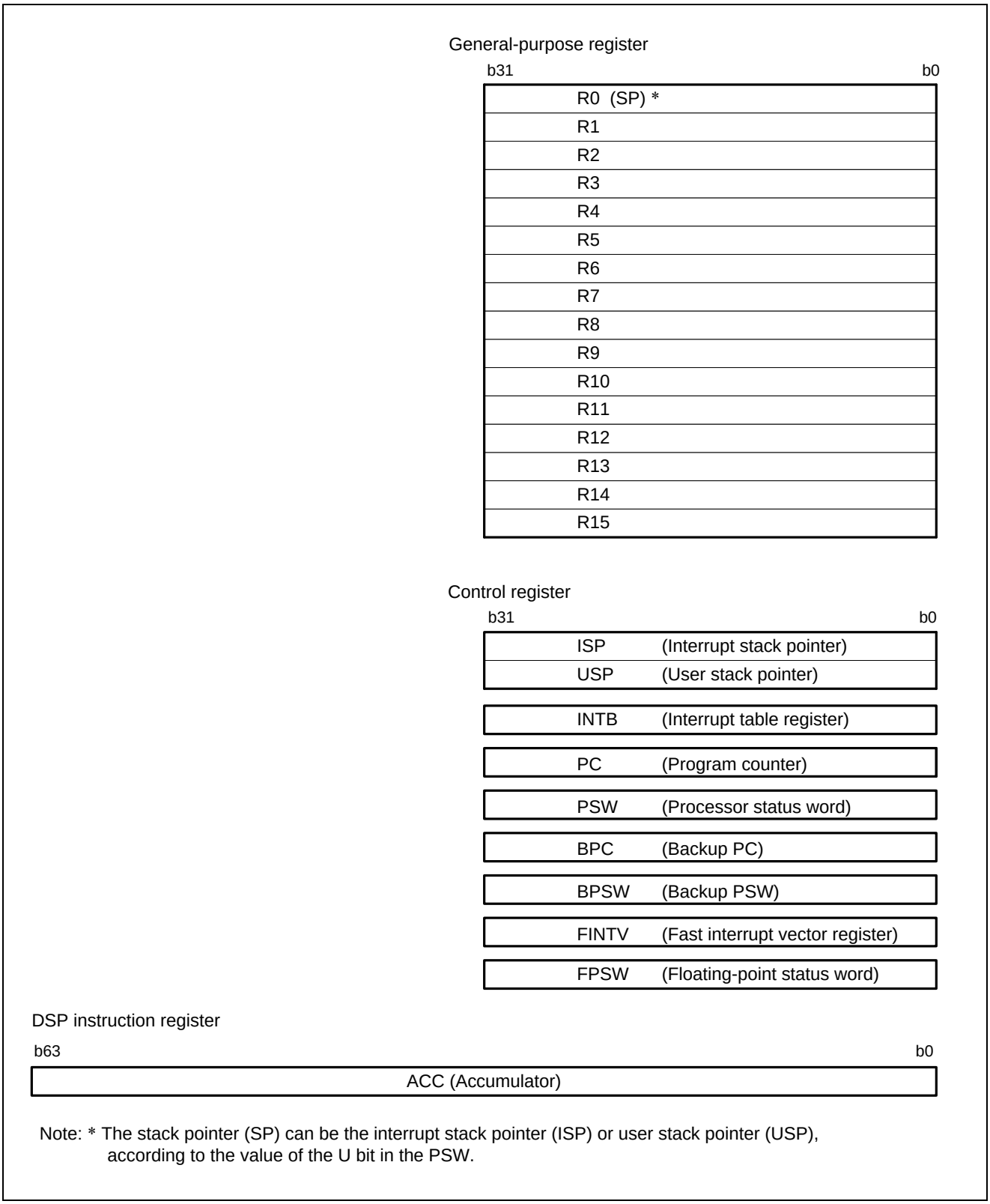



**Figure 2.1 Register Set of the CPU**



Downloaded from Arrow.com.

### 2.2.1 General-Purpose Registers (R0 to R15)

This CPU has sixteen general-purpose registers (R0 to R15). R1 to R15 can be used as data registers or address registers.

R0, a general-purpose register, also functions as the stack pointer (SP). The stack pointer is switched to operate as the interrupt stack pointer (ISP) or user stack pointer (USP) by the value of the stack pointer select bit (U) in the processor status word (PSW).

### 2.2.2 Control Registers

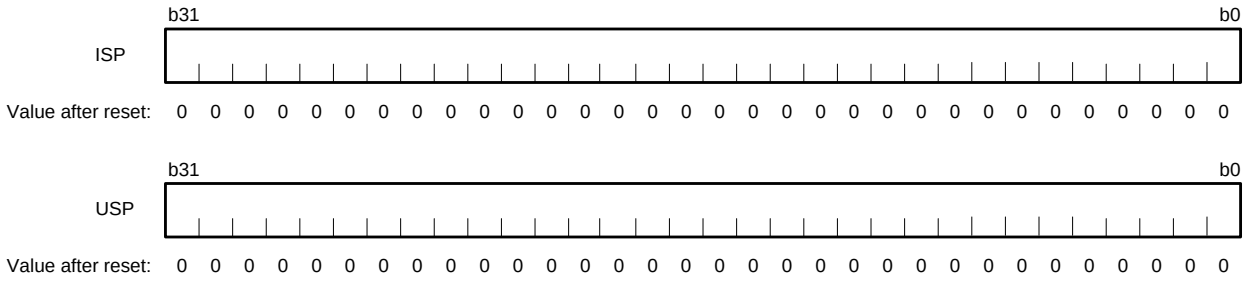
This CPU has the following nine control registers.

- Interrupt stack pointer (ISP)
- User stack pointer (USP)
- Interrupt table register (INTB)
- Program counter (PC)
- Processor status word (PSW)
- Backup PC (BPC)
- Backup PSW (BPSW)
- Fast interrupt vector register (FINTV)
- Floating-point status word (FPSW)



Downloaded from Arrow.com.

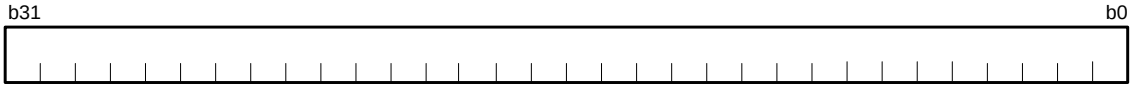
### 2.2.2.1 Interrupt Stack Pointer (ISP)/User Stack Pointer (USP)




The stack pointer (SP) can be either of two types, the interrupt stack pointer (ISP) or the user stack pointer (USP). Whether the stack pointer operates as the ISP or USP depends on the value of the stack pointer select bit (U) in the processor status word (PSW).

Set the ISP or USP to a multiple of four, as this reduces the numbers of cycles required to execute interrupt sequences and instructions entailing stack manipulation.

### 2.2.2.2 Interrupt Table Register (INTB)




Value after reset: Undefined

The interrupt table register (INTB) specifies the address where the relocatable vector table starts.

### 2.2.2.3 Program Counter (PC)




Value after reset: Value allocated to addresses in the range from FFFFFFFCh to FFFFFFFFh

The program counter (PC) indicates the address of the instruction being executed.



Downloaded from Arrow.com.

### 2.2.2.4 Processor Status Word (PSW)

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | IPL[2:0] * | | | — | — | — | PM | — | — | U | I |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | O | S | Z | C |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: * The MVTIPL instruction is not supported in the RX610 Group. When writing to PSW.IPL[2:0], use the MVTC instruction.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | C | Carry Flag | 0: No carry has occurred.<br>1: A carry has occurred. | R/W |
| b1 | Z | Zero Flag | 0: Result is non-zero.<br>1: Result is 0. | R/W |
| b2 | S | Sign Flag | 0: Result is a positive value or 0.<br>1: Result is a negative value. | R/W |
| b3 | O | Overflow Flag | 0: No overflow has occurred.<br>1: An overflow has occurred. | R/W |
| b15 to b4 | — | Reserved | The value read is always 0. When writing, write 0 to these bits. | R/W |
| b16 | I*1 | Interrupt Enable | 0: Interrupt disabled.<br>1: Interrupt enabled. | R/W |
| b17 | U*1 | Stack Pointer Select | 0: Interrupt stack pointer (ISP) is selected.<br>1: User stack pointer (USP) is selected. | R/W |
| b19, b18 | — | Reserved | The value read is always 0. When writing, write 0 to these bits. | R/W |
| b20 | PM*1*2*3 | Processor Mode Select | 0: Supervisor mode is selected.<br>1: User mode is selected. | R/W |
| b23 to b21 | — | Reserved | The value read is always 0. When writing, write 0 to these bits. | R/W |
| b26 to b24 | IPL[2:0]*1*4 | Processor Interrupt Priority Level | b26 b24<br>0 0 0: Priority level 0 (lowest)<br>0 0 1: Priority level 1<br>0 1 0: Priority level 2<br>0 1 1: Priority level 3<br>1 0 0: Priority level 4<br>1 0 1: Priority level 5<br>1 1 0: Priority level 6<br>1 1 1: Priority level 7 (highest) | R/W |
| b31 to b27 | — | Reserved | The value read is always 0. When writing, write 0 to these bits. | R/W |



Downloaded from Arrow.com.

Notes: 1. In user mode, writing to the IPL[2:0], PM, U, and I bits by an MVTC or a POPC instruction is ignored.

2. In supervisor mode, writing to the PM bit by an MVTC or a POPC instruction is ignored, but writing to the other bits is possible.

3. Switching from supervisor mode to user mode requires execution of an RTE instruction after having set the PM bit in the PSW saved on the stack to 1 or executing an RTFI instruction after having set the PM bit in the BPSW to 1.

4. The MVTIPL instruction is not supported in the RX610 Group. When writing to PSW.IPL[2:0], use the MVTC instruction.

The processor status word (PSW) indicates results of instruction execution or the state of the CPU.

### C Flag (Carry Flag)

This flag indicates whether a carry, borrow, or shift-out has occurred as the result of an operation.

### Z Flag (Zero Flag)

This flag indicates that the result of an operation was 0.

### S Flag (Sign Flag)

This flag indicates that the result of an operation was negative.

### O Flag (Overflow Flag)

This flag indicates that an overflow occurred during an operation.

### I Bit (Interrupt Enable)

This bit enables interrupt requests. When an exception is accepted, the value of this bit becomes 0.

### U Bit (Stack Pointer Select)

This bit specifies the stack pointer as either the ISP or USP. When an exception request is accepted, this bit is set to 0. When the processor mode is switched from supervisor mode to user mode, this bit is set to 1.

### PM Bit (Processor Mode Select)

This bit specifies the processor mode. When an exception is accepted, the value of this bit becomes 0.

### IPL[2:0] Bits (Processor Interrupt Priority Level)

The IPL[2:0] bits specify the processor interrupt priority level as one of eight levels from zero to seven, wherein priority level zero is the lowest and priority level seven the highest. When the priority level of a requested interrupt is higher than the processor interrupt priority level, the interrupt is enabled. Setting the IPL[2:0] bits to level seven (111b) disables all interrupt requests. The IPL[2:0] bits are set to level seven (111b) when a non-maskable interrupt is generated. When interrupts in general are generated, the bits are set to the priority levels of accepted interrupts.



Downloaded from Arrow.com.

### 2.2.2.5 Backup PC (BPC)




Value after reset: Undefined

The backup PC (BPC) is provided to speed up response to interrupts.

After a fast interrupt has been generated, the contents of the program counter (PC) are saved in the BPC.

### 2.2.2.6 Backup PSW (BPSW)




Value after reset: Undefined

The backup PSW (BPSW) is provided to speed up response to interrupts.

After a fast interrupt has been generated, the contents of the processor status word (PSW) are saved in the BPSW. The allocation of bits in the BPSW corresponds to that in the PSW.

### 2.2.2.7 Fast Interrupt Vector Register (FINTV)




Value after reset: Undefined

The fast interrupt vector register (FINTV) is provided to speed up response to interrupts.

The FINTV specifies a branch destination address when a fast interrupt has been generated.



Downloaded from Arrow.com.

### 2.2.2.8 Floating-Point Status Word (FPSW)

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FS | FX | FU | FZ | FO | FV | — | — | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | EX | EU | EZ | EO | EV | — | DN | CE | CX | CU | CZ | CO | CV | RM[1 0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | RM[1:0] | Floating-Point Rounding-Mode Setting | b1 b0<br>0 0: Rounding to the nearest value<br>0 1: Rounding to 0<br>1 0: Rounding to +∞<br>1 1: Rounding to –∞ | R/W |
| b2 | CV | Invalid Operation Cause Flag | 0: No invalid operation has been encountered.<br>1: Invalid operation has been encountered. | R/(W)*1 |
| b3 | CO | Overflow Cause Flag | 0: No overflow has occurred.<br>1: Overflow has occurred. | R/(W)*1 |
| b4 | CZ | Division-by-Zero Cause Flag | 0: No division-by-zero has occurred.<br>1: Division-by-zero has occurred. | R/(W)*1 |
| b5 | CU | Underflow Cause Flag | 0: No underflow has occurred.<br>1: Underflow has occurred. | R/(W)*1 |
| b6 | CX | Inexact Cause Flag | 0: No inexact exception has been generated.<br>1: Inexact exception has been generated. | R/(W)*1 |
| b7 | CE | Un-Implemented Processing Cause Flag | 0: No un-implemented processing has been encountered.<br>1: Un-implemented process has been encountered. | R/(W)*1 |
| b8 | DN | 0 Flush Bit of Denormalized Number | 0: A denormalized number is handled as a denormalized number.<br>1: A denormalized number is handled as 0.*2 | R/W |
| b9 | — | Reserved | The value read is always 0. When writing, write 0 to this bit. | R/W |
| b10 | EV | Invalid Operation Exception Enable | 0: Invalid operation exception is masked.<br>1: Invalid operation exception is enabled. | R/W |
| b11 | EO | Overflow Exception Enable | 0: Overflow exception is masked.<br>1: Overflow exception is enabled. | R/W |
| b12 | EZ | Division-by-Zero Exception Enable | 0: Division-by-zero exception is masked.<br>1: Division-by-zero exception is enabled. | R/W |
| b13 | EU | Underflow Exception Enable | 0: Underflow exception is masked.<br>1: Underflow exception is enabled. | R/W |
| b14 | EX | Inexact Exception Enable | 0: Inexact exception is masked.<br>1: Inexact exception is enabled. | R/W |
| b25 to b15 | — | Reserved | The value read is always 0. When writing, write 0 to these bits. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b26 | FV*3 | Invalid Operation Flag | 0: No invalid operation has been encountered.<br>1: Invalid operation has been encountered.*8 | R/W |
| b27 | FO*4 | Overflow Flag | 0: No overflow has occurred.<br>1: Overflow has occurred.*8 | R/W |
| b28 | FZ*5 | Division-by-Zero Flag | 0: No division-by-zero has occurred.<br>1: Division-by-zero has occurred.*8 | R/W |
| b29 | FU*6 | Underflow Flag | 0: No underflow has occurred.<br>1: Underflow has occurred.*8 | R/W |
| b30 | FX*7 | Inexact Flag | 0: No inexact exception has been generated.<br>1: Inexact exception has been generated.*8 | R/W |
| b31 | FS | Floating-Point Error Summary Flag | This bit reflects the logical OR of the FU, FZ, FO, and FV flags. | R |

Notes: 1. Writing 0 to the bit clears it. Writing 1 to the bit does not affect its value.
2. Positive denormalized numbers are treated as +0, negative denormalized numbers as –0.
3. When the EV bit is set to 0, the FV flag is enabled.
4. When the EO bit is set to 0, the FO flag is enabled.
5. When the EZ bit is set to 0, the FZ flag is enabled.
6. When the EU bit is set to 0, the FU flag is enabled.
7. When the EX bit is set to 0, the FX flag is enabled.
8. Once the bit has been set to 1, this value is retained until it is cleared to 0 by software.

The floating-point status word (FPSW) indicates the results of floating-point operations.

When an exception handling enable bit (Ej) enables the exception handling (Ej = 1), the corresponding Cj flag indicates the source of the exception within the exception handling routine. If the exception handling is masked (Ej = 0), check the Fj flag at the end of a series of processing whether an exception is generated or not. The Fj flag is the accumulation type flag (j = X, U, Z, O, or V).

### RM[1:0] Bits (Floating-Point Rounding-Mode Setting)

These bits specify the floating-point rounding-mode.



Downloaded from Arrow.com.

**Explanation of Floating-Point Rounding Modes**

- Rounding to the nearest value (the default behavior): An inexact result is rounded to the available value that is closest to the result which would be obtained with an infinite number of digits. If two available values are equally close, rounding is to the even alternative.
- Rounding towards 0: An inexact result is rounded to the smallest available absolute value, i.e. in the direction of zero (simple truncation).
- Rounding towards +∞: An inexact result is rounded to the nearest available value in the direction of positive infinity.
- Rounding towards –∞: An inexact result is rounded to the nearest available value in the direction of negative infinity.

(1) Rounding to the nearest value is specified as the default mode and returns the most accurate value.

(2) Modes such as rounding towards 0, rounding towards +∞, and rounding towards –∞ are used to ensure precision when interval arithmetic is employed.

### CV Flag (Invalid Operation Cause Flag), CO Flag (Overflow Cause Flag), CZ Flag (Division-by-Zero Cause Flag), CU Flag (Underflow Cause Flag), CX Flag (Inexact Cause Flag), and CE Flag (Un-Implemented Processing Cause Flag)

Floating-point exceptions include the five specified in the IEEE754 standard, namely overflow, underflow, inexact, division-by-zero, and invalid operation. For a further floating-point exception that is generated upon detection of unimplemented processing, the corresponding flag (CE) is set to 1.

- The bit that has been set to 1 is cleared to 0 when the FPU instruction is executed.
- When 0 is written to the bit by the MVTC and POPC instructions, the bit is set to 0; the bit retains the previous value when 1 is written by the instruction.

### DN Flag (0 Flush Bit of Denormalized Number)

When this bit is set to 0, a denormalized number is handled as a denormalized number. When this bit is set to 1, a denormalized number is handled as 0.

### EV Bit (Invalid Operation Exception Enable), EO Bit (Overflow Exception Enable), EZ Bit (Division-by-Zero Exception Enable), EU Bit (Underflow Exception Enable), and EX Bit (Inexact Exception Enable)

When any of five floating-point exceptions specified in the IEEE754 standard is generated by the FPU instruction, the bit decides whether the CPU will start handling the exception. When the bit is set to 0, the exception handling is masked; when the bit is set to 1, the exception handling is enabled.



Downloaded from Arrow.com.

#### FV Flag (Invalid Operation flag), FO Flag (Overflow Flag), FZ Flag (Division-by-Zero Flag), FU Flag (Underflow Flag), and FX Flag (Inexact Flag)

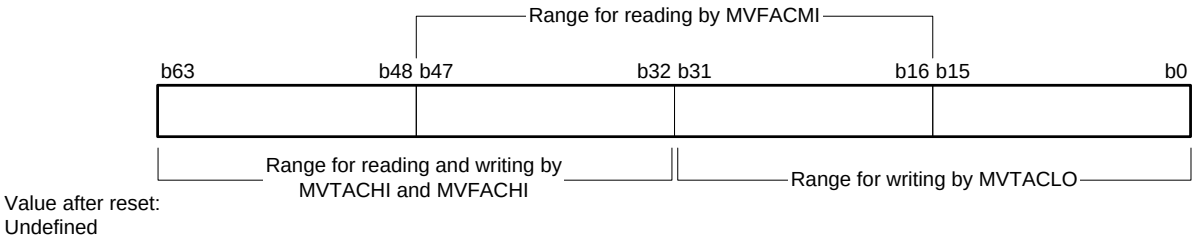
While the exception handling enable bit (Ej) is 0 (exception handling is masked), if any of five floating-point exceptions specified in the IEEE754 standard is generated, the corresponding bit is set to 1.

- When Ej is 1 (exception handling is enabled), the value of the flag remains.
- When the corresponding flag is set to 1, it remains 1 until it is cleared to 0 by software. (Accumulation flag)

#### FS Flag (Floating-Point Error Summary Flag)

This bit reflects the logical OR of the FU, FZ, FO, and FV flags.

### 2.2.2.9 Accumulator (ACC)




The accumulator (ACC) is a 64-bit register used for DSP instructions. The accumulator is also used for the multiply and multiply-and-accumulate instructions; EMUL, EMULU, FMUL, MUL, and RMPA, in which case the prior value in the accumulator is modified by execution of the instruction.

Use the MVTACHI and MVTACLO instructions for writing to the accumulator. The MVTACHI and MVTACLO instructions write data to the higher-order 32 bits (bits 63 to 32) and the lower-order 32 bits (bits 31 to 0), respectively.

Use the MVFACHI and MVFACMI instructions for reading data from the accumulator. The MVFACHI and MVFACMI instructions read data from the higher-order 32 bits (bits 63 to 32) and the middle 32 bits (bits 47 to 16), respectively.



Downloaded from Arrow.com.

## 2.3 Processor Mode

The RX CPU supports two processor modes, supervisor and user. These processor modes enable the realization of a hierarchical CPU resource protection.

Each processor mode imposes a level on rights of access to the CPU resource and the instructions that can be executed. Supervisor mode carries greater rights than those of user mode.

The initial state after a reset is supervisor mode.

### 2.3.1 Supervisor Mode

In supervisor mode, all CPU resources are accessible and all instructions are available. However, writing to the processor mode select bit (PM) in the processor status word (PSW) by executing an MVTC or a POPC instruction will be ignored. For details on how to write to the PM bit, refer to section 2.2.2.4, Processor Status Word (PSW).

### 2.3.2 User Mode

In user mode, write access to the CPU resources listed below is restricted. The restriction applies to any instruction capable of write access.

- Some bits (bits IPL[2:0], PM, U, and I) in the processor status word (PSW)
- Interrupt stack pointer (ISP)
- Interrupt table register (INTB)
- Backup PSW (BPSW)
- Backup PC (BPC)
- Fast interrupt vector register (FINTV)

### 2.3.3 Privileged Instruction

Privileged instructions can only be executed in supervisor mode. Executing a privileged instruction in user mode produces a privileged instruction exception. Privileged instructions include the RTFI, RTE, and WAIT instructions.

### 2.3.4 Switching Between Processor Modes

Manipulating the processor mode select bit (PM) in the processor status word (PSW) switches the processor mode. However, rewriting to the PM bit by executing an MVTC or a POPC instruction is prohibited. Switch the processor mode by following the procedures described below.

(1) Switching from user mode to supervisor mode

After an exception has been generated, the PM bit in the PSW is set to 0 and the CPU switches to supervisor mode. The pre-processing by hardware is executed in supervisor mode. The state of the processor mode before the exception was generated is retained in the PM bit in the copy of the PSW that is saved on the stack.

(2) Switching from supervisor mode to user mode

Executing an RTE instruction when the value of the copy of the PM bit in the PSW that has been preserved on the stack is "1" or an RTFI instruction when the value of the copy of the PM bit in the PSW that has been preserved in the backup PSW (BPSW ) is "1" causes a transition to user mode. In the transition to user mode, the value of the stack pointer designation bit (the U bit in the PSW) becomes "1".



Downloaded from Arrow.com.

## 2.4 Data Types

The RX CPU can handle four types of data: integer, floating-point, bit, and string.

### 2.4.1 Integer

An integer can be signed or unsigned. For signed integers, negative values are represented by two's complements.

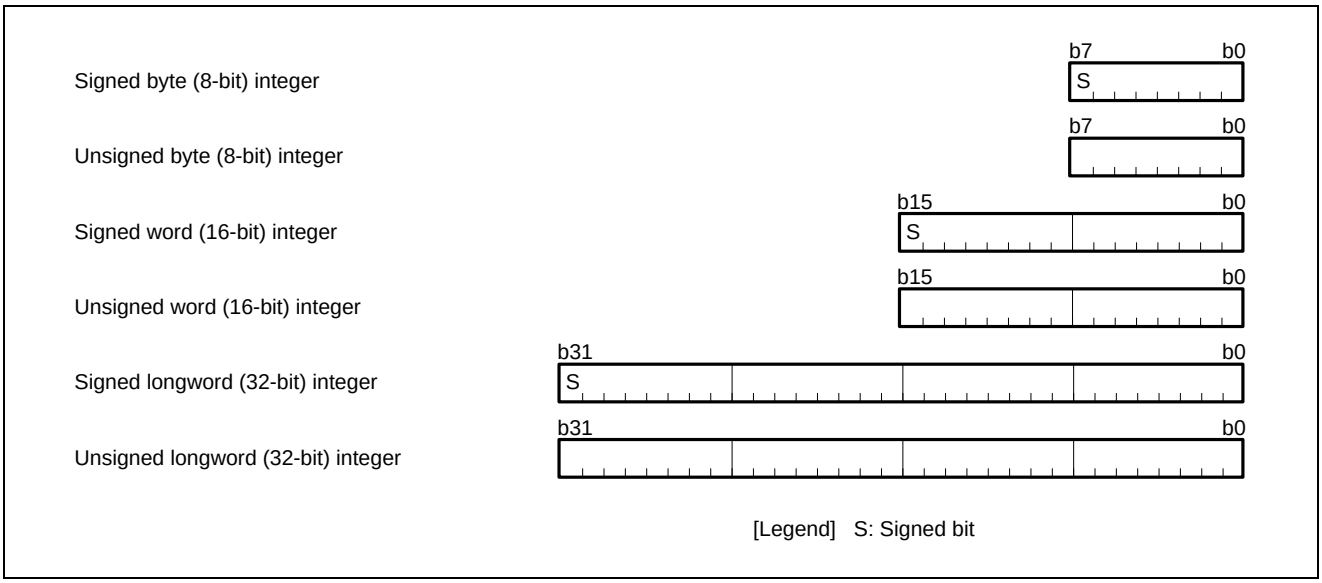



**Figure 2.2 Integer**

### 2.4.2 Floating-Point

Floating-point support is for the single-precision floating-point type specified in IEEE754; operands of this type can be used in eight floating-point operation instructions: FADD, FCMP, FDIV, FMUL, FSUB, FTOI, ITOF, and ROUND.

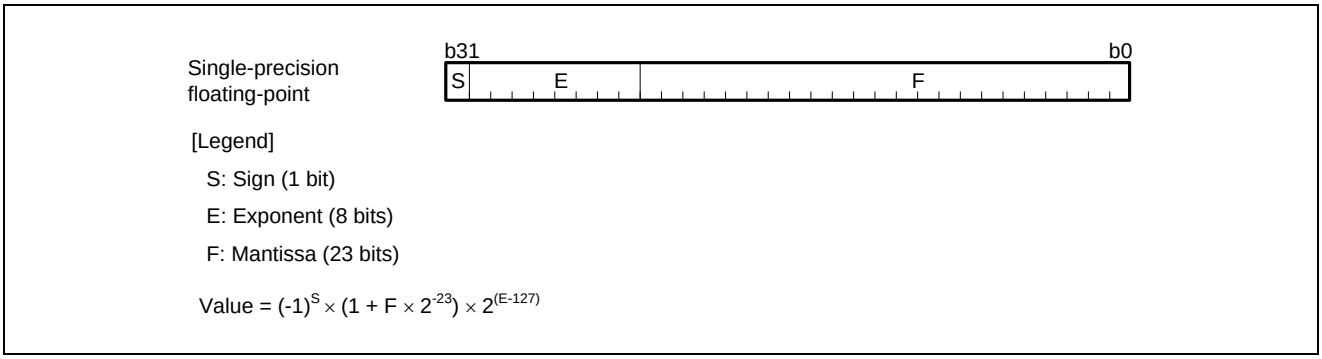



**Figure 2.3 Floating-Point**



Downloaded from Arrow.com.

The floating-point format supports the values listed below.

- 0 < E < 255 (normal numbers)
- E = 0 and F = 0 (signed zero)
- E = 0 and F > 0 (denormalized numbers)

Note: The number is treated as 0 when the DN bit in the FPSW is 1. When the DN bit is 0, an un-implemented processing exception is generated.

- E = 255 and F = 0 (infinity)
- E = 255 and F > 0 (NaN: Not-a-Number)

### 2.4.3 Bitwise Operations

Five bit-manipulation instructions are provided for bitwise operations: BCLR, BM*Cnd*, BNOT, BSET, and BTST.

A bit in a register is specified as the destination register and a bit number in the range from 31 to 0.

A bit in memory is specified as the destination address and a bit number from 7 to 0. The addressing modes available to specify addresses are register indirect and register relative.

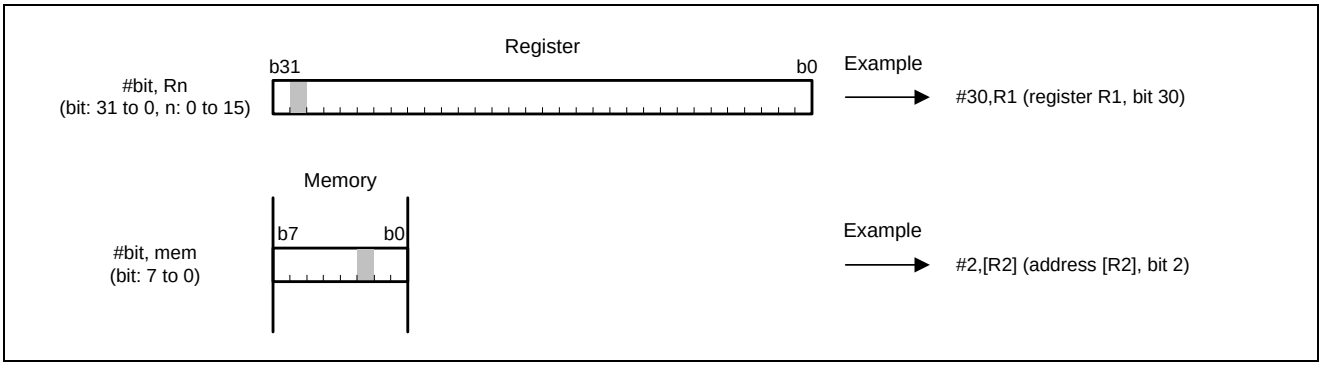



**Figure 2.4 Bit**

### 2.4.4 Strings

The string data type consists of an arbitrary number of consecutive byte (8-bit), word (16-bit), or longword (32-bit) units.

Seven string manipulation instructions are provided for use with strings: SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, and SWHILE.

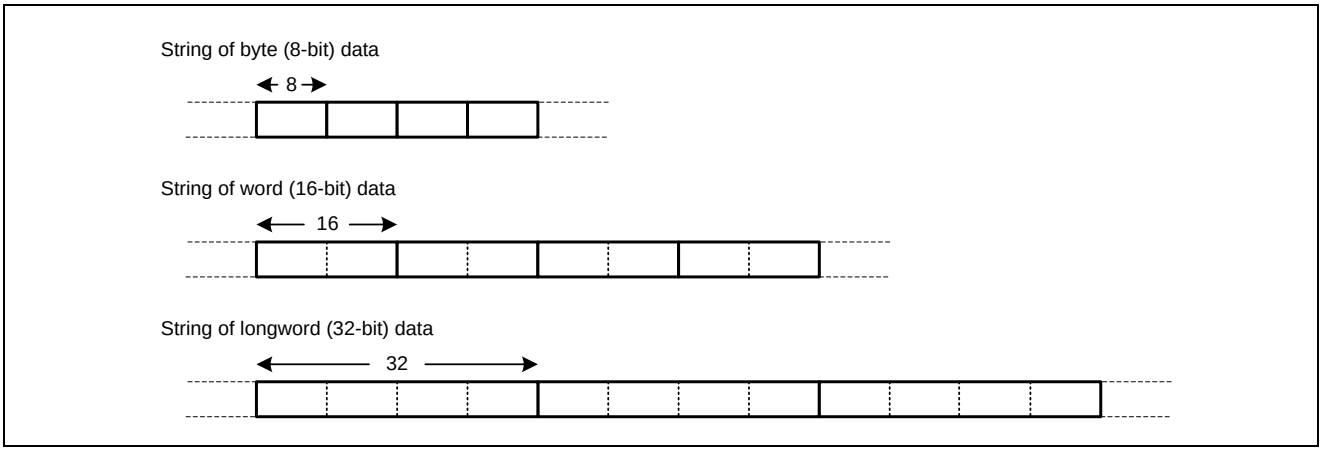



**Figure 2.5 String**



Downloaded from Arrow.com.

## 2.5 Endian

For the RX CPU, instructions are always little endian, but the treatment of data is selectable as little or big endian.

### 2.5.1 Switching the Endian

As arrangements of bytes, the RX610 Group supports both big endian, where the higher-order byte (MSB) is at location 0, and little endian, where the lower-order byte (LSB) is at location 0.

The endian is switched by changing the setting on a mode pin (MDE). For details on the endian setting, see section 3, Operating Modes.

Operations for access differ according to the endian setting and, depending on the instruction, whether 8-, 16- or 32-bit access has been selected. Operations for access in the various possible cases are described in tables 2.1 to 2.12.

In the tables, LL indicates bits D7 to D0 of the general register,
LH indicates bits D15 to D8 of the general register,
HL indicates bits D23 to D16 of the general register, and
HH indicates bits D31 to D24 of the general register.

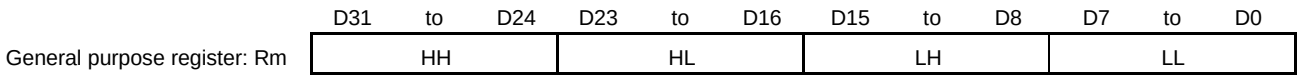



**Table 2.1 32-Bit Read Operations when Little Endian has been Selected**

| Operation / Address of src | Reading a 32-bit unit from address 0 | Reading a 32-bit unit from address 1 | Reading a 32-bit unit from address 2 | Reading a 32-bit unit from address 3 | Reading a 32-bit unit from address 4 |
|---|---|---|---|---|---|
| Address 0 | Transfer to LL | — | — | — | — |
| Address 1 | Transfer to LH | Transfer to LL | — | — | — |
| Address 2 | Transfer to HL | Transfer to LH | Transfer to LL | — | — |
| Address 3 | Transfer to HH | Transfer to HL | Transfer to LH | Transfer to LL | — |
| Address 4 | — | Transfer to HH | Transfer to HL | Transfer to LH | Transfer to LL |
| Address 5 | — | — | Transfer to HH | Transfer to HL | Transfer to LH |
| Address 6 | — | — | — | Transfer to HH | Transfer to HL |
| Address 7 | — | — | — | — | Transfer to HH |



Downloaded from Arrow.com.

**Table 2.2 32-Bit Read Operations when Big Endian has been Selected**

| Operation / Address of src | Reading a 32-bit unit from address 0 | Reading a 32-bit unit from address 1 | Reading a 32-bit unit from address 2 | Reading a 32-bit unit from address 3 | Reading a 32-bit unit from address 4 |
|---|---|---|---|---|---|
| Address 0 | Transfer to HH | — | — | — | — |
| Address 1 | Transfer to HL | Transfer to HH | — | — | — |
| Address 2 | Transfer to LH | Transfer to HL | Transfer to HH | — | — |
| Address 3 | Transfer to LL | Transfer to LH | Transfer to HL | Transfer to HH | — |
| Address 4 | — | Transfer to LL | Transfer to LH | Transfer to HL | Transfer to HH |
| Address 5 | — | — | Transfer to LL | Transfer to LH | Transfer to HL |
| Address 6 | — | — | — | Transfer to LL | Transfer to LH |
| Address 7 | — | — | — | — | Transfer to LL |

**Table 2.3 32-Bit Write Operations when Little Endian has been Selected**

| Operation / Address of dest | Writing a 32-bit unit to address 0 | Writing a 32-bit unit to address 1 | Writing a 32-bit unit to address 2 | Writing a 32-bit unit to address 3 | Writing a 32-bit unit to address 4 |
|---|---|---|---|---|---|
| Address 0 | Transfer from LL | — | — | — | — |
| Address 1 | Transfer from LH | Transfer from LL | — | — | — |
| Address 2 | Transfer from HL | Transfer from LH | Transfer from LL | — | — |
| Address 3 | Transfer from HH | Transfer from HL | Transfer from LH | Transfer from LL | — |
| Address 4 | — | Transfer from HH | Transfer from HL | Transfer from LH | Transfer from LL |
| Address 5 | — | — | Transfer from HH | Transfer from HL | Transfer from LH |
| Address 6 | — | — | — | Transfer from HH | Transfer from HL |
| Address 7 | — | — | — | — | Transfer from HH |

**Table 2.4 32-Bit Write Operations when Big Endian has been Selected**

| Operation / Address of dest | Writing a 32-bit unit to address 0 | Writing a 32-bit unit to address 1 | Writing a 32-bit unit to address 2 | Writing a 32-bit unit to address 3 | Writing a 32-bit unit to address 4 |
|---|---|---|---|---|---|
| Address 0 | Transfer from HH | — | — | — | — |
| Address 1 | Transfer from HL | Transfer from HH | — | — | — |
| Address 2 | Transfer from LH | Transfer from HL | Transfer from HH | — | — |
| Address 3 | Transfer from LL | Transfer from LH | Transfer from HL | Transfer from HH | — |
| Address 4 | — | Transfer from LL | Transfer from LH | Transfer from HL | Transfer from HH |
| Address 5 | — | — | Transfer from LL | Transfer from LH | Transfer from HL |
| Address 6 | — | — | — | Transfer from LL | Transfer from LH |
| Address 7 | — | — | — | — | Transfer from LL |



Downloaded from Arrow.com.

**Table 2.5 16-Bit Read Operations when Little Endian has been Selected**

| Operation / Address of src | Reading a 16-bit unit from address 0 | Reading a 16-bit unit from address 1 | Reading a 16-bit unit from address 2 | Reading a 16-bit unit from address 3 | Reading a 16-bit unit from address 4 | Reading a 16-bit unit from address 5 | Reading a 16-bit unit from address 6 |
|---|---|---|---|---|---|---|---|
| Address 0 | Transfer to LL | — | — | — | — | — | — |
| Address 1 | Transfer to LH | Transfer to LL | — | — | — | — | — |
| Address 2 | — | Transfer to LH | Transfer to LL | — | — | — | — |
| Address 3 | — | — | Transfer to LH | Transfer to LL | — | — | — |
| Address 4 | — | — | — | Transfer to LH | Transfer to LL | — | — |
| Address 5 | — | — | — | — | Transfer to LH | Transfer to LL | — |
| Address 6 | — | — | — | — | — | Transfer to LH | Transfer to LL |
| Address 7 | — | — | — | — | — | — | Transfer to LH |

**Table 2.6 16-Bit Read Operations when Big Endian has been Selected**

| Operation / Address of src | Reading a 16-bit unit from address 0 | Reading a 16-bit unit from address 1 | Reading a 16-bit unit from address 2 | Reading a 16-bit unit from address 3 | Reading a 16-bit unit from address 4 | Reading a 16-bit unit from address 5 | Reading a 16-bit unit from address 6 |
|---|---|---|---|---|---|---|---|
| Address 0 | Transfer to LH | — | — | — | — | — | — |
| Address 1 | Transfer to LL | Transfer to LH | — | — | — | — | — |
| Address 2 | — | Transfer to LL | Transfer to LH | — | — | — | — |
| Address 3 | — | — | Transfer to LL | Transfer to LH | — | — | — |
| Address 4 | — | — | — | Transfer to LL | Transfer to LH | — | — |
| Address 5 | — | — | — | — | Transfer to LL | Transfer to LH | — |
| Address 6 | — | — | — | — | — | Transfer to LL | Transfer to LH |
| Address 7 | — | — | — | — | — | — | Transfer to LL |

**Table 2.7 16-Bit Write Operations when Little Endian has been Selected**

| Operation / Address of dest | Writing a 16-bit unit to address 0 | Writing a 16-bit unit to address 1 | Writing a 16-bit unit to address 2 | Writing a 16-bit unit to address 3 | Writing a 16-bit unit to address 4 | Writing a 16-bit unit to address 5 | Writing a 16-bit unit to address 6 |
|---|---|---|---|---|---|---|---|
| Address 0 | Transfer from LL | — | — | — | — | — | — |
| Address 1 | Transfer from LH | Transfer from LL | — | — | — | — | — |
| Address 2 | — | Transfer from LH | Transfer from LL | — | — | — | — |
| Address 3 | — | — | Transfer from LH | Transfer from LL | — | — | — |
| Address 4 | — | — | — | Transfer from LH | Transfer from LL | — | — |
| Address 5 | — | — | — | — | Transfer from LH | Transfer from LL | — |
| Address 6 | — | — | — | — | — | Transfer from LH | Transfer from LL |
| Address 7 | — | — | — | — | — | — | Transfer from LH |



Downloaded from Arrow.com.

**Table 2.8 16-Bit Write Operations when Big Endian has been Selected**

| Operation / Address of dest | Writing a 16-bit unit to address 0 | Writing a 16-bit unit to address 1 | Writing a 16-bit unit to address 2 | Writing a 16-bit unit to address 3 | Writing a 16-bit unit to address 4 | Writing a 16-bit unit to address 5 | Writing a 16-bit unit to address 6 |
|---|---|---|---|---|---|---|---|
| Address 0 | Transfer from LH | — | — | — | — | — | — |
| Address 1 | Transfer from LL | Transfer from LH | — | — | — | — | — |
| Address 2 | — | Transfer from LL | Transfer from LH | — | — | — | — |
| Address 3 | — | — | Transfer from LL | Transfer from LH | — | — | — |
| Address 4 | — | — | — | Transfer from LL | Transfer from LH | — | — |
| Address 5 | — | — | — | — | Transfer from LL | Transfer from LH | — |
| Address 6 | — | — | — | — | — | Transfer from LL | Transfer from LH |
| Address 7 | — | — | — | — | — | — | Transfer from LL |

**Table 2.9 8-Bit Read Operations when Little Endian has been Selected**

| Operation / Address of src | Reading an 8-bit unit from address 0 | Reading an 8-bit unit from address 1 | Reading an 8-bit unit from address 2 | Reading an 8-bit unit from address 3 |
|---|---|---|---|---|
| Address 0 | Transfer to LL | — | — | — |
| Address 1 | — | Transfer to LL | — | — |
| Address 2 | — | — | Transfer to LL | — |
| Address 3 | — | — | — | Transfer to LL |

**Table 2.10 8-Bit Read Operations when Big Endian has been Selected**

| Operation / Address of src | Reading an 8-bit unit from address 0 | Reading an 8-bit unit from address 1 | Reading an 8-bit unit from address 2 | Reading an 8-bit unit from address 3 |
|---|---|---|---|---|
| Address 0 | Transfer to LL | — | — | — |
| Address 1 | — | Transfer to LL | — | — |
| Address 2 | — | — | Transfer to LL | — |
| Address 3 | — | — | — | Transfer to LL |

**Table 2.11 8-Bit Write Operations when Little Endian has been Selected**

| Operation / Address of dest | Writing an 8-bit unit to address 0 | Writing an 8-bit unit to address 1 | Writing an 8-bit unit to address 2 | Writing an 8-bit unit to address 3 |
|---|---|---|---|---|
| Address 0 | Transfer from LL | — | — | — |
| Address 1 | — | Transfer from LL | — | — |
| Address 2 | — | — | Transfer from LL | — |
| Address 3 | — | — | — | Transfer from LL |



Downloaded from Arrow.com.

**Table 2.12 8-Bit Write Operations when Big Endian has been Selected**

| Operation / Address of dest | Writing an 8-bit unit to address 0 | Writing an 8-bit unit to address 1 | Writing an 8-bit unit to address 2 | Writing an 8-bit unit to address 3 |
|---|---|---|---|---|
| Address 0 | Transfer from LL | — | — | — |
| Address 1 | — | Transfer from LL | — | — |
| Address 2 | — | — | Transfer from LL | — |
| Address 3 | — | — | — | Transfer from LL |

## 2.5.2 Access to I/O Registers

The addresses of I/O registers are fixed, and this is regardless of whether the setting on the MDE pin is for little endian or big endian. Accordingly, changes to the endian do not affect access to I/O registers. For the arrangements of I/O registers, refer to the descriptions of registers in the relevant sections.

## 2.5.3 Notes on Access to I/O Registers

Ensure that access to I/O registers is in accord with the following rules.

- For I/O registers designated as having a bus width of eight bits, use instructions for an eight-bit bus width.
- For I/O registers designated as having a bus width of 16 bits, use instructions for a 16-bit bus width.
- For I/O registers designated as having a bus width of 32 bits, use instructions for a 32-bit bus width.



Downloaded from Arrow.com.

## 2.5.4 Data Arrangement

### 2.5.4.1 Data Arrangement in Registers

Figure 2.6 shows the relation between the sizes of registers and bit numbers.

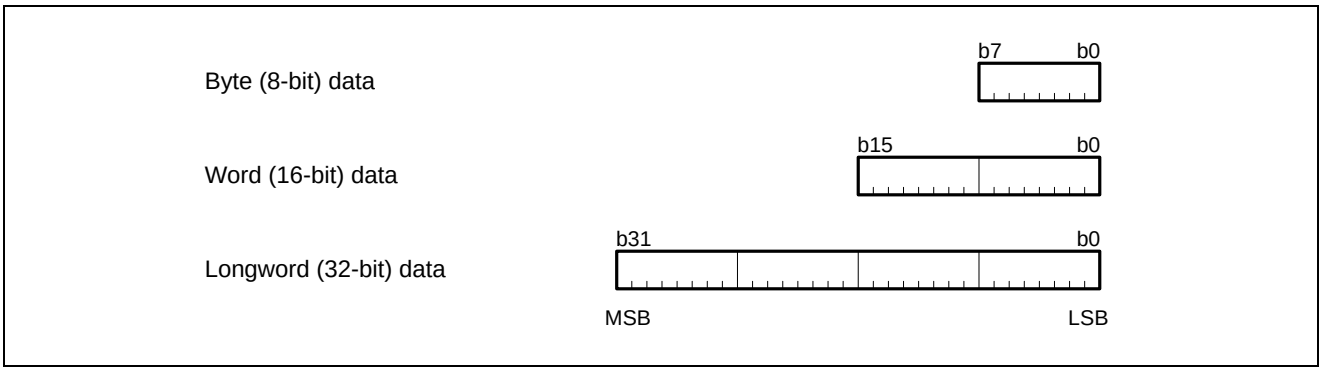



**Figure 2.6 Data Arrangement in Registers**

### 2.5.4.2 Data Arrangement in Memory

Data in memory have three sizes: byte (8-bit), word (16-bit), and longword (32-bit). The data arrangement is selectable as little endian or big endian. Figure 2.7 shows the arrangement of data in memory.

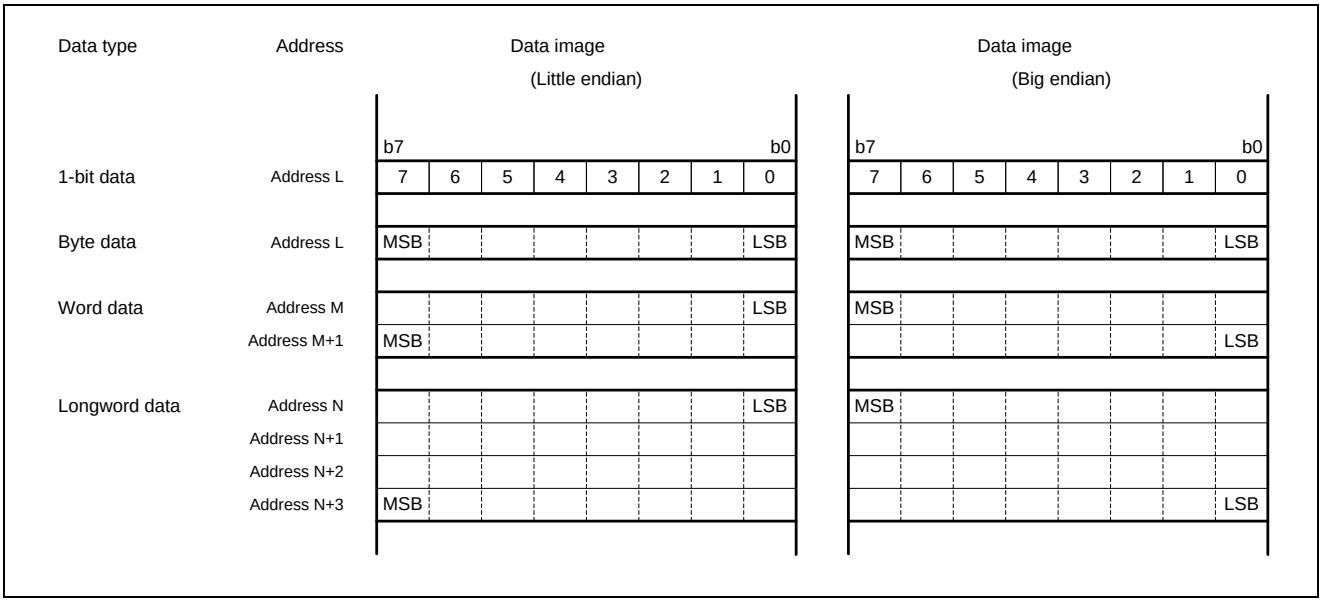



**Figure 2.7 Data Arrangement in Memory**

## 2.5.5 Notes on Arrangement of Instruction Code

When the endian setting for the external address space is different from that for the chip, no instruction code can be arranged in the area. The instruction code should be arranged in the external address space whose endian setting is the same as that for the chip



Downloaded from Arrow.com.

## 2.6 Vector Table

There are two types of vector table: fixed and relocatable. Each vector in the vector table consists of four bytes and specifies the address where the corresponding exception processing routine starts.

### 2.6.1 Fixed Vector Table

The fixed vector table is allocated to a fixed address range. The individual vectors for the privileged instruction exception, undefined instruction exception, floating-point exception, non-maskable interrupt, and reset are allocated to addresses in the range from FFFFFF80h to FFFFFFFFh. Figure 2.8 shows the fixed vector table.

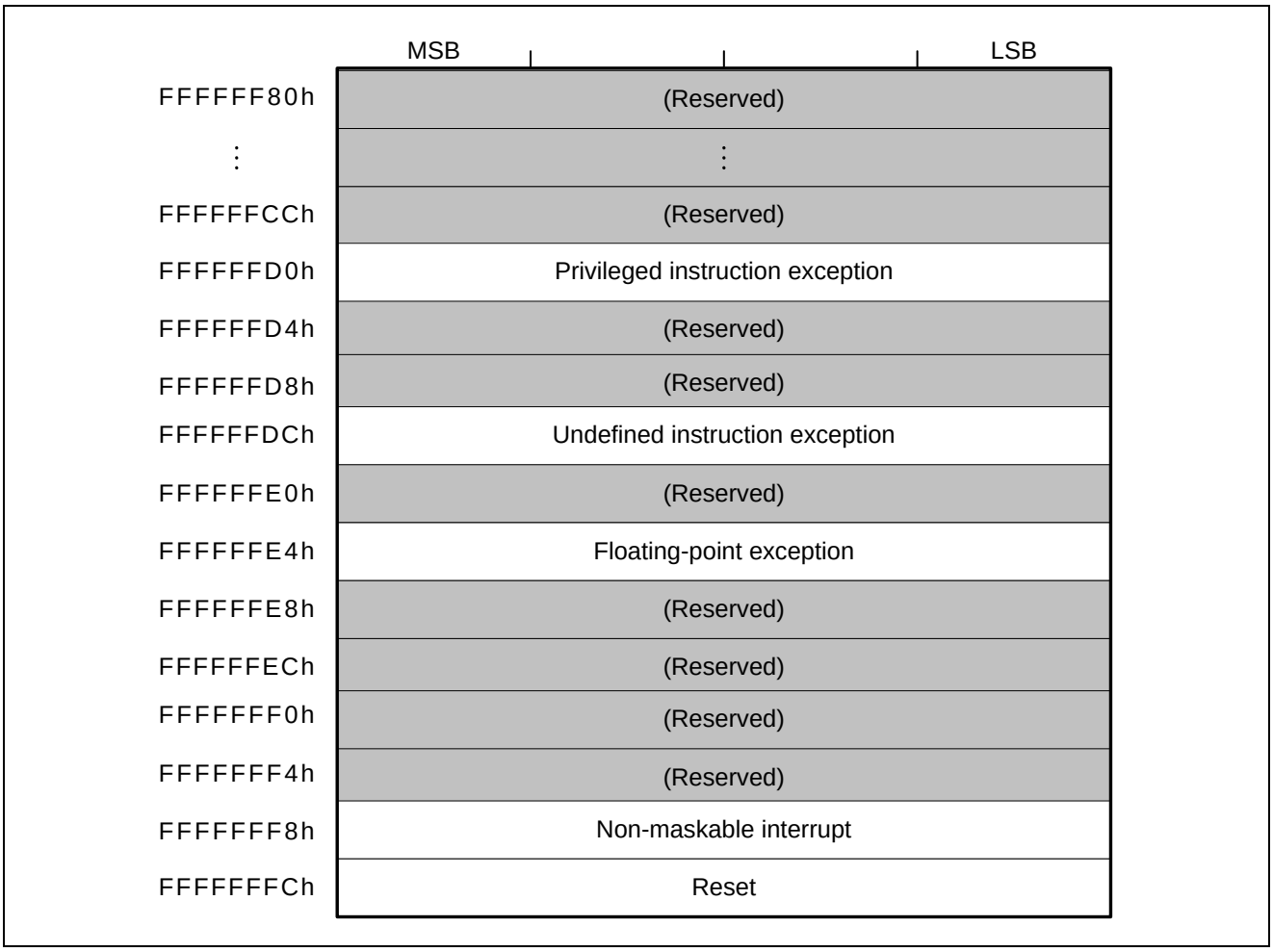



**Figure 2.8 Fixed Vector Table**



Downloaded from Arrow.com.

### 2.6.2 Relocatable Vector Table

The address where the relocatable vector table is placed can be adjusted. The table is a 1,024-byte region that contains all vectors for unconditional traps and interrupts and starts at the address (IntBase) specified in the interrupt table register (INTB). Figure 2.9 shows the relocatable vector table.

Each vector in the relocatable vector table has a vector number from 0 to 255. Each of the INT instructions, which act as the sources of unconditional traps, is allocated to the vector that has the same number as that of the instruction itself (from 0 to 255). The BRK instruction is allocated to the vector with number 0. Furthermore, vector numbers within the set from 0 to 255 may also be allocated to other interrupt sources on a per-product basis. For more on the interrupt vector numbers, see section 10.3.1, Interrupt Vector Table.

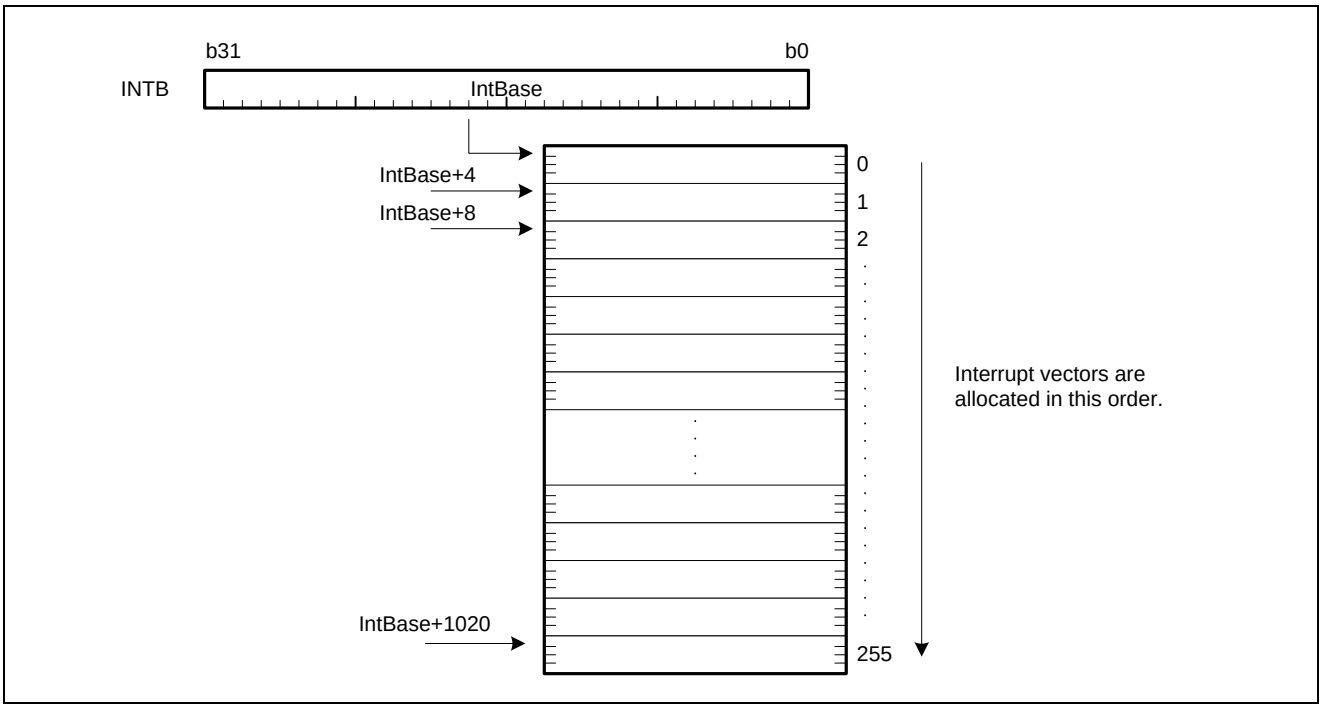



**Figure 2.9 Relocatable Vector Table**



Downloaded from Arrow.com.

## 2.7 Operation of Instructions

### 2.7.1 Data Prefetching by the RMPA Instruction and the String-Manipulation Instructions

The RMPA instruction and the string-manipulation instructions (SCMPU, SMOVB, SMOVF, SMOVU, SUNTIL, SWHILE, and SSTR instructions are not included) may prefetch data from the memory to speed up the read processing. Data is prefetched from the prefetching start position with three bytes as the upper limit. The prefetching start positions of each operation are shown below.

- RMPA instruction: The multiplicand address specified by R1, and the multiplier address specified by R2
- SCMPU instruction: The source address specified by R1 for comparison, and the destination address specified by R2 for comparison
- SUNTIL and SWHILE instructions: The destination address specified by R1 for comparison
- SMOVB, SMOVF, and SMOVU instructions: The source address specified by R2 for transfer



Downloaded from Arrow.com.

## 2.8 Pipeline

### 2.8.1 Overview

The RX CPU has 5-stage pipeline structure. The RX CPU instruction is converted into one or more micro-operations, which are then executed in pipeline processing. In the pipeline stage, the IF stage is executed in the unit of instructions, while the D and subsequent stages are executed in the unit of micro-operations.

The operation of pipeline and respective stages is described below.

#### (1) IF stage (instruction fetch stage)

In the IF stage, the CPU fetches instructions from the memory. As the RX CPU has four 8-byte instruction queues, it fetches instructions until the instruction queue is full, regardless of the completion of decoding in the D (decoding) stage.

#### (2) D stage (decoding stage)

The CPU decodes instructions in the D stage and converts them into micro-operations. The CPU reads the register information (RF) in this stage and executes a bypass process (BYP) if the result of the preceding instruction will be used in a subsequent instruction. The write of operation result to the register (RW) can be executed with the register reference by using the bypass process.

#### (3) E stage (execution stage)

Operations and address calculations (OP) are processed in the E stage.

#### (4) M stage (memory access stage)

Operand memory accesses (OA) are processed in the M stage. This stage is used only when the memory is accessed, and is divided into two sub-stages, M1 and M2. The RX CPU enables respective memory accesses for M1 and M2.

- M1 stage (memory-access stage 1)
  Operand memory access (OA1, OA2) is processed.

  Store operation: The pipeline processing ends when a write request is received via the bus.

  Load operation: The operation proceeds to the M2 stage when a read request is received via the bus. If a request and load data are received at the same timing (no-wait memory access), the operation proceeds to the WB stage.

- M2 stage (memory-access stage 2)
  Operand memory access (OA2) is processed. The CPU waits for the load data in the M2 stage. When the load data is received, the operation proceeds to the WB stage.

#### (5) WB stage (write-back stage)

The operation result and the data read from memory are written to the register (RW) in the WB stage. The data read from memory and the other type of data, such as the operation result, can be written to the register in the same clock cycles.



Downloaded from Arrow.com.

Figure 2.10 shows the pipeline configuration and its operation.

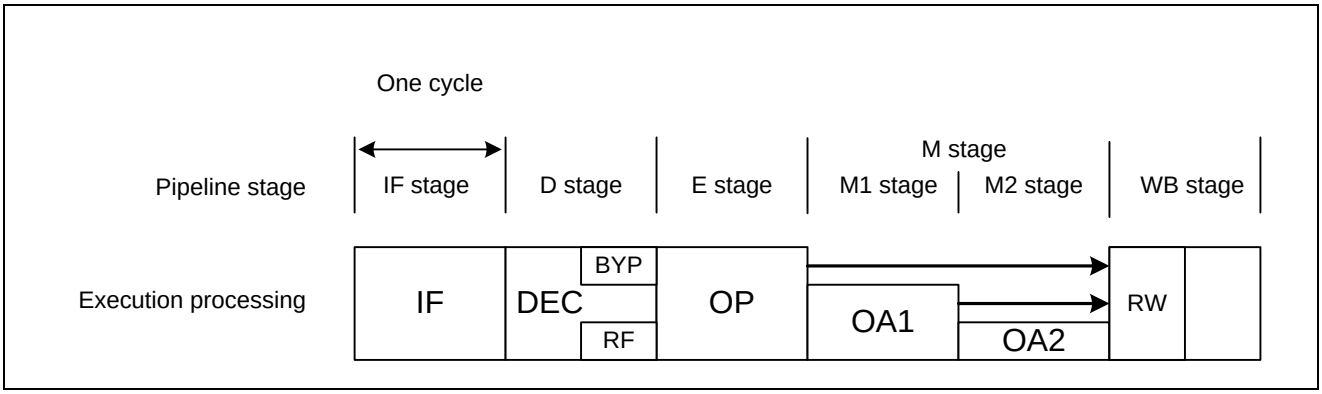



**Figure 2.10 Pipeline Configuration and its Operation**



Downloaded from Arrow.com.

### 2.8.2 Instructions and Pipeline Processing

The operands in the table below indicate the following meaning.

#IMM: Immediate
Rs, Rs2, Rd, Rd2, Ri, Rb: General-purpose register, CR: Control register
dsp: dsp5, dsp8, dsp16, dsp24
pcdsp: pcdsp3, pcdsp8, pcdsp16, pcdsp24

#### 2.8.2.1 Instructions Converted into Single Micro-Operation and Pipeline Processing

The table below lists the instructions that are converted into a single micro-operation. The number of cycles in the table indicates the number of cycles during no-wait memory access.

**Table 2.13 Instructions that are Converted into a Single Micro-Operation**

| Instruction | Mnemonic (indicates the common operation when the size is omitted) | Reference Figure | Number of Cycles |
|---|---|---|---|
| Arithmetic/logic instructions (register-register, immediate-register) Except EMUL, EMULU, RMPA, DIV, and DIVU | • {ABS, ADC (omitted), XOR} “#IMM, Rd”/“Rd” /“Rs, Rd”/“Rs, Rs2, Rd” | Figure 2.11 | 1 |
| Arithmetic/logic instructions (division) | • DIV “#IMM, Rd”/“Rs, Rd” | Figure 2.11 | 3 to 20*1 |
| | • DIVU “#IMM, Rd”/“Rs, Rd” | Figure 2.11 | 2 to 18*1 |
| Data transfer instructions (register-register, immediate-register) | • {MOV, MOVU, REVL, REVW} “#IMM, Rd”/“Rs, Rd”<br>• SCCnd “Rd”<br>• {STNZ, STZ} “#IMM, Rd” | Figure 2.11 | 1 |
| Transfer instructions (load operation) | • {MOV, MOVU} “[Rs], Rd”/“dsp[Rs], Rd” /“[Rs+], Rd”/“[-Rs], Rd”/“Rs, [Ri, Rb]”<br>• POP “Rd” | Figure 2.12 | Throughput: 1<br>Latency: 2*2 |
| Transfer instructions (store operation) | • MOV “Rs, [Rd]”/“Rs, dsp[Rd]”/“Rs, [Rd+]” /“Rs, [-Rd]”/“Rs, [Ri, Rb]”<br>• PUSH “Rs”<br>• PUSHC “CR” | Figure 2.13 | 1 |
| Bit manipulation instructions (register) | • {BCLR, BNOT, BSET, BTST} “#IMM, Rd”/“Rs, Rd”<br>• BMCnd “#IMM, Rd” | Figure 2.11 | 1 |
| Branch instructions | • BCnd “pcdsp”<br>• {BRA, BSR} “pcdsp”/“Rs”<br>• {JMP, JSR} “Rs” | Figure 2.22 | Branch taken: 3<br>Branch not taken: 1 |
| Floating-point operation instructions (register-register, immediate-register) | • FCMP “#IMM, Rd”/“Rs, Rd” | Figure 2.11 | 1 |
| System manipulation instructions | • CLRPSW, SETPSW “#IMM”<br>• MVTC “#IMM, CR”/“Rs, CR”<br>• MVFC “CR, Rd” | — | 1 |

Notes: 1. The number of cycles for the dividing instruction varies according to the divisor and dividend.
2. For the number of cycles for throughput and latency, see section 2.8.3, Calculation of the Instruction Processing Time.



Downloaded from Arrow.com.

Figures 2.11 to 2.13 show the operation of instructions that are converted into a basic single micro-operation.

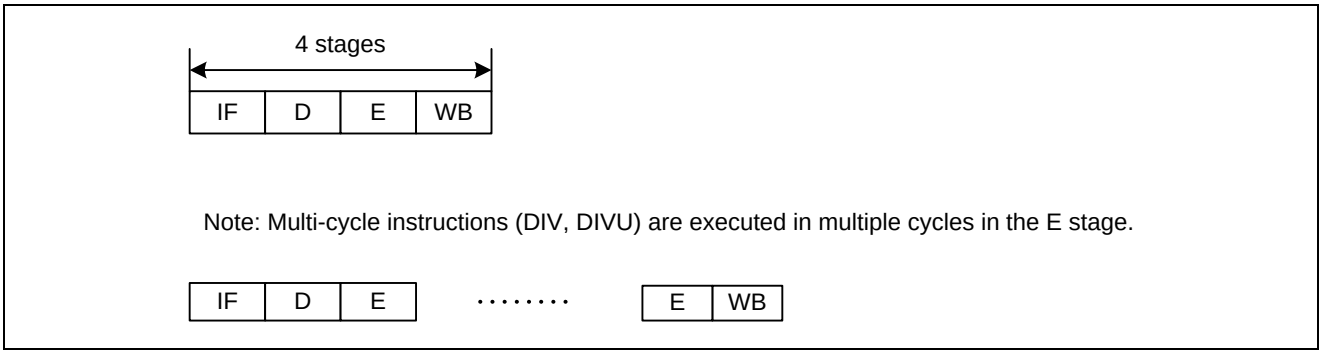



**Figure 2.11 Operation for Register-Register, Immediate-Register**

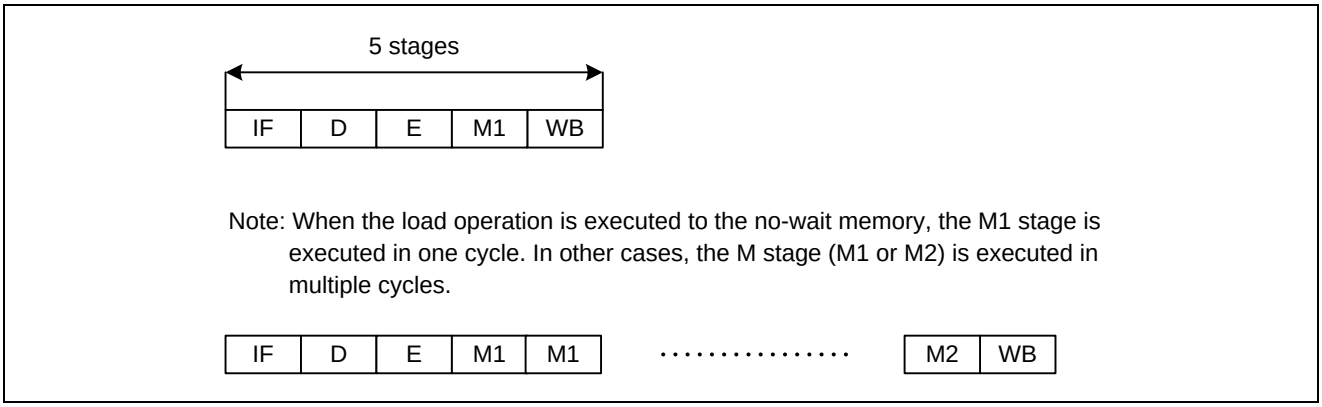



**Figure 2.12 Load Operation**

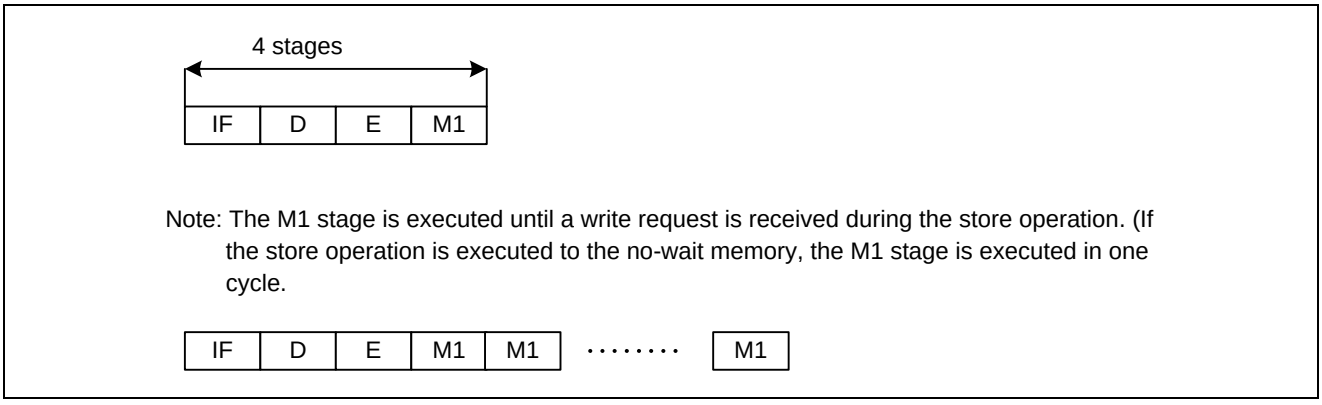



**Figure 2.13 Store Operation**



Downloaded from Arrow.com.

### 2.8.2.2 Instructions Converted into Multiple Micro-Operations and Pipeline Processing

The table below lists the instructions that are converted into multiple micro-operations. The number of cycles in the table indicates the number of cycles during no-wait memory access.

**Table 2.14 Instructions that are Converted into Multiple Micro-Operations**

| Instruction | Mnemonic (indicates the common operation when the size is omitted) | Reference Figure | Number of Cycles |
|---|---|---|---|
| Arithmetic/logic instructions (memory source operand) | • ADC, ADD (omitted), XOR} "[Rs], Rd" /"dsp[Rs], Rd" | Figure 2.14 | 3 |
| Arithmetic/logic instructions (division) | • DIV "[Rs], Rd / dsp[Rs], Rd" | — | 5 to 22 |
| | • DIVU "[Rs], Rd / dsp[Rs], Rd" | — | 4 to 20 |
| Arithmetic/logic instructions (multiplier: 32 x 32 → 64 bits) (register-register, register-immediate) | • {EMUL, EMULU} "#IMM, Rd"/"Rs, Rd" | Figure 2.16 | 2 |
| Arithmetic/logic instructions (multiply-and-accumulate operation) | • RMPA.B | — | 6+7×floor(n/4)+4×(n%4)<br>n: Number of processing bytes*1 |
| | • RMPA.W | — | 6+5×floor(n/2)+4×(n%2)<br>n: Number of processing words*1 |
| | • RMPA.L | — | 6+4n<br>n: Number of processing longwords*1 |
| Data transfer instructions (memory-memory transfer) | • MOV "[Rs], [Rd]"/"dsp[Rs], [Rd]" /"[Rs], dsp[Rd]"/"dsp[Rs], [Rd]"<br>• PUSH "[Rs]"/"dsp[Rs]" | Figure 2.15 | 3 |
| Bit manipulation instructions (memory source operand) | • {BCLR, BNOT, BSET, BTST} "#IMM, [Rd]" /"#IMM, dsp[Rd]"<br>• BMCnd "#IMM, [Rd]"/"#IMM, dsp[Rd]" | Figure 2.15 | 3 |
| Transfer instructions (load operation) | • POPC "CR" | — | Throughput: 3<br>Latency: 4*2 |
| Transfer instructions (store operation of multiple registers) | • PUSHM "Rs-Rs2" | — | n<br>n: Number of registers*3 |
| Transfer instructions (store operation of multiple registers) | • POPM "Rs-Rs2" | — | Throughput: n<br>Latency: n + 1<br>n: Number of registers*2*4 |
| Transfer instructions (register-register) | • XCHG "Rs, Rd" | Figure 2.17 | 2 |
| Transfer instructions (memory-register) | • XCHG "[Rs], Rd"/"dsp[Rs], Rd" | Figure 2.18 | 2 |
| Branch instructions | • RTS | — | 5 |
| | • RTSD "#IMM" | — | 5 |
| | • RTSD "#IMM, Rd-Rd2" | — | Throughput: n<5?5:1+n<br>Latency: n<4?5:2+n<br>n: Number of registers*2 |



Downloaded from Arrow.com.

| Instruction | Mnemonic (indicates the common operation when the size is omitted) | Reference Figure | Number of Cycles |
|---|---|---|---|
| String manipulation instructions*5 | • SCMPU | — | 2+4×floor(n/4)+4×(n%4)<br>n: Number of comparison bytes*1 |
| | • SMOVB | — | n>3?<br>6+3×floor(n/4)+3×(n%4):<br>2+3n<br>n: Number of transfer bytes*1 |
| | • SMOVF, SMOVU | — | 2+3×floor(n/4)+3×(n%4)<br>n: Number of transfer bytes*1 |
| | • SSTR.B | — | 2+floor(n/4)+n%4<br>n: Number of transfer bytes*1 |
| | • SSTR.W | — | 2+floor(n/2)+n%2<br>n: Number of transfer words*1 |
| | • SSTR.L | — | 2+n<br>n: Number of transfer longwords |
| | • SUNTIL.B, SWHILE.B | — | 3+3×floor(n/4)+3×(n%4)<br>n: Number of comparison bytes*1 |
| | • SUNTIL.W, SWHILE.W | — | 3+3×floor(n/2)+3×(n%2)<br>n: Number of comparison words*1 |
| | • SUNTIL.L, SWHILE.L | — | 3+3×n<br>n: Number of comparison longwords |
| Floating-point operation instructions (register-register, immediate-register) | • {FADD, FSUB} “#IMM, Rd”/ “Rs, Rd” | Figure 2.19 | 4 |
| | • FMUL “#IMM, Rd”/“Rs, Rd” | — | 3 |
| | • FDIV “#IMM, Rd”/“Rs, Rd” | — | 16 |
| | • {FTOI, ROUND, ITOF} “Rs, Rd” | — | 2 |
| Floating-point operation instructions (memory source operand) | • {FADD, FSUB} “[Rs], Rd”/“dsp[Rs], Rd” | — | 6 |
| | • FMUL “[Rs], Rd”/“dsp[Rs], Rd” | — | 5 |
| | • FDIV “[Rs], Rd”/“dsp[Rs], Rd” | — | 18 |
| | • {FTOI, ROUND, ITOF} “[Rs], Rd” /“dsp[Rs], Rd” | — | 4 |
| System manipulation instructions | • RTE | — | 6 |
| | • RTFI | — | 3 |

Notes: 1. floor(x): Max. integer that is smaller than x

2. For the number of cycles for throughput and latency, see section 2.8.3, Calculation of the Instruction Processing Time.
3. The PUSHM instruction is converted into multiple store operations. The pipeline processing is the same as the one for the store operations of the MOV instruction, where the operation is repeated for the number of specified registers.
4. The POPM instruction is converted into multiple store operations. The pipeline processing is the same as the one for the store operations of the MOV instruction, where the operation is repeated for the number of specified registers.
5. Each of the SCMPU, SMOVU, SWHILE, and SUNTIL instructions ends the execution regardless of the specified cycles, if the end condition is satisfied during execution.



Downloaded from Arrow.com.

Figures 2.14 to 2.19 show the operation of instructions that are converted into basic multiple micro-operations. Small letters in figures below indicate micro-operations.

[Legend]
mop: Micro-operation, stall: Pipeline stall

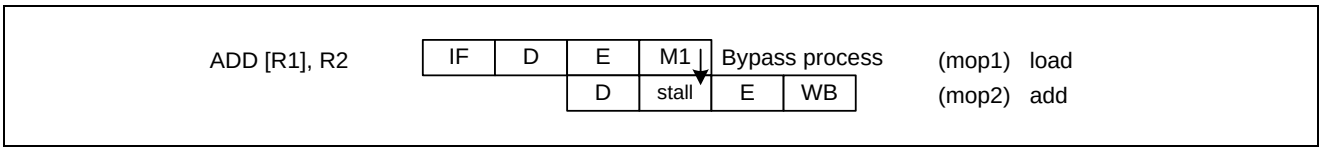



**Figure 2.14 Arithmetic/Logic Instruction (Memory Source Operand)**

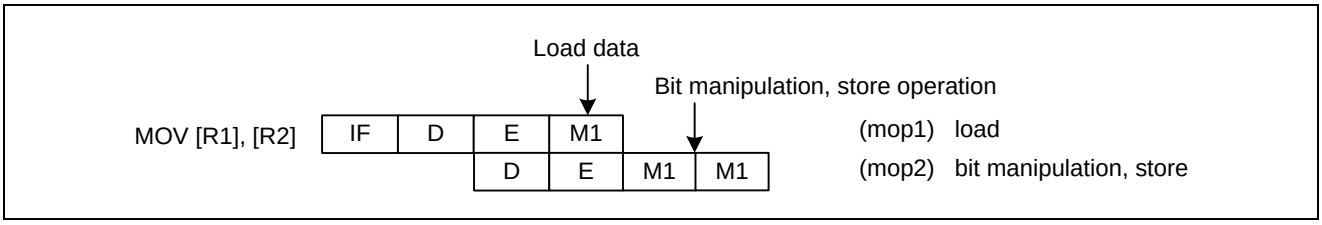



**Figure 2.15 MOV Instruction (Memory-Memory), Bit Manipulation Instruction (Memory Source Operand)**

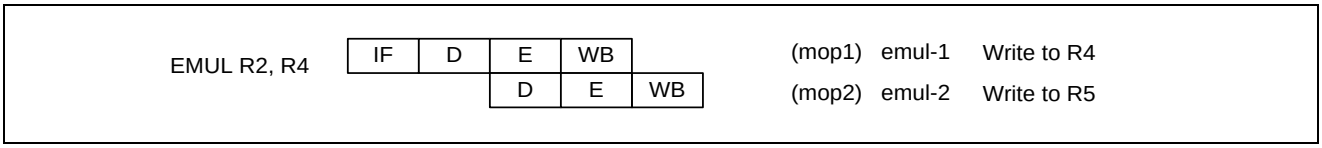



**Figure 2.16 EMUL, EMULU Instructions (Register- Register, Register-Immediate)**

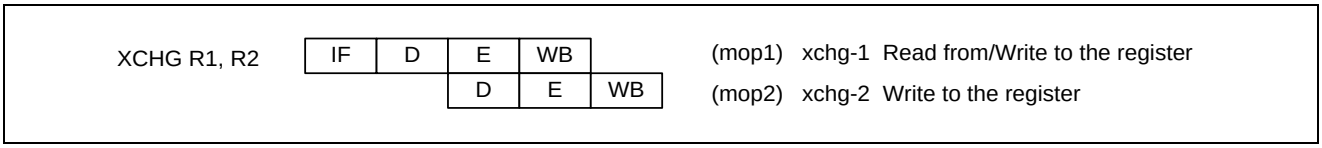



**Figure 2.17 XCHG Instruction (Registers)**

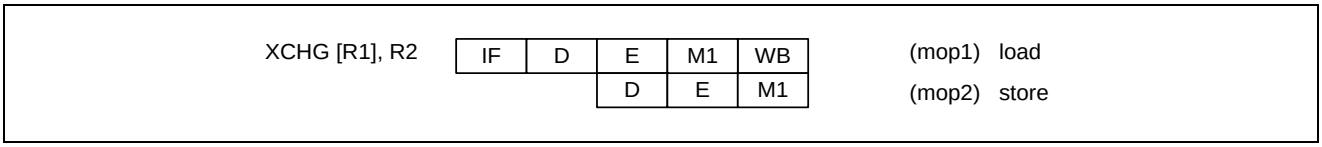



**Figure 2.18 XCHG Instruction (Memory Source Operand)**

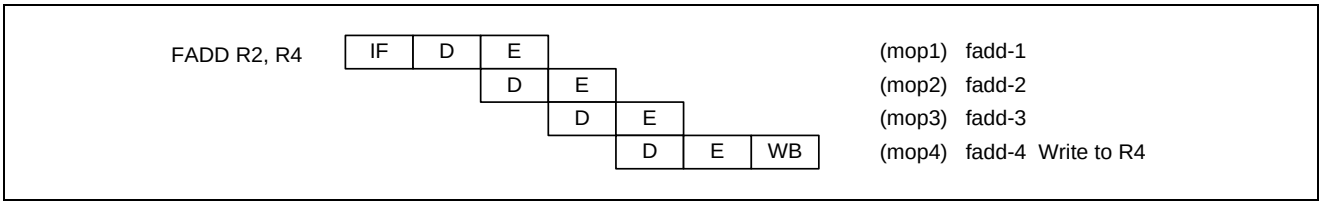



**Figure 2.19 Floating-Point Operation Instruction (Register-Register, Immediate-Register)**



Downloaded from Arrow.com.

### 2.8.2.3 Pipeline Basic Operation

In the ideal pipeline processing, each stage is executed in one cycle, though all instructions may not be pipelined in due to the processing and the branch execution.

The CPU controls the pipeline stage with the IF stage in the unit of instructions, while the D and subsequent stages in the unit of micro-operations.

The figures below show the pipeline processing of typical cases. Small letters in figures indicate micro-operations.

[Legend]
mop: Micro-operation, stall: Pipeline stall

#### (1) Pipeline Flow with Stalls

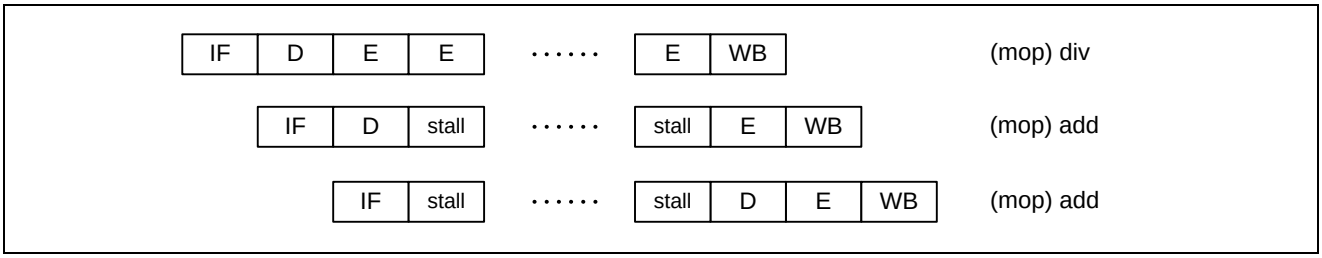



**Figure 2.20 When an Instruction which Requires Multiple Cycles is Executed in the E Stage**

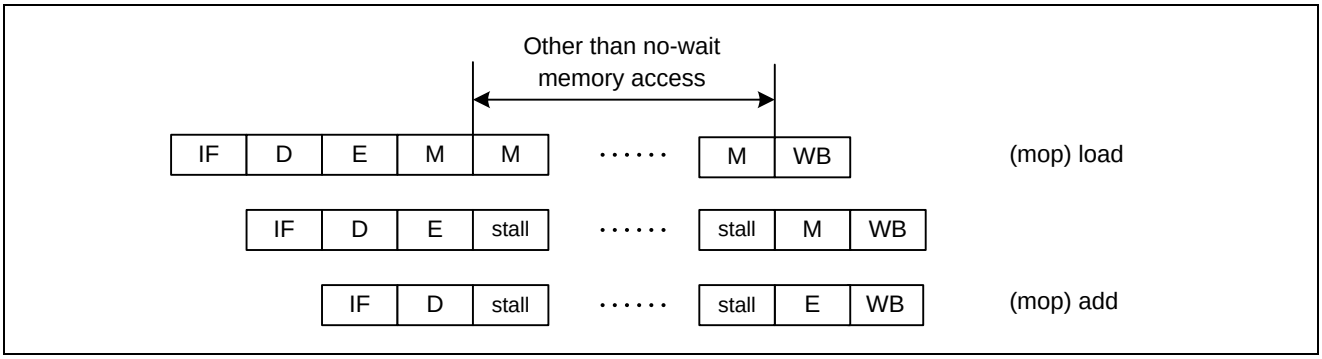



**Figure 2.21 When an Instruction which Requires more than One Cycle for its Operand Access is Executed**

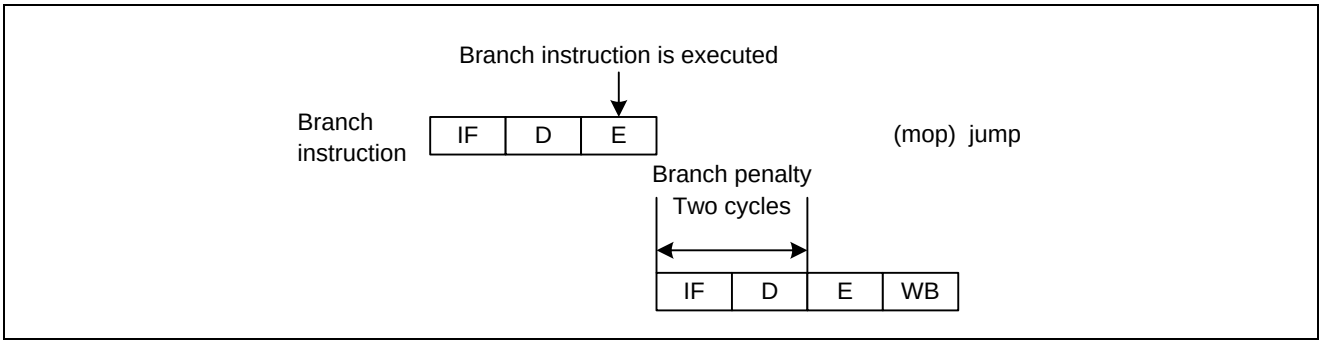



**Figure 2.22 When a Branch Instruction is Executed (While the Condition is Satisfied for Unconditional/Conditional Branch Instruction)**



Downloaded from Arrow.com.

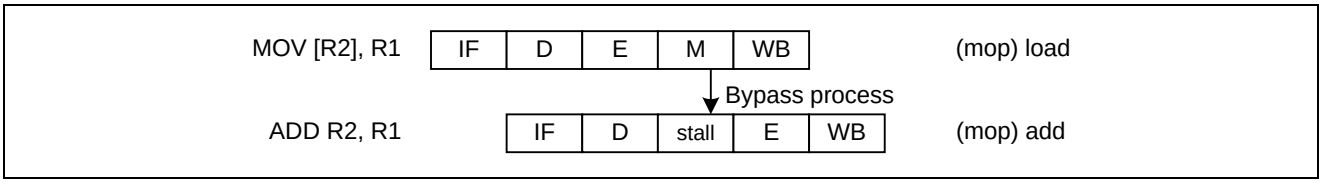



**Figure 2.23 When the Subsequent Instruction Uses an Operand Read from the Memory**

### (2) Pipeline Flow with no Stall

#### (a) Bypass process

Even when the result of the preceding instruction will be used in a subsequent instruction, the operation processing between registers is pipelined in by the bypass process.

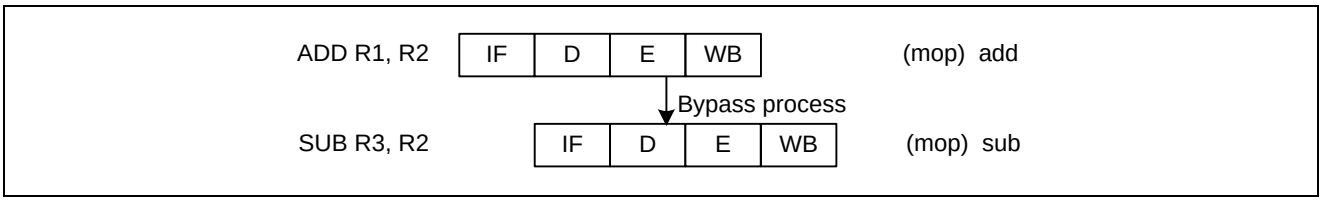



**Figure 2.24 Bypass Process**

#### (b) When WB stages for the memory load and for the operation are overlapped

Even when the WB stages for the memory load and for the operation are overlapped, the operation processing is pipelined in, because the load data and the operation result can be written to the register at the same timing.

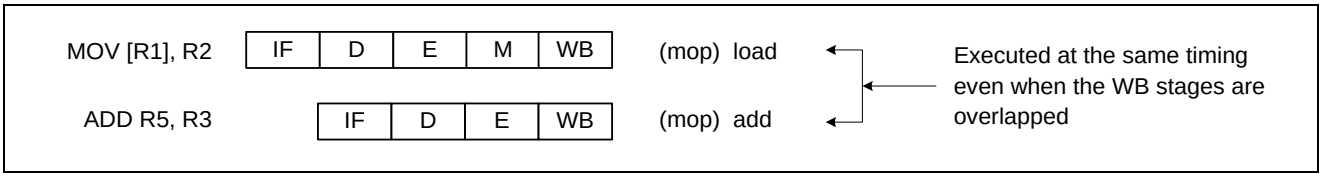



**Figure 2.25 When WB Stages for the Memory Load and for the Operation are Overlapped**



Downloaded from Arrow.com.

### (c) When subsequent instruction writes to the same register before the end of memory load

Even when the subsequent instruction writes to the same register before the end of memory load, the operation processing is pipelined in, because the WB stage for the memory load is canceled.

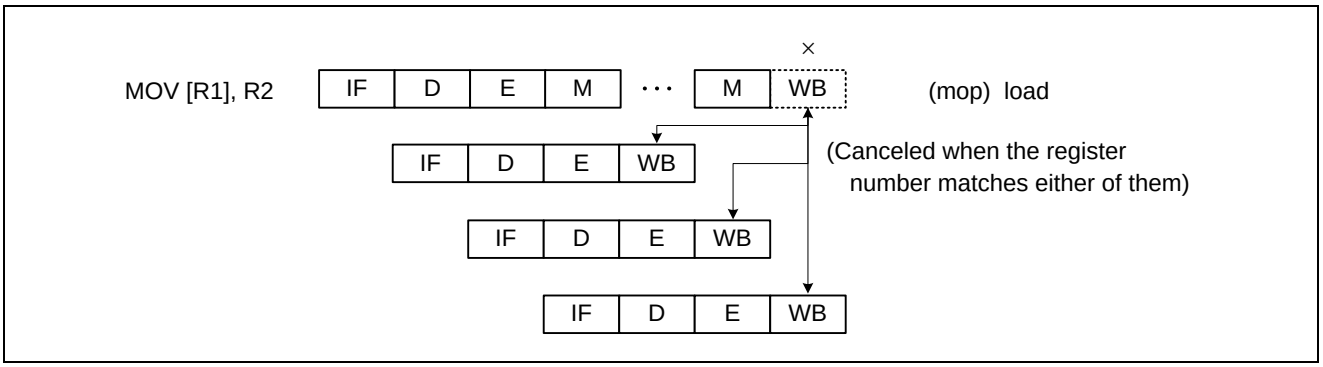



**Figure 2.26 When Subsequent Instruction Writes to the Same Register before the End of Memory Load**

### (d) When the load data is not used by the subsequent instruction

When the load data is not used by the subsequent instruction, the subsequent operations are in fact executed earlier and the operation processing ends (out-of-order completion).

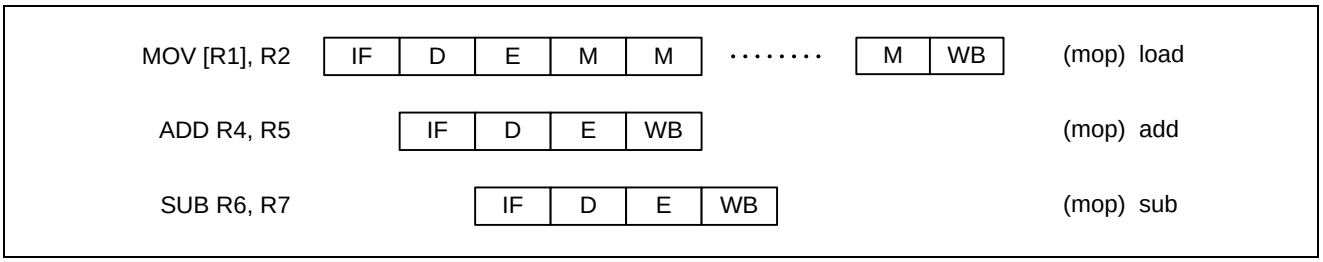



**Figure 2.27 When Load Data is not Used by the Subsequent Instruction**

## 2.8.3 Calculation of the Instruction Processing Time

Though the instruction processing time of the CPU varies according to the pipeline processing, the approximate time can be calculated in the following methods.

- Count the number of cycles (see tables 2.13 and 2.14)
  When the load data is used by the subsequent instruction, the number of cycles described as " latency" is counted as the number of cycles for the memory load instruction. For the cycles other than the memory load instruction, the number of cycles described as "throughput" is counted.
- If the instruction fetch stall is generated, the number of cycles increments.
- Depending on the system configuration, multiple cycles are required for the memory access. The number of memory access cycles in the RX610 Group is on a per-product basis.



Downloaded from Arrow.com.

### 2.8.4 Numbers of Cycles for Response to Interrupts

Table 2.15 lists numbers of cycles taken by processing for response to interrupts.

**Table 2.15 Numbers of Cycles for Response to Interrupts**

| Type of Interrupt Request/Details of Processing | Fast Interrupt | Other Interrupts |
|---|---|---|
| ICU<br>Judgment of priority order | 2 cycles | 2 cycles |
| CPU<br>Number of cycles from notification to acceptance of the interrupt request | N cycles (varies with the instruction being executed at the time the interrupt was received) | |
| CPU—Pre-processing by hardware<br>Saving the current PC and PSW values in RAM (or in control registers in the case of the fast interrupt)<br>Reading of the vector<br>Branching to the start of the interrupt Exception handling routine | 4 cycles | 6 cycles |

Times calculated from the values in table 2.15 will be applicable when access to memory from the CPU is always processed with no waiting. The on-chip RAM and ROM in products of the RX62N/RX621 Groups always allows such access. Numbers of cycles for response to interrupts can be minimized by placing program code (and vectors) in on-chip ROM and the stack in on-chip RAM. Furthermore, place the addresses where handlers start on eight-byte boundaries.

For information on the number of cycles from notification to acceptance of the interrupt request, indicated by N in the table above, see tables 2.13, Instructions that are Converted into a Single Micro-Operation, and 2.14, Instructions that are Converted into Multiple Micro-Operations.

The timing of interrupt acceptance depends on the state of the CPU's pipelines. For more information on this, see section 9.3.1, Timing of Acceptance and Saved PC Values.



Downloaded from Arrow.com.

# 3. Operating Modes

## 3.1 Operating Mode Types and Selection

Operating modes are specified by the MD1 and MD0 pins and the ROME and EXBE bits in the system control register 0 (SYSCR0).

The endian can be selected in each operating mode. Endian is specified by the MDE pin. For details on the endian, see section 11, Buses.

Note:Do not change the state of pins MDE, MD1, and MD0 while the LSI is working. Do not select any combination other than those specified in table 3.1.

**Table 3.1 Selection of Operating Modes by the Mode Pins**

| Mode Pin | | SYSCR0 Initial State | | | | |
|---|---|---|---|---|---|---|
| **MD1** | **MD0** | **ROME** | **EXBE** | **Operating Mode** | **On-Chip ROM*** | **External Bus** |
| 0 | 1 | 1 | 0 | Boot mode | Enabled | Disabled |
| 1 | 0 | 1 | 0 | User boot mode | Enabled | Disabled |
| 1 | 1 | 1 | 0 | Single-chip mode | Enabled | Disabled |

Note: * The on-chip ROM is classified into two flash memories: ROM and data flash. For details, see section 26, ROM (Flash Memory for Code Storage), and section 27, Data Flash (Flash Memory for Data Storage).

**Table 3.2 Selection of Operating Modes by Register Setting**

| SYSCR0 | | | | |
|---|---|---|---|---|
| **ROME** | **EXBE** | **Operating Mode** | **On-Chip ROM*** | **External Bus** |
| 0 | 0 | Single-chip mode, user boot mode | Disabled | Disabled |
| 1 | 0 | | Enabled | Disabled |
| 0 | 1 | On-chip ROM disabled extended mode | Disabled | Enabled |
| 1 | 1 | On-chip ROM enabled extended mode | Enabled | Enabled |

Note: * The on-chip ROM is classified into two flash memories: ROM and data flash. For details, see section 26, ROM (Flash Memory for Code Storage), and section 27, Data Flash (Flash Memory for Data Storage).

**Table 3.3 Selection of Endian**

| Mode Pin | |
|---|---|
| **MDE** | **Endian** |
| 0 | Little endian |
| 1 | Big endian |



Downloaded from Arrow.com.

## 3.2 Register Descriptions

Table 3.4 lists the registers related to operating modes.

**Table 3.4 Registers Related to Operating Modes**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Mode monitor register | MDMONR | 10000000 x00000xxb | 0008 0000h | 16 |
| Mode status register | MDSR | 00000000 00001001b | 0008 0002h | 16 |
| System control register 0 | SYSCR0 | 00000000 00000001b | 0008 0006h | 16 |
| System control register 1 | SYSCR1 | 00000000 00000001b | 0008 0008h | 16 |

### 3.2.1 Mode Monitor Register (MDMONR)

Address: 0008 0000h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | MDE | — | — | — | — | — | MD1 | MD0 |
| Value after reset: | x* | 0 | 0 | 0 | 0 | 0 | x* | x* |

Note: * Depends on the setting of the mode pins (MDE, MD1, and MD0).

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | MD0 | MD0 Status Flag | 0: The MD0 pin is 0<br>1: The MD0 pin is 1 | R |
| b1 | MD1 | MD1 Status Flag | 0: The MD1 pin is 0<br>1: The MD1 pin is 1 | R |
| b6 to b2 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b7 | MDE | MDE Status Flag | 0: The MDE pin is 0 (little endian)<br>1: The MDE pin is 1 (big endian) | R |
| b14 to b8 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b15 | — | Reserved | This bit is always read as 1 and cannot be modified. | R |

MDMONR indicates the status of the mode pins.



Downloaded from Arrow.com.

### 3.2.2 Mode Status Register (MDSR)

Address: 0008 0002h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | UBTS | — | BOTS | BSW[1:0] | | EXB | IROM |
| Value after reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | IROM | On-Chip ROM Startup Status Flag | 0: The on-chip ROM is disabled at startup<br>1: The on-chip ROM is enabled at startup | R |
| b1 | EXB | External Bus Startup Status Flag | 0: The external bus is disabled at startup<br>1: The external bus is enabled at startup | R |
| b3, b2 | BSW[1:0] | External Bus Width Flags | 0 0: The 16-bit bus is activated<br>0 1: Reserved<br>1 0: The 8-bit bus is activated<br>1 1: Reserved | R |
| b4 | BOTS | Boot Mode Startup Flag | 0: Started with a mode except boot mode<br>1: Started with boot mode | R |
| b5 | — | Reserved | This bit is always read as 0 and cannot be modified. | R |
| b6 | UBTS | User Boot Mode Startup Flag | 0: Started with a mode except user boot mode<br>1: Started with user boot mode | R |
| b15 to b7 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |

MDSR indicates the internal status of this LSI at startup.



Downloaded from Arrow.com.

### 3.2.3 System Control Register 0 (SYSCR0)

Address: 0008 0006h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | KEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | EXBE | ROME |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ROME | On-Chip FLASH Enable | 0: The on-chip FLASH is disabled<br>1: The on-chip ROM is enabled | R/W |
| b1 | EXBE | External Bus Enable | 0: The external bus is disabled<br>1: The external bus is enabled | R/W |
| b7 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | KEY[7:0] | SYSCR0 Key Code | 5Ah: Modifying SYSCR0 is enabled<br>Other codes: Modifying SYSCR0 is disabled<br>These bits are always read as 00h. | R/W |

SYSCR0 is used to enable or disable the on-chip ROM and the external bus.

#### ROME Bit (On-Chip FLASH Enable)

The ROME bit enables or disables the on-chip ROM (ROM, data flash).

While this bit is 1, it can be cleared to 0. While this bit is 0, it cannot be set to 1. Once the on-chip ROM is disabled by clearing this bit to 0, the on-chip ROM can no longer be enabled with the ROME bit. A 0 should not be written to this bit during access to the on-chip ROM.

After writing a 0 to this bit to disable the on-chip ROM, always make sure that the ROME bit has been changed to 0 before proceeding to the next processing.

#### EXBE Bit (External Bus Enable)

The EXBE bit enables* or disables the external bus.

Write 0 to this bit while the external bus cycle is not performed.

Be careful when disabling the external bus because the external bus and the internal bus are concurrently activated in some cases. When the EXBE bit is changed, the bus should be accessed after the change in the EXBE bit is completed.

When the EXBE bit is changed, the I/O port setting should also be changed simultaneously. For details, see section 14, I/O Ports.



Downloaded from Arrow.com.

**KEY[7:0] Bits (SYSCR0 Key Code)**

The KEY[7:0] bits enable or disable modifying SYSCR0.

When writing a value to the ROME or EXBE bit, write 5Ah to the KEY[7:0] bits simultaneously. If SYSCR0 is modified with a KEY[7:0] value other than 5Ah, the ROME and EXBE values remain unchanged.



Downloaded from Arrow.com.

### 3.2.4 System Control Register 1 (SYSCR1)

Address: 0008 0008h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RAME |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | RAME | RAM Enable | 0: The on-chip RAM is disabled<br>1: The on-chip RAM is enabled | R/W |
| b15 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

SYSCR1 is used to enable or disable the on-chip RAM.

**RAME Bit (RAM Enable)**

The RAME bit enables or disables the on-chip RAM.

The RAME bit is initialized to 1 after a reset is released.

A 0 should not be written to this bit during access to the on-chip RAM.

When accessing the on-chip RAM immediately after changing the RAME bit from 0 (on-chip RAM disabled) to 1 (on-chip RAM enabled), always make sure that the RAME bit is 1 before the access.

Even when the RAME bit is cleared to 0, the on-chip RAM retains its value*.

To retain the value in the on-chip RAM, keep the specified RAM standby voltage (VRAM). For details, see section 29, Electrical Characteristics.



Downloaded from Arrow.com.

## 3.3 Details of Operating Modes

### 3.3.1 Single-Chip Mode

In this mode, the on-chip ROM is enabled or disabled, the external bus is disabled (EXBE bit = 0 in SYSCR0), and all I/O ports can be used as input/output ports.

The on-chip ROM is enabled when this LSI is started. While the on-chip ROM is enabled (ROME bit = 1 in SYSCR0), it can be disabled by clearing the ROME bit to 0. While the on-chip ROM is disabled (ROME bit = 0), it cannot be enabled by setting the ROME bit to 1.

Setting the EXBE bit in SYSCR0 to 1 causes a transition to on-chip ROM enabled extended mode or on-chip ROM disabled extended mode where the external bus is available.

### 3.3.2 On-Chip ROM Enabled Extended Mode

In this mode, the on-chip ROM is enabled (ROME bit = 1 in SYSCR0) and the external bus is available as external extended mode (EXBE bit = 1 in SYSCR0). This mode allows some I/O ports to be used as data bus input/output, address bus output, or bus control signal input/output. For details, see section 14, I/O Ports.

The external bus width can be changed by the setting of external bus width selection (BSIZE[1:0] bits in CSiCNT (i = 0 to 7)). For details, see section 11, Buses.

Writing 0 to the EXBE bit causes a transition to single-chip mode (on-chip ROM enabled).

Writing 0 to the ROME bit causes a transition to on-chip ROM disabled extended mode.

### 3.3.3 On-Chip ROM Disabled Extended Mode

In this mode, the on-chip ROM is disabled (ROME bit = 0 in SYSCR0) and the external bus is available as external extended mode (EXBE bit = 1 in SYSCR0). This mode allows some I/O ports to be used as data bus input/output, address bus output, or bus control signal input/output. For details, see section 14, I/O Ports.

The external bus width can be changed by the setting of external bus width selection (BSIZE[1:0] bits in CSiCNT (i = 0 to 7)). For details, see section 11, Buses.

In this mode, the on-chip ROM cannot be enabled by setting the ROME bit to 1.

Writing 0 to the EXBE bit causes a transition to single-chip mode (on-chip ROM disabled).

### 3.3.4 Boot Mode

Boot mode is provided for the flash memory. This mode functions in the same manner as single-chip mode except for data write/erase to the flash memory. For details, see section 26, ROM (Flash Memory for Code Storage), and section 27, Data Flash (Flash Memory for Data Storage).

### 3.3.5 User Boot Mode

User boot mode is provided for the flash memory. This mode functions in the same manner as single-chip mode except for data write/erase to the flash memory. For details, see section 26, ROM (Flash Memory for Code Storage), and section 27, Data Flash (Flash Memory for Data Storage).



Downloaded from Arrow.com.

## 3.4 Transitions of Operating Modes

### 3.4.1 Operating Mode Transitions According to Mode Pin Setting

Figure 3.1 shows operating mode transitions according to the setting of pins MD1 and MD0. Operating modes can shift in the direction of arrow.

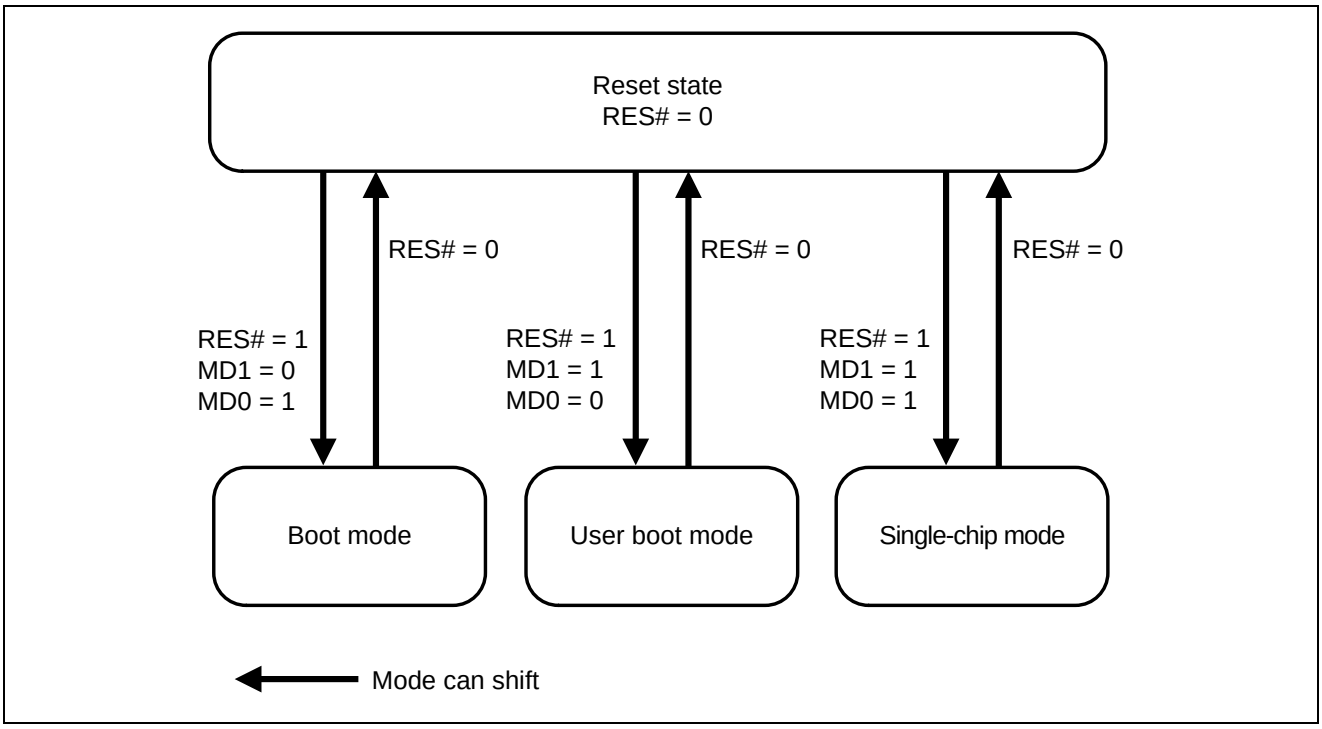



**Figure 3.1 Setting of Pins MD1 and MD0 and Operating Modes**



Downloaded from Arrow.com.

### 3.4.2 Operating Mode Transitions According to Register Setting

Figure 3.2 shows operating mode transitions according to the setting of the ROME and EXBE bits in SYSCR0. Operating modes can shift in the direction of arrow.

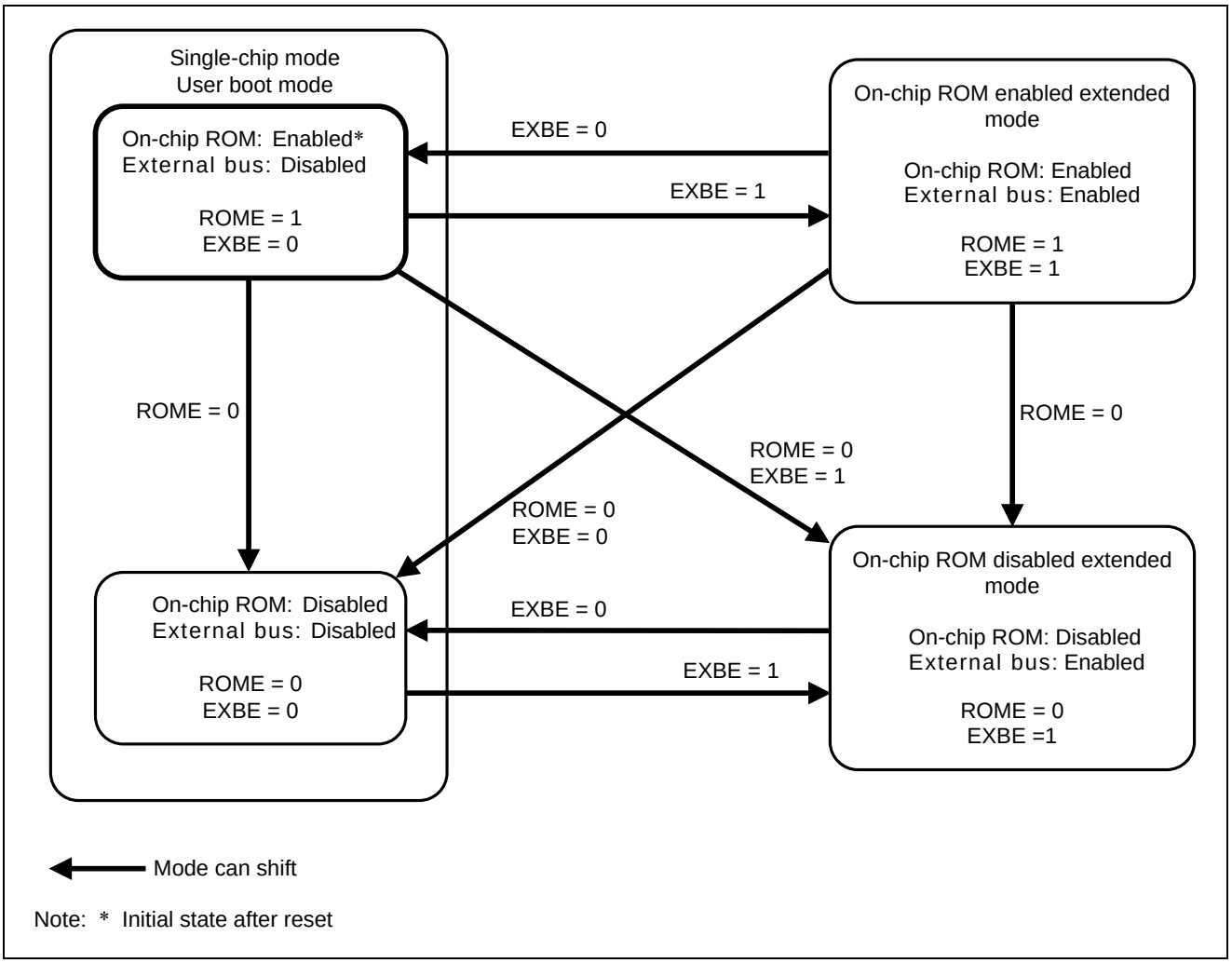



**Figure 3.2 Setting of Bits ROME and EXBE and Operating Modes**



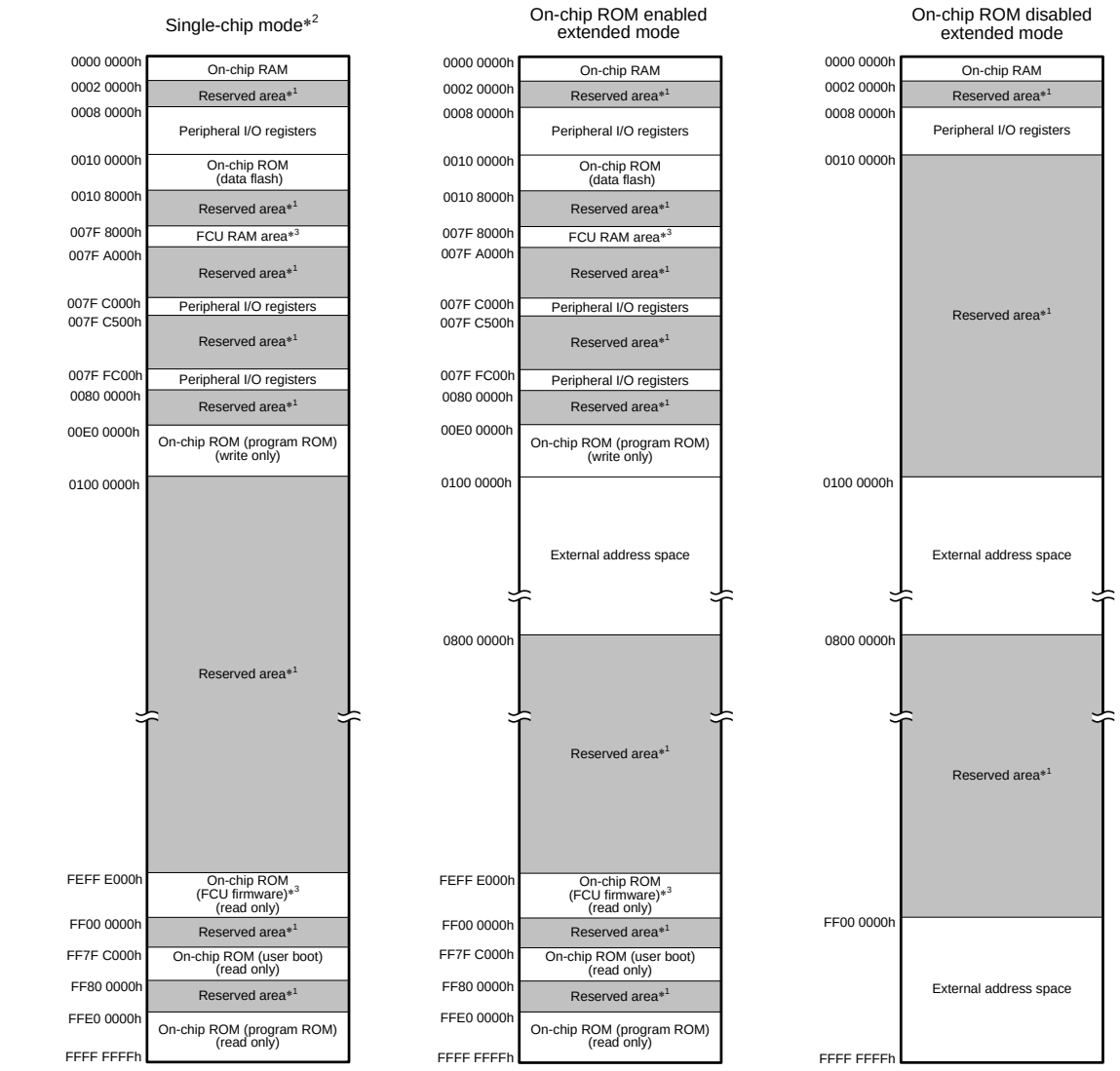
Downloaded from Arrow.com.

# 4. Address Space

## 4.1 Address Space

This MCU has a 4-Gbyte address space, consisting of the range of addresses from 0000 0000h to FFFF FFFFh. That is, linear access to an address space of up to 4 Gbytes is possible, and this contains both program and data areas.

Figures 4.1 to 4.4 show the memory maps in the respective operating modes of each product. Accessible areas will differ according to the operating mode and states of control bits.

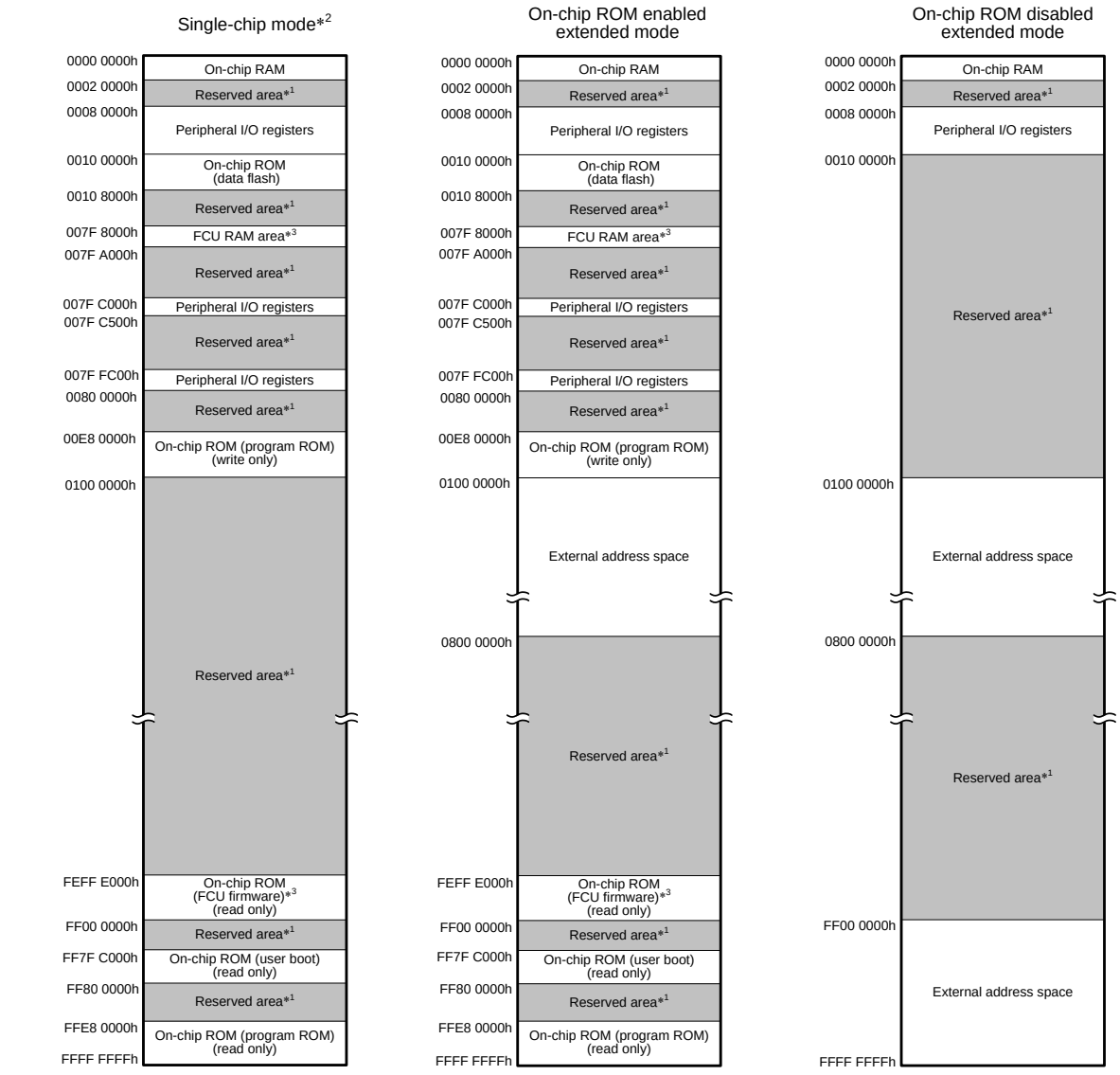



Notes: 1. A reserved area should not be accessed.
2. The address space in boot mode and user boot mode is the same as the address space in single-chip mode.
3. For details on the FCU, see section 26, ROM (Flash Memory for Code Storage) and section 27, Data Flash (Flash Memory for Data Storage).

**Figure 4.1 Memory Map of the R5F56108**



Downloaded from Arrow.com.

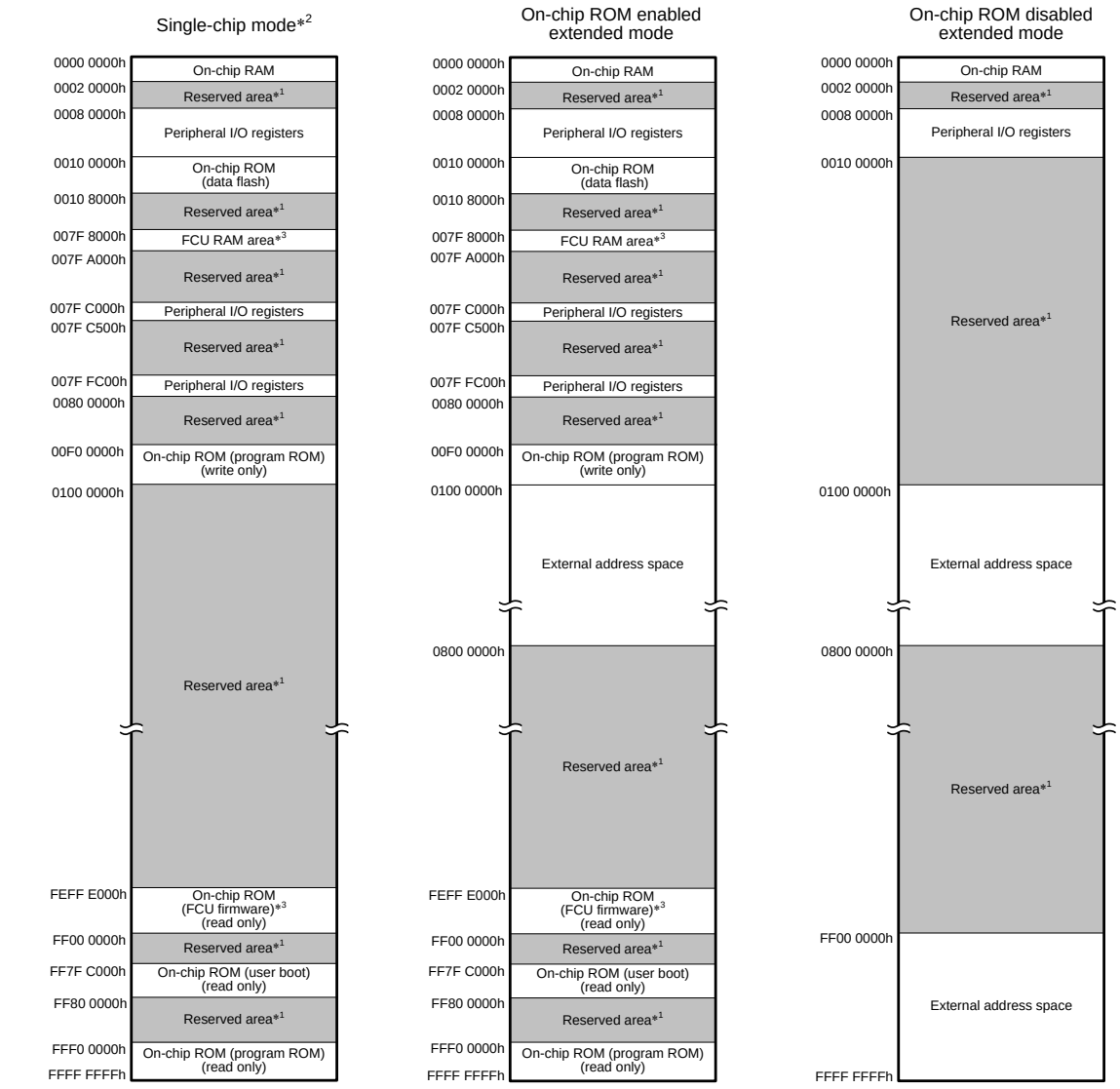



Notes: 1. A reserved area should not be accessed.
2. The address space in boot mode and user boot mode is he same as the address space in single-chip mode.
3. For details on the FCU, see section 26, ROM (Flash Memory for Code Storage) and section 27, Data Flash (Flash Memory for Data Storage).

**Figure 4.2 Memory Map of the R5F56107**



Downloaded from Arrow.com.

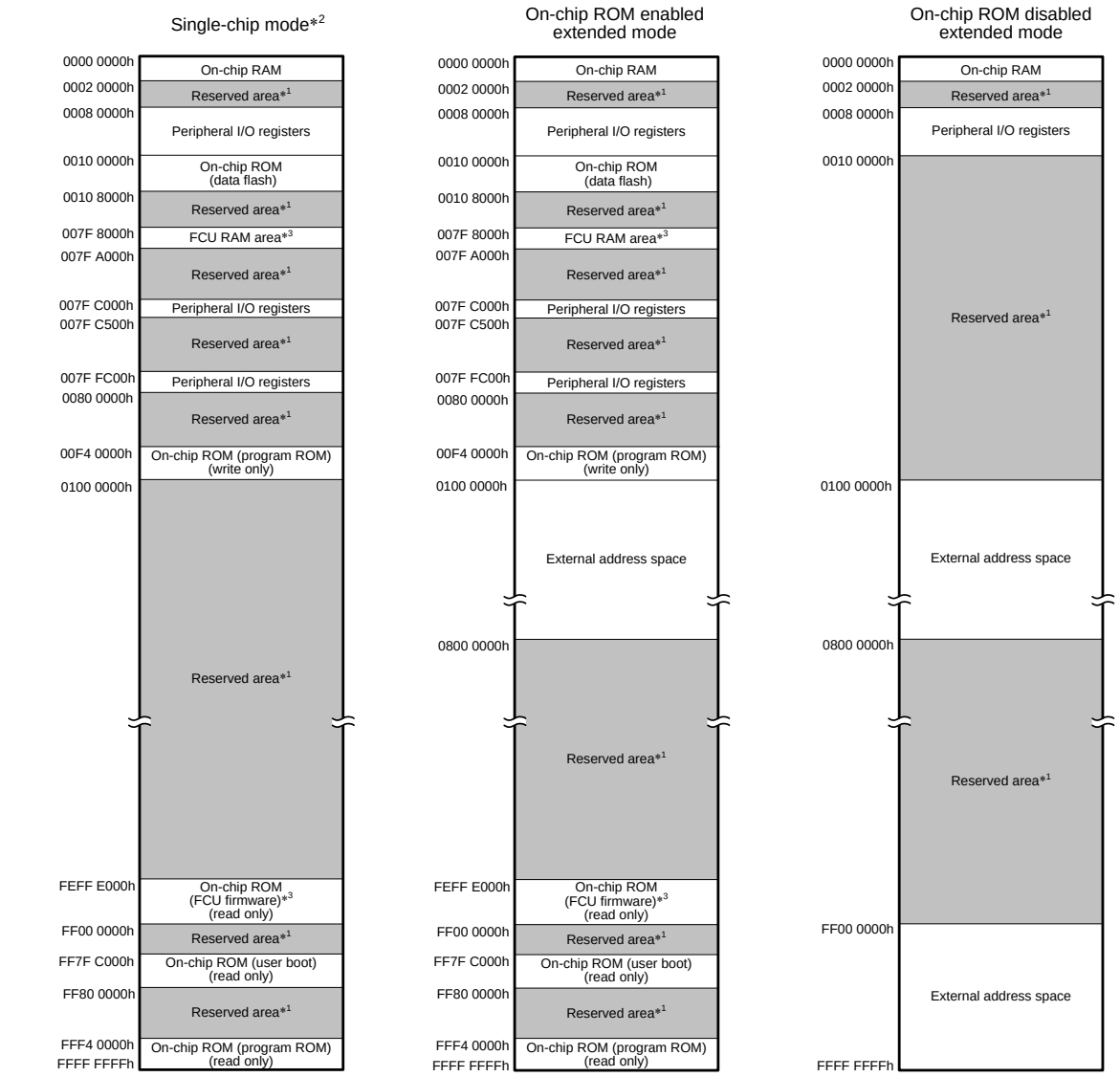



Notes: 1. A reserved area should not be accessed.
2. The address space in boot mode and user boot mode is the same as the address space in single-chip mode.
3. For details on the FCU, see section 26, ROM (Flash Memory for Code Storage) and section 27, Data Flash (Flash Memory for Data Storage).

**Figure 4.3 Memory Map of the R5F56106**



Downloaded from Arrow.com.




Notes: 1. A reserved area should not be accessed.
2. The address space in boot mode and user boot mode is the same as the address space in single-chip mode.
3. For details on the FCU, see section 26, ROM (Flash Memory for Code Storage) and section 27, Data Flash (Flash Memory for Data Storage).

**Figure 4.4 Memory Map of the R5F56104**



Downloaded from Arrow.com.

## 4.2 External Address Space

The external address space is divided into up to 8 areas, each corresponding to the CSi# signal output from a CSi# (i = 0 to 7) pin. Figure 4.5 shows the address ranges corresponding to the individual CSi# signals (CSi areas, i = 0 to 7) in on-chip ROM disabled external extended mode.

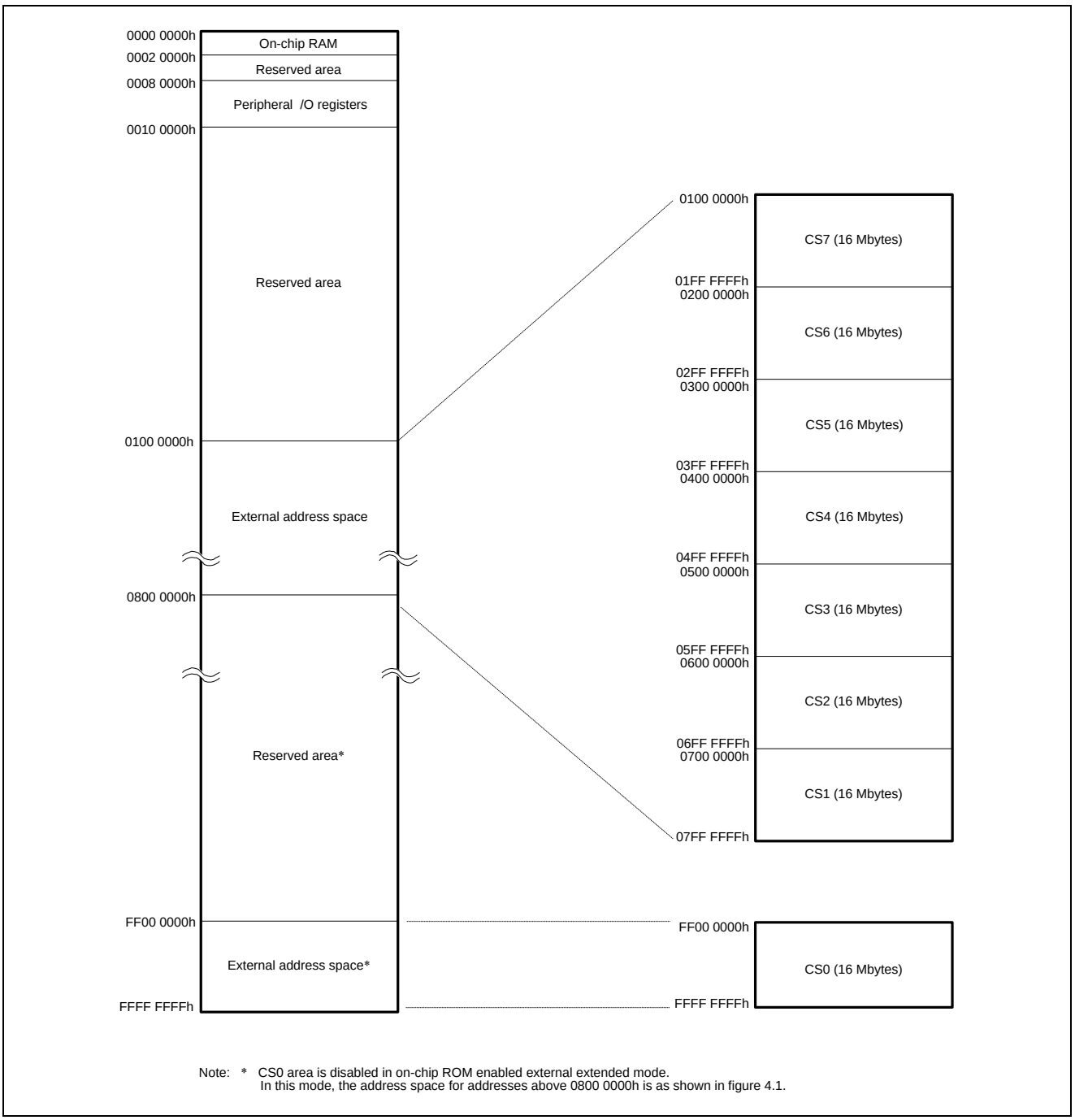



**Figure 4.5 Correspondence between External Address Spaces and CSi Areas (In On-Chip ROM Disabled External Extended Mode)**



Downloaded from Arrow.com.

# 5. I/O Registers

This section gives information on the on-chip I/O register addresses and bit configurations. The information is given as shown below. Notes on writing to registers are also given at the end.

1. I/O register addresses (address order)

- Registers are listed from the lower allocation addresses.
- Registers are classified according to functional modules (abbreviations).
- The number of access cycles indicates the number of states based on the specified reference clock.
- Among the I/O register area, addresses not listed in the list of registers are reserved. Reserved addresses must not be accessed. Do not access these addresses; otherwise, the operation when accessing these bits and subsequent operations cannot be guaranteed.
- The access size is specified for each register. The registers must be accessed with the specified access size.

2. I/O register bits

- Bit configurations of the registers are listed in the same order as the register addresses.
- Reserved bits are indicated by "—" in the bit name column.
- Space in the bit name field indicates that the entire register is allocated to either the counter or data.
- For the registers of 16 or 32 bits, the MSB is listed first.

3. Notes on writing to I/O registers

When writing to an I/O register, the CPU starts executing the subsequent instruction before completing I/O register write. This may cause the subsequent instruction to be executed before the post-update I/O register value is reflected on the operation.

As described in the following examples, special care is required for the cases in which the subsequent instruction must be executed after the post-update I/O register value is actually reflected.

[Examples of cases requiring special care]

- The subsequent instruction must be executed while an interrupt request is disabled with the IENj bit in IERm of the ICU (interrupt request enable bit) cleared to 0.
- A WAIT instruction is executed immediately after the preprocessing for causing a transition to the low power consumption state.

In the above cases, after writing to an I/O register, wait until the write operation is completed using the following procedure and then execute the subsequent instruction.

(a) Write to an I/O register.

(b) Read the value from the I/O register to a general register.

(c) Execute the operation using the value read.

(d) Execute the subsequent instruction.



Downloaded from Arrow.com.

[Instruction examples]

- Byte-size I/O registers

```
MOV.L   #SFR_ADDR, R1
MOV.B   #SFR_DATA, [R1]
CMP     [R1].UB, R1
;; Next process
```

- Word-size I/O registers

```
MOV.L   #SFR_ADDR, R1
MOV.W   #SFR_DATA, [R1]
CMP     [R1].W, R1
;; Next process
```

- Longword-size I/O registers

```
MOV.L   #SFR_ADDR, R1
MOV.L   #SFR_DATA, [R1]
CMP     [R1].L, R1
;; Next process
```

If multiple registers are written to and a subsequent instruction should be executed after the write operations are entirely completed, only read the I/O register that was last written to and execute the operation using the value; it is not necessary to read or execute operation for al the registers that were written to.

4. Number of Access Cycles to I/O Registers

The number of access cycles to I/O registers is obtained by following equation.*

Number of access cycles to I/O registers = Number of bus cycles for internal main bus 1 +
Number of divided cycles for clock synchronization +
Number of bus cycles for internal peripheral bus 1 (or 2)

The number of bus cycles of internal peripheral bus 1 (or 2) differs according to the register to be accessed. For the number of access cycles to each I/O register, see table 5.1, List of I/O Registers.

When peripheral functions connected to internal peripheral bus 2 or registers for the external bus control unit (except for bus error related registers) are accessed, the number of divided cycles for clock synchronization is added.

Although the number of divided cycles for clock synchronization differs depending on the number of frequency ratio or bus access timing, the sum of the number of bus cycles for internal main bus 1 and the number of divided cycles for clock synchronization will be one PCLK (or BCLK) at a maximum. Therefore, one PCLK (or BCLK) is added to the number of access cycles shown in table 5.1.

Note: * This applies to the number of cycles when the access from the CPU does not conflict with the instruction fetching to the external memory or bus access from the different bus master (DMAC or DTC).



Downloaded from Arrow.com.

## 5.1 I/O Register Addresses (Address Order)

**Table 5.1 List of I/O Registers (Address Order)**

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 0000h | SYSTEM | Mode monitor register | MDMONR | 16 | 16 | 3 ICLK |
| 0008 0002h | SYSTEM | Mode status register | MDSR | 16 | 16 | 3 ICLK |
| 0008 0006h | SYSTEM | System control register 0 | SYSCR0 | 16 | 16 | 3 ICLK |
| 0008 0008h | SYSTEM | System control register 1 | SYSCR1 | 16 | 16 | 3 ICLK |
| 0008 000Ch | SYSTEM | Standby control register | SBYCR | 16 | 16 | 3 ICLK |
| 0008 0010h | SYSTEM | Module stop control register A | MSTPCRA | 32 | 32 | 3 ICLK |
| 0008 0014h | SYSTEM | Module stop control register B | MSTPCRB | 32 | 32 | 3 ICLK |
| 0008 0018h | SYSTEM | Module stop control register C | MSTPCRC | 32 | 32 | 3 ICLK |
| 0008 0020h | SYSTEM | System clock control register | SCKCR | 32 | 32 | 3 ICLK |
| 0008 1300h | BSC | Bus error source clear register | BERCLR | 8 | 8 | 2 ICLK |
| 0008 1304h | BSC | Bus error monitor enable register | BEREN | 8 | 8 | 2 ICLK |
| 0008 1306h | BSC | Bus error interrupt enable register | BERIE | 8 | 8 | 2 ICLK |
| 0008 2000h | DMAC0 | DMA current transfer source address register | DMCSA | 32 | 32 | 4 to 5 ICLK |
| 0008 2004h | DMAC0 | DMA current transfer destination address register | DMCDA | 32 | 32 | 4 to 5 ICLK |
| 0008 2008h | DMAC0 | DMA current transfer byte count register | DMCBC | 32 | 32 | 4 to 5 ICLK |
| 0008 200Ch | DMAC0 | DMA mode register | DMMOD | 32 | 32 | 4 to 5 ICLK |
| 0008 2010h | DMAC1 | DMA current transfer source address register | DMCSA | 32 | 32 | 4 to 5 ICLK |
| 0008 2014h | DMAC1 | DMA current transfer destination address register | DMCDA | 32 | 32 | 4 to 5 ICLK |
| 0008 2018h | DMAC1 | DMA current transfer byte count register | DMCBC | 32 | 32 | 4 to 5 ICLK |
| 0008 201Ch | DMAC1 | DMA mode register | DMMOD | 32 | 32 | 4 to 5 ICLK |
| 0008 2020h | DMAC2 | DMA current transfer source address register | DMCSA | 32 | 32 | 4 to 5 ICLK |
| 0008 2024h | DMAC2 | DMA current transfer destination address register | DMCDA | 32 | 32 | 4 to 5 ICLK |
| 0008 2028h | DMAC2 | DMA current transfer byte count register | DMCBC | 32 | 32 | 4 to 5 ICLK |
| 0008 202Ch | DMAC2 | DMA mode register | DMMOD | 32 | 32 | 4 to 5 ICLK |
| 0008 2030h | DMAC3 | DMA current transfer source address register | DMCSA | 32 | 32 | 4 to 5 ICLK |
| 0008 2034h | DMAC3 | DMA current transfer destination address register | DMCDA | 32 | 32 | 4 to 5 ICLK |
| 0008 2038h | DMAC3 | DMA current transfer byte count register | DMCBC | 32 | 32 | 4 to 5 ICLK |
| 0008 203Ch | DMAC3 | DMA mode register | DMMOD | 32 | 32 | 4 to 5 ICLK |
| 0008 2200h | DMAC0 | DMA reload transfer source address register | DMRSA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2204h | DMAC0 | DMA reload transfer destination address register | DMRDA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2208h | DMAC0 | DMA reload transfer byte count register | DMRBC | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2210h | DMAC1 | DMA reload transfer source address register | DMRSA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2214h | DMAC1 | DMA reload transfer destination address register | DMRDA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2218h | DMAC1 | DMA reload transfer byte count register | DMRBC | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2220h | DMAC2 | DMA reload transfer source address register | DMRSA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2224h | DMAC2 | DMA reload transfer destination address register | DMRDA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2228h | DMAC2 | DMA reload transfer byte count register | DMRBC | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2230h | DMAC3 | DMA reload transfer source address register | DMRSA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2234h | DMAC3 | DMA reload transfer destination address register | DMRDA | 32 | 32 | 4 to 5 ICLK *8 |
| 0008 2238h | DMAC3 | DMA reload transfer byte count register | DMRBC | 32 | 32 | 4 to 5 ICLK *8 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 2400h | DMAC0 | DMA control register A | DMCRA | 32 | 32 | 3 ICLK |
| 0008 2404h | DMAC0 | DMA control register B | DMCRB | 8 | 8 | 3 ICLK |
| 0008 2405h | DMAC0 | DMA control register C | DMCRC | 8 | 8 | 3 ICLK |
| 0008 2406h | DMAC0 | DMA control register D | DMCRD | 8 | 8 | 3 ICLK |
| 0008 2407h | DMAC0 | DMA control register E | DMCRE | 8 | 8 | 3 ICLK |
| 0008 2408h | DMAC1 | DMA control register A | DMCRA | 32 | 32 | 3 ICLK |
| 0008 240Ch | DMAC1 | DMA control register B | DMCRB | 8 | 8 | 3 ICLK |
| 0008 240Dh | DMAC1 | DMA control register C | DMCRC | 8 | 8 | 3 ICLK |
| 0008 240Eh | DMAC1 | DMA control register D | DMCRD | 8 | 8 | 3 ICLK |
| 0008 240Fh | DMAC1 | DMA control register E | DMCRE | 8 | 8 | 3 ICLK |
| 0008 2410h | DMAC2 | DMA control register A | DMCRA | 32 | 32 | 3 ICLK |
| 0008 2414h | DMAC2 | DMA control register B | DMCRB | 8 | 8 | 3 ICLK |
| 0008 2415h | DMAC2 | DMA control register C | DMCRC | 8 | 8 | 3 ICLK |
| 0008 2416h | DMAC2 | DMA control register D | DMCRD | 8 | 8 | 3 ICLK |
| 0008 2417h | DMAC2 | DMA control register E | DMCRE | 8 | 8 | 3 ICLK |
| 0008 2418h | DMAC3 | DMA control register A | DMCRA | 32 | 32 | 3 ICLK |
| 0008 241Ch | DMAC3 | DMA control register B | DMCRB | 8 | 8 | 3 ICLK |
| 0008 241Dh | DMAC3 | DMA control register C | DMCRC | 8 | 8 | 3 ICLK |
| 0008 241Eh | DMAC3 | DMA control register D | DMCRD | 8 | 8 | 3 ICLK |
| 0008 241Fh | DMAC3 | DMA control register E | DMCRE | 8 | 8 | 3 ICLK |
| 0008 2502h | DMAC common | DMA start control register | DMSCNT | 8 | 8 | 3 ICLK |
| 0008 250Bh | DMAC common | DMA interrupt control register | DMICNT | 8 | 8 | 3 ICLK |
| 0008 2517h | DMAC common | DMA transfer end detect register | DMEDET | 8 | 8 | 3 ICLK |
| 0008 251Bh | DMAC common | DMA arbitration status register | DMASTS | 8 | 8 | 3 ICLK |
| 0008 3002h | BSC | CS0 mode register | CS0MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3004h | BSC | CS0 wait control register 1 | CS0WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3008h | BSC | CS0 wait control register 2 | CS0WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3012h | BSC | CS1 mode register | CS1MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3014h | BSC | CS1 wait control register 1 | CS1WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3018h | BSC | CS1 wait control register 2 | CS1WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3022h | BSC | CS2 mode register | CS2MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3024h | BSC | CS2 wait control register 1 | CS2WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3028h | BSC | CS2 wait control register 2 | CS2WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3032h | BSC | CS3 mode register | CS3MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3034h | BSC | CS3 wait control register 1 | CS3WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3038h | BSC | CS3 wait control register 2 | CS3WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3042h | BSC | CS4 mode register | CS4MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3044h | BSC | CS4 wait control register 1 | CS4WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3048h | BSC | CS4 wait control register 2 | CS4WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3052h | BSC | CS5 mode register | CS5MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3054h | BSC | CS5 wait control register 1 | CS5WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3058h | BSC | CS5 wait control register 2 | CS5WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3062h | BSC | CS6 mode register | CS6MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3064h | BSC | CS6 wait control register 1 | CS6WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3068h | BSC | CS6 wait control register 2 | CS6WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 3072h | BSC | CS7 mode register | CS7MOD | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3074h | BSC | CS7 wait control register 1 | CS7WCNT1 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3078h | BSC | CS7 wait control register 2 | CS7WCNT2 | 32 | 32 | 1 to 2 BCLK *7 |
| 0008 3802h | BSC | CS0 control register | CS0CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 380Ah | BSC | CS0 recovery cycle register | CS0REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3812h | BSC | CS1 control register | CS1CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 381Ah | BSC | CS1 recovery cycle register | CS1REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3822h | BSC | CS2 control register | CS2CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 382Ah | BSC | CS2 recovery cycle register | CS2REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3832h | BSC | CS3 control register | CS3CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 383Ah | BSC | CS3 recovery cycle register | CS3REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3842h | BSC | CS4 control register | CS4CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 384Ah | BSC | CS4 recovery cycle register | CS4REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3852h | BSC | CS5 control register | CS5CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 385Ah | BSC | CS5 recovery cycle register | CS5REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3862h | BSC | CS6 control register | CS6CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 386Ah | BSC | CS6 recovery cycle register | CS6REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 3872h | BSC | CS7 control register | CS7CNT | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 387Ah | BSC | CS7 recovery cycle register | CS7REC | 16 | 16 | 1 to 2 BCLK *7 |
| 0008 7010h | ICU | Interrupt request register 016 | IR016 | 8 | 8 | 2 ICLK |
| 0008 7015h | ICU | Interrupt request register 021 | IR021 | 8 | 8 | 2 ICLK |
| 0008 7017h | ICU | Interrupt request register 023 | IR023 | 8 | 8 | 2 ICLK |
| 0008 701Ch | ICU | Interrupt request register 028 | IR028 | 8 | 8 | 2 ICLK |
| 0008 701Dh | ICU | Interrupt request register 029 | IR029 | 8 | 8 | 2 ICLK |
| 0008 701Eh | ICU | Interrupt request register 030 | IR030 | 8 | 8 | 2 ICLK |
| 0008 701Fh | ICU | Interrupt request register 031 | IR031 | 8 | 8 | 2 ICLK |
| 0008 7040h | ICU | Interrupt request register 064 | IR064 | 8 | 8 | 2 ICLK |
| 0008 7041h | ICU | Interrupt request register 065 | IR065 | 8 | 8 | 2 ICLK |
| 0008 7042h | ICU | Interrupt request register 066 | IR066 | 8 | 8 | 2 ICLK |
| 0008 7043h | ICU | Interrupt request register 067 | IR067 | 8 | 8 | 2 ICLK |
| 0008 7044h | ICU | Interrupt request register 068 | IR068 | 8 | 8 | 2 ICLK |
| 0008 7045h | ICU | Interrupt request register 069 | IR069 | 8 | 8 | 2 ICLK |
| 0008 7046h | ICU | Interrupt request register 070 | IR070 | 8 | 8 | 2 ICLK |
| 0008 7047h | ICU | Interrupt request register 071 | IR071 | 8 | 8 | 2 ICLK |
| 0008 7048h | ICU | Interrupt request register 072 | IR072 | 8 | 8 | 2 ICLK |
| 0008 7049h | ICU | Interrupt request register 073 | IR073 | 8 | 8 | 2 ICLK |
| 0008 704Ah | ICU | Interrupt request register 074 | IR074 | 8 | 8 | 2 ICLK |
| 0008 704Bh | ICU | Interrupt request register 075 | IR075 | 8 | 8 | 2 ICLK |
| 0008 704Ch | ICU | Interrupt request register 076 | IR076 | 8 | 8 | 2 ICLK |
| 0008 704Dh | ICU | Interrupt request register 077 | IR077 | 8 | 8 | 2 ICLK |
| 0008 704Eh | ICU | Interrupt request register 078 | IR078 | 8 | 8 | 2 ICLK |
| 0008 704Fh | ICU | Interrupt request register 079 | IR079 | 8 | 8 | 2 ICLK |
| 0008 7060h | ICU | Interrupt request register 096 | IR096 | 8 | 8 | 2 ICLK |
| 0008 7062h | ICU | Interrupt request register 098 | IR098 | 8 | 8 | 2 ICLK |
| 0008 7063h | ICU | Interrupt request register 099 | IR099 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 7064h | ICU | Interrupt request register 100 | IR100 | 8 | 8 | 2 ICLK |
| 0008 7065h | ICU | Interrupt request register 101 | IR101 | 8 | 8 | 2 ICLK |
| 0008 7068h | ICU | Interrupt request register 104 | IR104 | 8 | 8 | 2 ICLK |
| 0008 7069h | ICU | Interrupt request register 105 | IR105 | 8 | 8 | 2 ICLK |
| 0008 706Ah | ICU | Interrupt request register 106 | IR106 | 8 | 8 | 2 ICLK |
| 0008 706Bh | ICU | Interrupt request register 107 | IR107 | 8 | 8 | 2 ICLK |
| 0008 706Ch | ICU | Interrupt request register 108 | IR108 | 8 | 8 | 2 ICLK |
| 0008 706Fh | ICU | Interrupt request register 111 | IR111 | 8 | 8 | 2 ICLK |
| 0008 7070h | ICU | Interrupt request register 112 | IR112 | 8 | 8 | 2 ICLK |
| 0008 7073h | ICU | Interrupt request register 115 | IR115 | 8 | 8 | 2 ICLK |
| 0008 7074h | ICU | Interrupt request register 116 | IR116 | 8 | 8 | 2 ICLK |
| 0008 7075h | ICU | Interrupt request register 117 | IR117 | 8 | 8 | 2 ICLK |
| 0008 7076h | ICU | Interrupt request register 118 | IR118 | 8 | 8 | 2 ICLK |
| 0008 7078h | ICU | Interrupt request register 120 | IR120 | 8 | 8 | 2 ICLK |
| 0008 7079h | ICU | Interrupt request register 121 | IR121 | 8 | 8 | 2 ICLK |
| 0008 707Ah | ICU | Interrupt request register 122 | IR122 | 8 | 8 | 2 ICLK |
| 0008 707Bh | ICU | Interrupt request register 123 | IR123 | 8 | 8 | 2 ICLK |
| 0008 707Ch | ICU | Interrupt request register 124 | IR124 | 8 | 8 | 2 ICLK |
| 0008 707Dh | ICU | Interrupt request register 125 | IR125 | 8 | 8 | 2 ICLK |
| 0008 707Eh | ICU | Interrupt request register 126 | IR126 | 8 | 8 | 2 ICLK |
| 0008 707Fh | ICU | Interrupt request register 127 | IR127 | 8 | 8 | 2 ICLK |
| 0008 7080h | ICU | Interrupt request register 128 | IR128 | 8 | 8 | 2 ICLK |
| 0008 7083h | ICU | Interrupt request register 131 | IR131 | 8 | 8 | 2 ICLK |
| 0008 7084h | ICU | Interrupt request register 132 | IR132 | 8 | 8 | 2 ICLK |
| 0008 7085h | ICU | Interrupt request register 133 | IR133 | 8 | 8 | 2 ICLK |
| 0008 7086h | ICU | Interrupt request register 134 | IR134 | 8 | 8 | 2 ICLK |
| 0008 7088h | ICU | Interrupt request register 136 | IR136 | 8 | 8 | 2 ICLK |
| 0008 7089h | ICU | Interrupt request register 137 | IR137 | 8 | 8 | 2 ICLK |
| 0008 708Ah | ICU | Interrupt request register 138 | IR138 | 8 | 8 | 2 ICLK |
| 0008 708Bh | ICU | Interrupt request register 139 | IR139 | 8 | 8 | 2 ICLK |
| 0008 708Ch | ICU | Interrupt request register 140 | IR140 | 8 | 8 | 2 ICLK |
| 0008 708Dh | ICU | Interrupt request register 141 | IR141 | 8 | 8 | 2 ICLK |
| 0008 708Eh | ICU | Interrupt request register 142 | IR142 | 8 | 8 | 2 ICLK |
| 0008 7091h | ICU | Interrupt request register 145 | IR145 | 8 | 8 | 2 ICLK |
| 0008 7092h | ICU | Interrupt request register 146 | IR146 | 8 | 8 | 2 ICLK |
| 0008 7095h | ICU | Interrupt request register 149 | IR149 | 8 | 8 | 2 ICLK |
| 0008 7096h | ICU | Interrupt request register 150 | IR150 | 8 | 8 | 2 ICLK |
| 0008 7097h | ICU | Interrupt request register 151 | IR151 | 8 | 8 | 2 ICLK |
| 0008 7098h | ICU | Interrupt request register 152 | IR152 | 8 | 8 | 2 ICLK |
| 0008 709Ah | ICU | Interrupt request register 154 | IR154 | 8 | 8 | 2 ICLK |
| 0008 709Bh | ICU | Interrupt request register 155 | IR155 | 8 | 8 | 2 ICLK |
| 0008 709Ch | ICU | Interrupt request register 156 | IR156 | 8 | 8 | 2 ICLK |
| 0008 709Dh | ICU | Interrupt request register 157 | IR157 | 8 | 8 | 2 ICLK |
| 0008 709Eh | ICU | Interrupt request register 158 | IR158 | 8 | 8 | 2 ICLK |
| 0008 709Fh | ICU | Interrupt request register 159 | IR159 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 70A0h | ICU | Interrupt request register 160 | IR160 | 8 | 8 | 2 ICLK |
| 0008 70A1h | ICU | Interrupt request register 161 | IR161 | 8 | 8 | 2 ICLK |
| 0008 70A2h | ICU | Interrupt request register 162 | IR162 | 8 | 8 | 2 ICLK |
| 0008 70A5h | ICU | Interrupt request register 165 | IR165 | 8 | 8 | 2 ICLK |
| 0008 70A6h | ICU | Interrupt request register 166 | IR166 | 8 | 8 | 2 ICLK |
| 0008 70A7h | ICU | Interrupt request register 167 | IR167 | 8 | 8 | 2 ICLK |
| 0008 70A8h | ICU | Interrupt request register 168 | IR168 | 8 | 8 | 2 ICLK |
| 0008 70AAh | ICU | Interrupt request register 170 | IR170 | 8 | 8 | 2 ICLK |
| 0008 70ABh | ICU | Interrupt request register 171 | IR171 | 8 | 8 | 2 ICLK |
| 0008 70AEh | ICU | Interrupt request register 174 | IR174 | 8 | 8 | 2 ICLK |
| 0008 70AFh | ICU | Interrupt request register 175 | IR175 | 8 | 8 | 2 ICLK |
| 0008 70B0h | ICU | Interrupt request register 176 | IR176 | 8 | 8 | 2 ICLK |
| 0008 70B1h | ICU | Interrupt request register 177 | IR177 | 8 | 8 | 2 ICLK |
| 0008 70B2h | ICU | Interrupt request register 178 | IR178 | 8 | 8 | 2 ICLK |
| 0008 70B3h | ICU | Interrupt request register 179 | IR179 | 8 | 8 | 2 ICLK |
| 0008 70B4h | ICU | Interrupt request register 180 | IR180 | 8 | 8 | 2 ICLK |
| 0008 70B5h | ICU | Interrupt request register 181 | IR181 | 8 | 8 | 2 ICLK |
| 0008 70B6h | ICU | Interrupt request register 182 | IR182 | 8 | 8 | 2 ICLK |
| 0008 70B7h | ICU | Interrupt request register 183 | IR183 | 8 | 8 | 2 ICLK |
| 0008 70B8h | ICU | Interrupt request register 184 | IR184 | 8 | 8 | 2 ICLK |
| 0008 70B9h | ICU | Interrupt request register 185 | IR185 | 8 | 8 | 2 ICLK |
| 0008 70C6h | ICU | Interrupt request register 198 | IR198 | 8 | 8 | 2 ICLK |
| 0008 70C7h | ICU | Interrupt request register 199 | IR199 | 8 | 8 | 2 ICLK |
| 0008 70C8h | ICU | Interrupt request register 200 | IR200 | 8 | 8 | 2 ICLK |
| 0008 70C9h | ICU | Interrupt request register 201 | IR201 | 8 | 8 | 2 ICLK |
| 0008 70D6h | ICU | Interrupt request register 214 | IR214 | 8 | 8 | 2 ICLK |
| 0008 70D7h | ICU | Interrupt request register 215 | IR215 | 8 | 8 | 2 ICLK |
| 0008 70D8h | ICU | Interrupt request register 216 | IR216 | 8 | 8 | 2 ICLK |
| 0008 70D9h | ICU | Interrupt request register 217 | IR217 | 8 | 8 | 2 ICLK |
| 0008 70DAh | ICU | Interrupt request register 218 | IR218 | 8 | 8 | 2 ICLK |
| 0008 70DBh | ICU | Interrupt request register 219 | IR219 | 8 | 8 | 2 ICLK |
| 0008 70DCh | ICU | Interrupt request register 220 | IR220 | 8 | 8 | 2 ICLK |
| 0008 70DDh | ICU | Interrupt request register 221 | IR221 | 8 | 8 | 2 ICLK |
| 0008 70DEh | ICU | Interrupt request register 222 | IR222 | 8 | 8 | 2 ICLK |
| 0008 70DFh | ICU | Interrupt request register 223 | IR223 | 8 | 8 | 2 ICLK |
| 0008 70E0h | ICU | Interrupt request register 224 | IR224 | 8 | 8 | 2 ICLK |
| 0008 70E1h | ICU | Interrupt request register 225 | IR225 | 8 | 8 | 2 ICLK |
| 0008 70E2h | ICU | Interrupt request register 226 | IR226 | 8 | 8 | 2 ICLK |
| 0008 70E3h | ICU | Interrupt request register 227 | IR227 | 8 | 8 | 2 ICLK |
| 0008 70E4h | ICU | Interrupt request register 228 | IR228 | 8 | 8 | 2 ICLK |
| 0008 70E5h | ICU | Interrupt request register 229 | IR229 | 8 | 8 | 2 ICLK |
| 0008 70E6h | ICU | Interrupt request register 230 | IR230 | 8 | 8 | 2 ICLK |
| 0008 70E7h | ICU | Interrupt request register 231 | IR231 | 8 | 8 | 2 ICLK |
| 0008 70E8h | ICU | Interrupt request register 232 | IR232 | 8 | 8 | 2 ICLK |
| 0008 70E9h | ICU | Interrupt request register 233 | IR233 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 70EAh | ICU | Interrupt request register 234 | IR234 | 8 | 8 | 2 ICLK |
| 0008 70EBh | ICU | Interrupt request register 235 | IR235 | 8 | 8 | 2 ICLK |
| 0008 70ECh | ICU | Interrupt request register 236 | IR236 | 8 | 8 | 2 ICLK |
| 0008 70EDh | ICU | Interrupt request register 237 | IR237 | 8 | 8 | 2 ICLK |
| 0008 70EEh | ICU | Interrupt request register 238 | IR238 | 8 | 8 | 2 ICLK |
| 0008 70EFh | ICU | Interrupt request register 239 | IR239 | 8 | 8 | 2 ICLK |
| 0008 70F0h | ICU | Interrupt request register 240 | IR240 | 8 | 8 | 2 ICLK |
| 0008 70F1h | ICU | Interrupt request register 241 | IR241 | 8 | 8 | 2 ICLK |
| 0008 70F6h | ICU | Interrupt request register 246 | IR246 | 8 | 8 | 2 ICLK |
| 0008 70F7h | ICU | Interrupt request register 247 | IR247 | 8 | 8 | 2 ICLK |
| 0008 70F8h | ICU | Interrupt request register 248 | IR248 | 8 | 8 | 2 ICLK |
| 0008 70F9h | ICU | Interrupt request register 249 | IR249 | 8 | 8 | 2 ICLK |
| 0008 70FAh | ICU | Interrupt request register 250 | IR250 | 8 | 8 | 2 ICLK |
| 0008 70FBh | ICU | Interrupt request register 251 | IR251 | 8 | 8 | 2 ICLK |
| 0008 70FCh | ICU | Interrupt request register 252 | IR252 | 8 | 8 | 2 ICLK |
| 0008 70FDh | ICU | Interrupt request register 253 | IR253 | 8 | 8 | 2 ICLK |
| 0008 711Ch | ICU | Interrupt request destination setting register 028 | ISELR028 | 8 | 8 | 2 ICLK |
| 0008 711Dh | ICU | Interrupt request destination setting register 029 | ISELR029 | 8 | 8 | 2 ICLK |
| 0008 711Eh | ICU | Interrupt request destination setting register 030 | ISELR030 | 8 | 8 | 2 ICLK |
| 0008 711Fh | ICU | Interrupt request destination setting register 031 | ISELR031 | 8 | 8 | 2 ICLK |
| 0008 7140h | ICU | Interrupt request destination setting register 064 | ISELR064 | 8 | 8 | 2 ICLK |
| 0008 7141h | ICU | Interrupt request destination setting register 065 | ISELR065 | 8 | 8 | 2 ICLK |
| 0008 7142h | ICU | Interrupt request destination setting register 066 | ISELR066 | 8 | 8 | 2 ICLK |
| 0008 7143h | ICU | Interrupt request destination setting register 067 | ISELR067 | 8 | 8 | 2 ICLK |
| 0008 7144h | ICU | Interrupt request destination setting register 068 | ISELR068 | 8 | 8 | 2 ICLK |
| 0008 7145h | ICU | Interrupt request destination setting register 069 | ISELR069 | 8 | 8 | 2 ICLK |
| 0008 7146h | ICU | Interrupt request destination setting register 070 | ISELR070 | 8 | 8 | 2 ICLK |
| 0008 7147h | ICU | Interrupt request destination setting register 071 | ISELR071 | 8 | 8 | 2 ICLK |
| 0008 7148h | ICU | Interrupt request destination setting register 072 | ISELR072 | 8 | 8 | 2 ICLK |
| 0008 7149h | ICU | Interrupt request destination setting register 073 | ISELR073 | 8 | 8 | 2 ICLK |
| 0008 714Ah | ICU | Interrupt request destination setting register 074 | ISELR074 | 8 | 8 | 2 ICLK |
| 0008 714Bh | ICU | Interrupt request destination setting register 075 | ISELR075 | 8 | 8 | 2 ICLK |
| 0008 714Ch | ICU | Interrupt request destination setting register 076 | ISELR076 | 8 | 8 | 2 ICLK |
| 0008 714Dh | ICU | Interrupt request destination setting register 077 | ISELR077 | 8 | 8 | 2 ICLK |
| 0008 714Eh | ICU | Interrupt request destination setting register 078 | ISELR078 | 8 | 8 | 2 ICLK |
| 0008 714Fh | ICU | Interrupt request destination setting register 079 | ISELR079 | 8 | 8 | 2 ICLK |
| 0008 7162h | ICU | Interrupt request destination setting register 098 | ISELR098 | 8 | 8 | 2 ICLK |
| 0008 7163h | ICU | Interrupt request destination setting register 099 | ISELR099 | 8 | 8 | 2 ICLK |
| 0008 7164h | ICU | Interrupt request destination setting register 100 | ISELR100 | 8 | 8 | 2 ICLK |
| 0008 7165h | ICU | Interrupt request destination setting register 101 | ISELR101 | 8 | 8 | 2 ICLK |
| 0008 7168h | ICU | Interrupt request destination setting register 104 | ISELR104 | 8 | 8 | 2 ICLK |
| 0008 7169h | ICU | Interrupt request destination setting register 105 | ISELR105 | 8 | 8 | 2 ICLK |
| 0008 716Ah | ICU | Interrupt request destination setting register 106 | ISELR106 | 8 | 8 | 2 ICLK |
| 0008 716Bh | ICU | Interrupt request destination setting register 107 | ISELR107 | 8 | 8 | 2 ICLK |
| 0008 716Fh | ICU | Interrupt request destination setting register 111 | ISELR111 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 7170h | ICU | Interrupt request destination setting register 112 | ISELR112 | 8 | 8 | 2 ICLK |
| 0008 7175h | ICU | Interrupt request destination setting register 117 | ISELR117 | 8 | 8 | 2 ICLK |
| 0008 7176h | ICU | Interrupt request destination setting register 118 | ISELR118 | 8 | 8 | 2 ICLK |
| 0008 717Ah | ICU | Interrupt request destination setting register 122 | ISELR122 | 8 | 8 | 2 ICLK |
| 0008 717Bh | ICU | Interrupt request destination setting register 123 | ISELR123 | 8 | 8 | 2 ICLK |
| 0008 717Ch | ICU | Interrupt request destination setting register 124 | ISELR124 | 8 | 8 | 2 ICLK |
| 0008 717Dh | ICU | Interrupt request destination setting register 125 | ISELR125 | 8 | 8 | 2 ICLK |
| 0008 717Fh | ICU | Interrupt request destination setting register 127 | ISELR127 | 8 | 8 | 2 ICLK |
| 0008 7180h | ICU | Interrupt request destination setting register 128 | ISELR128 | 8 | 8 | 2 ICLK |
| 0008 7185h | ICU | Interrupt request destination setting register 133 | ISELR133 | 8 | 8 | 2 ICLK |
| 0008 7186h | ICU | Interrupt request destination setting register 134 | ISELR134 | 8 | 8 | 2 ICLK |
| 0008 718Ah | ICU | Interrupt request destination setting register 138 | ISELR138 | 8 | 8 | 2 ICLK |
| 0008 718Bh | ICU | Interrupt request destination setting register 139 | ISELR139 | 8 | 8 | 2 ICLK |
| 0008 718Ch | ICU | Interrupt request destination setting register 140 | ISELR140 | 8 | 8 | 2 ICLK |
| 0008 718Dh | ICU | Interrupt request destination setting register 141 | ISELR141 | 8 | 8 | 2 ICLK |
| 0008 7191h | ICU | Interrupt request destination setting register 145 | ISELR145 | 8 | 8 | 2 ICLK |
| 0008 7192h | ICU | Interrupt request destination setting register 146 | ISELR146 | 8 | 8 | 2 ICLK |
| 0008 7197h | ICU | Interrupt request destination setting register 151 | ISELR151 | 8 | 8 | 2 ICLK |
| 0008 7198h | ICU | Interrupt request destination setting register 152 | ISELR152 | 8 | 8 | 2 ICLK |
| 0008 719Ch | ICU | Interrupt request destination setting register 156 | ISELR156 | 8 | 8 | 2 ICLK |
| 0008 719Dh | ICU | Interrupt request destination setting register 157 | ISELR157 | 8 | 8 | 2 ICLK |
| 0008 719Eh | ICU | Interrupt request destination setting register 158 | ISELR158 | 8 | 8 | 2 ICLK |
| 0008 719Fh | ICU | Interrupt request destination setting register 159 | ISELR159 | 8 | 8 | 2 ICLK |
| 0008 71A1h | ICU | Interrupt request destination setting register 161 | ISELR161 | 8 | 8 | 2 ICLK |
| 0008 71A2h | ICU | Interrupt request destination setting register 162 | ISELR162 | 8 | 8 | 2 ICLK |
| 0008 71A7h | ICU | Interrupt request destination setting register 167 | ISELR167 | 8 | 8 | 2 ICLK |
| 0008 71A8h | ICU | Interrupt request destination setting register 168 | ISELR168 | 8 | 8 | 2 ICLK |
| 0008 71AEh | ICU | Interrupt request destination setting register 174 | ISELR174 | 8 | 8 | 2 ICLK |
| 0008 71AFh | ICU | Interrupt request destination setting register 175 | ISELR175 | 8 | 8 | 2 ICLK |
| 0008 71B1h | ICU | Interrupt request destination setting register 177 | ISELR177 | 8 | 8 | 2 ICLK |
| 0008 71B2h | ICU | Interrupt request destination setting register 178 | ISELR178 | 8 | 8 | 2 ICLK |
| 0008 71B4h | ICU | Interrupt request destination setting register 180 | ISELR180 | 8 | 8 | 2 ICLK |
| 0008 71B5h | ICU | Interrupt request destination setting register 181 | ISELR181 | 8 | 8 | 2 ICLK |
| 0008 71B7h | ICU | Interrupt request destination setting register 183 | ISELR183 | 8 | 8 | 2 ICLK |
| 0008 71B8h | ICU | Interrupt request destination setting register 184 | ISELR184 | 8 | 8 | 2 ICLK |
| 0008 71C6h | ICU | Interrupt request destination setting register 198 | ISELR198 | 8 | 8 | 2 ICLK |
| 0008 71C7h | ICU | Interrupt request destination setting register 199 | ISELR199 | 8 | 8 | 2 ICLK |
| 0008 71C8h | ICU | Interrupt request destination setting register 200 | ISELR200 | 8 | 8 | 2 ICLK |
| 0008 71C9h | ICU | Interrupt request destination setting register 201 | ISELR201 | 8 | 8 | 2 ICLK |
| 0008 71D7h | ICU | Interrupt request destination setting register 215 | ISELR215 | 8 | 8 | 2 ICLK |
| 0008 71D8h | ICU | Interrupt request destination setting register 216 | ISELR216 | 8 | 8 | 2 ICLK |
| 0008 71DBh | ICU | Interrupt request destination setting register 219 | ISELR219 | 8 | 8 | 2 ICLK |
| 0008 71DCh | ICU | Interrupt request destination setting register 220 | ISELR220 | 8 | 8 | 2 ICLK |
| 0008 71DFh | ICU | Interrupt request destination setting register 223 | ISELR223 | 8 | 8 | 2 ICLK |
| 0008 71E0h | ICU | Interrupt request destination setting register 224 | ISELR224 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 71E3h | ICU | Interrupt request destination setting register 227 | ISELR227 | 8 | 8 | 2 ICLK |
| 0008 71E4h | ICU | Interrupt request destination setting register 228 | ISELR228 | 8 | 8 | 2 ICLK |
| 0008 71E7h | ICU | Interrupt request destination setting register 231 | ISELR231 | 8 | 8 | 2 ICLK |
| 0008 71E8h | ICU | Interrupt request destination setting register 232 | ISELR232 | 8 | 8 | 2 ICLK |
| 0008 71EBh | ICU | Interrupt request destination setting register 235 | ISELR235 | 8 | 8 | 2 ICLK |
| 0008 71ECh | ICU | Interrupt request destination setting register 236 | ISELR236 | 8 | 8 | 2 ICLK |
| 0008 71EFh | ICU | Interrupt request destination setting register 239 | ISELR239 | 8 | 8 | 2 ICLK |
| 0008 71F0h | ICU | Interrupt request destination setting register 240 | ISELR240 | 8 | 8 | 2 ICLK |
| 0008 71F7h | ICU | Interrupt request destination setting register 247 | ISELR247 | 8 | 8 | 2 ICLK |
| 0008 71F8h | ICU | Interrupt request destination setting register 248 | ISELR248 | 8 | 8 | 2 ICLK |
| 0008 71FBh | ICU | Interrupt request destination setting register 251 | ISELR251 | 8 | 8 | 2 ICLK |
| 0008 71FCh | ICU | Interrupt request destination setting register 252 | ISELR252 | 8 | 8 | 2 ICLK |
| 0008 71FDh | ICU | Interrupt request destination setting register 253 | ISELR253 | 8 | 8 | 2ICLK |
| 0008 7202h | ICU | Interrupt request enable register 02 | IER02 | 8 | 8 | 2 ICLK |
| 0008 7203h | ICU | Interrupt request enable register 03 | IER03 | 8 | 8 | 2 ICLK |
| 0008 7208h | ICU | Interrupt request enable register 08 | IER08 | 8 | 8 | 2 ICLK |
| 0008 7209h | ICU | Interrupt request enable register 09 | IER09 | 8 | 8 | 2 ICLK |
| 0008 720Ch | ICU | Interrupt request enable register 0C | IER0C | 8 | 8 | 2 ICLK |
| 0008 720Dh | ICU | Interrupt request enable register 0D | IER0D | 8 | 8 | 2 ICLK |
| 0008 720Eh | ICU | Interrupt request enable register 0E | IER0E | 8 | 8 | 2 ICLK |
| 0008 720Fh | ICU | Interrupt request enable register 0F | IER0F | 8 | 8 | 2 ICLK |
| 0008 7210h | ICU | Interrupt request enable register 10 | IER10 | 8 | 8 | 2 ICLK |
| 0008 7211h | ICU | Interrupt request enable register 11 | IER11 | 8 | 8 | 2 ICLK |
| 0008 7212h | ICU | Interrupt request enable register 12 | IER12 | 8 | 8 | 2 ICLK |
| 0008 7213h | ICU | Interrupt request enable register 13 | IER13 | 8 | 8 | 2 ICLK |
| 0008 7214h | ICU | Interrupt request enable register 14 | IER14 | 8 | 8 | 2 ICLK |
| 0008 7215h | ICU | Interrupt request enable register 15 | IER15 | 8 | 8 | 2 ICLK |
| 0008 7216h | ICU | Interrupt request enable register 16 | IER16 | 8 | 8 | 2 ICLK |
| 0008 7217h | ICU | Interrupt request enable register 17 | IER17 | 8 | 8 | 2 ICLK |
| 0008 7218h | ICU | Interrupt request enable register 18 | IER18 | 8 | 8 | 2 ICLK |
| 0008 7219h | ICU | Interrupt request enable register 19 | IER19 | 8 | 8 | 2 ICLK |
| 0008 721Ah | ICU | Interrupt request enable register 1A | IER1A | 8 | 8 | 2 ICLK |
| 0008 721Bh | ICU | Interrupt request enable register 1B | IER1B | 8 | 8 | 2 ICLK |
| 0008 721Ch | ICU | Interrupt request enable register 1C | IER1C | 8 | 8 | 2 ICLK |
| 0008 721Dh | ICU | Interrupt request enable register 1D | IER1D | 8 | 8 | 2 ICLK |
| 0008 721Eh | ICU | Interrupt request enable register 1E | IER1E | 8 | 8 | 2 ICLK |
| 0008 721Fh | ICU | Interrupt request enable register 1F | IER1F | 8 | 8 | 2 ICLK |
| 0008 7300h | ICU | Interrupt priority register 00 | IPR00 | 8 | 8 | 2 ICLK |
| 0008 7301h | ICU | Interrupt priority register 01 | IPR01 | 8 | 8 | 2 ICLK |
| 0008 7302h | ICU | Interrupt priority register 02 | IPR02 | 8 | 8 | 2 ICLK |
| 0008 7304h | ICU | Interrupt priority register 04 | IPR04 | 8 | 8 | 2 ICLK |
| 0008 7305h | ICU | Interrupt priority register 05 | IPR05 | 8 | 8 | 2 ICLK |
| 0008 7306h | ICU | Interrupt priority register 06 | IPR06 | 8 | 8 | 2 ICLK |
| 0008 7307h | ICU | Interrupt priority register 07 | IPR07 | 8 | 8 | 2 ICLK |
| 0008 7320h | ICU | Interrupt priority register 20 | IPR20 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 7321h | ICU | Interrupt priority register 21 | IPR21 | 8 | 8 | 2 ICLK |
| 0008 7322h | ICU | Interrupt priority register 22 | IPR22 | 8 | 8 | 2 ICLK |
| 0008 7323h | ICU | Interrupt priority register 23 | IPR23 | 8 | 8 | 2 ICLK |
| 0008 7324h | ICU | Interrupt priority register 24 | IPR24 | 8 | 8 | 2 ICLK |
| 0008 7325h | ICU | Interrupt priority register 25 | IPR25 | 8 | 8 | 2 ICLK |
| 0008 7326h | ICU | Interrupt priority register 26 | IPR26 | 8 | 8 | 2 ICLK |
| 0008 7327h | ICU | Interrupt priority register 27 | IPR27 | 8 | 8 | 2 ICLK |
| 0008 7328h | ICU | Interrupt priority register 28 | IPR28 | 8 | 8 | 2 ICLK |
| 0008 7329h | ICU | Interrupt priority register 29 | IPR29 | 8 | 8 | 2 ICLK |
| 0008 732Ah | ICU | Interrupt priority register 2A | IPR2A | 8 | 8 | 2 ICLK |
| 0008 732Bh | ICU | Interrupt priority register 2B | IPR2B | 8 | 8 | 2 ICLK |
| 0008 732Ch | ICU | Interrupt priority register 2C | IPR2C | 8 | 8 | 2 ICLK |
| 0008 732Dh | ICU | Interrupt priority register 2D | IPR2D | 8 | 8 | 2 ICLK |
| 0008 732Eh | ICU | Interrupt priority register 2E | IPR2E | 8 | 8 | 2 ICLK |
| 0008 732Fh | ICU | Interrupt priority register 2F | IPR2F | 8 | 8 | 2 ICLK |
| 0008 7340h | ICU | Interrupt priority register 40 | IPR40 | 8 | 8 | 2 ICLK |
| 0008 7344h | ICU | Interrupt priority register 44 | IPR44 | 8 | 8 | 2 ICLK |
| 0008 7345h | ICU | Interrupt priority register 45 | IPR45 | 8 | 8 | 2 ICLK |
| 0008 7346h | ICU | Interrupt priority register 46 | IPR46 | 8 | 8 | 2 ICLK |
| 0008 7347h | ICU | Interrupt priority register 47 | IPR47 | 8 | 8 | 2 ICLK |
| 0008 734Ch | ICU | Interrupt priority register 4C | IPR4C | 8 | 8 | 2 ICLK |
| 0008 734Dh | ICU | Interrupt priority register 4D | IPR4D | 8 | 8 | 2 ICLK |
| 0008 734Eh | ICU | Interrupt priority register 4E | IPR4E | 8 | 8 | 2 ICLK |
| 0008 734Fh | ICU | Interrupt priority register 4F | IPR4F | 8 | 8 | 2 ICLK |
| 0008 7350h | ICU | Interrupt priority register 50 | IPR50 | 8 | 8 | 2 ICLK |
| 0008 7351h | ICU | Interrupt priority register 51 | IPR51 | 8 | 8 | 2 ICLK |
| 0008 7352h | ICU | Interrupt priority register 52 | IPR52 | 8 | 8 | 2 ICLK |
| 0008 7353h | ICU | Interrupt priority register 53 | IPR53 | 8 | 8 | 2 ICLK |
| 0008 7354h | ICU | Interrupt priority register 54 | IPR54 | 8 | 8 | 2 ICLK |
| 0008 7355h | ICU | Interrupt priority register 55 | IPR55 | 8 | 8 | 2 ICLK |
| 0008 7356h | ICU | Interrupt priority register 56 | IPR56 | 8 | 8 | 2 ICLK |
| 0008 7357h | ICU | Interrupt priority register 57 | IPR57 | 8 | 8 | 2 ICLK |
| 0008 7358h | ICU | Interrupt priority register 58 | IPR58 | 8 | 8 | 2 ICLK |
| 0008 7359h | ICU | Interrupt priority register 59 | IPR59 | 8 | 8 | 2 ICLK |
| 0008 735Ah | ICU | Interrupt priority register 5A | IPR5A | 8 | 8 | 2 ICLK |
| 0008 735Bh | ICU | Interrupt priority register 5B | IPR5B | 8 | 8 | 2 ICLK |
| 0008 735Ch | ICU | Interrupt priority register 5C | IPR5C | 8 | 8 | 2 ICLK |
| 0008 735Dh | ICU | Interrupt priority register 5D | IPR5D | 8 | 8 | 2 ICLK |
| 0008 735Eh | ICU | Interrupt priority register 5E | IPR5E | 8 | 8 | 2 ICLK |
| 0008 735Fh | ICU | Interrupt priority register 5F | IPR5F | 8 | 8 | 2 ICLK |
| 0008 7360h | ICU | Interrupt priority register 60 | IPR60 | 8 | 8 | 2 ICLK |
| 0008 7361h | ICU | Interrupt priority register 61 | IPR61 | 8 | 8 | 2 ICLK |
| 0008 7362h | ICU | Interrupt priority register 62 | IPR62 | 8 | 8 | 2 ICLK |
| 0008 7363h | ICU | Interrupt priority register 63 | IPR63 | 8 | 8 | 2 ICLK |
| 0008 7368h | ICU | Interrupt priority register 68 | IPR68 | 8 | 8 | 2 ICLK |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 7369h | ICU | Interrupt priority register 69 | IPR69 | 8 | 8 | 2 ICLK |
| 0008 736Ah | ICU | Interrupt priority register 6A | IPR6A | 8 | 8 | 2 ICLK |
| 0008 736Bh | ICU | Interrupt priority register 6B | IPR6B | 8 | 8 | 2 ICLK |
| 0008 7370h | ICU | Interrupt priority register 70 | IPR70 | 8 | 8 | 2 ICLK |
| 0008 7371h | ICU | Interrupt priority register 71 | IPR71 | 8 | 8 | 2 ICLK |
| 0008 7372h | ICU | Interrupt priority register 72 | IPR72 | 8 | 8 | 2 ICLK |
| 0008 7373h | ICU | Interrupt priority register 73 | IPR73 | 8 | 8 | 2 ICLK |
| 0008 7380h | ICU | Interrupt priority register 80 | IPR80 | 8 | 8 | 2 ICLK |
| 0008 7381h | ICU | Interrupt priority register 81 | IPR81 | 8 | 8 | 2 ICLK |
| 0008 7382h | ICU | Interrupt priority register 82 | IPR82 | 8 | 8 | 2 ICLK |
| 0008 7383h | ICU | Interrupt priority register 83 | IPR83 | 8 | 8 | 2 ICLK |
| 0008 7384h | ICU | Interrupt priority register 84 | IPR84 | 8 | 8 | 2 ICLK |
| 0008 7385h | ICU | Interrupt priority register 85 | IPR85 | 8 | 8 | 2 ICLK |
| 0008 7386h | ICU | Interrupt priority register 86 | IPR86 | 8 | 8 | 2 ICLK |
| 0008 7388h | ICU | Interrupt priority register 88 | IPR88 | 8 | 8 | 2 ICLK |
| 0008 7389h | ICU | Interrupt priority register 89 | IPR89 | 8 | 8 | 2 ICLK |
| 0008 738Ah | ICU | Interrupt priority register 8A | IPR8A | 8 | 8 | 2 ICLK |
| 0008 738Bh | ICU | Interrupt priority register 8B | IPR8B | 8 | 8 | 2 ICLK |
| 0008 738Ch | ICU | Interrupt priority register 8C | IPR8C | 8 | 8 | 2 ICLK |
| 0008 738Dh | ICU | Interrupt priority register 8D | IPR8D | 8 | 8 | 2 ICLK |
| 0008 738Eh | ICU | Interrupt priority register 8E | IPR8E | 8 | 8 | 2 ICLK |
| 0008 738Fh | ICU | Interrupt priority register 8F | IPR8F | 8 | 8 | 2 ICLK |
| 0008 73F0h | ICU | Fast interrupt register | FIR | 16 | 16 | 2 ICLK |
| 0008 7400h | DTC | DTC control register | DTCCR | 8 | 8 | 2 ICLK |
| 0008 7404h | DTC | DTC vector base register | DTCVBR | 32 | 32 | 2 ICLK |
| 0008 7408h | DTC | DTC address mode register | DTCADMOD | 8 | 8 | 2 ICLK |
| 0008 740Ch | DTC | DTC module start register | DTCST | 8 | 8 | 2 ICLK |
| 0008 8000h | CMT (unit 0) | Compare match timer start register 0 | CMSTR0 | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8002h | CMT0 | Compare match timer control register | CMCR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8004h | CMT0 | Compare match timer counter | CMCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8006h | CMT0 | Compare match timer constant register | CMCOR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8008h | CMT1 | Compare match timer control register | CMCR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 800Ah | CMT1 | Compare match timer counter | CMCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 800Ch | CMT1 | Compare match timer constant register | CMCOR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8010h | CMT (unit 1) | Compare match timer start register 1 | CMSTR1 | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8012h | CMT2 | Compare match timer control register | CMCR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8014h | CMT2 | Compare match timer counter | CMCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8016h | CMT2 | Compare match timer constant register | CMCOR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8018h | CMT3 | Compare match timer control register | CMCR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 801Ah | CMT3 | Compare match timer counter | CMCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 801Ch | CMT3 | Compare match timer constant register | CMCOR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8028h | WDT | Timer control/status register | TCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8028h | WDT | Write window A register | WINA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8029h | WDT | Timer counter | TCNT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 802Ah | WDT | Write window B register | WINB | 16 | 16 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 802Bh | WDT | Reset control/status register | RSTCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8040h | AD0 | A/D data register A | ADDRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8042h | AD0 | A/D data register B | ADDRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8044h | AD0 | A/D data register C | ADDRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8046h | AD0 | A/D data register D | ADDRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8050h | AD0 | A/D control/status register | ADCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8051h | AD0 | A/D control register | ADCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8052h | AD0 | ADDRy format select register | ADDPR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8053h | AD0 | A/D sampling state register | ADSSTR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8060h | AD1 | A/D data register A | ADDRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8062h | AD1 | A/D data register B | ADDRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8064h | AD1 | A/D data register C | ADDRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8066h | AD1 | A/D data register D | ADDRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8070h | AD1 | A/D control/status register | ADCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8071h | AD1 | A/D control register | ADCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8072h | AD1 | ADDRy format select register | ADDPR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8073h | AD1 | A/D sampling state register | ADSSTR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8080h | AD2 | A/D data register A | ADDRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8082h | AD2 | A/D data register B | ADDRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8084h | AD2 | A/D data register C | ADDRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8086h | AD2 | A/D data register D | ADDRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8090h | AD2 | A/D control/status register | ADCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8091h | AD2 | A/D control register | ADCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8092h | AD2 | ADDRy format select register | ADDPR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8093h | AD2 | A/D sampling state register | ADSSTR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80A0h | AD3 | A/D data register A | ADDRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80A2h | AD3 | A/D data register B | ADDRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80A4h | AD3 | A/D data register C | ADDRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80A6h | AD3 | A/D data register D | ADDRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80B0h | AD3 | A/D control/status register | ADCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80B1h | AD3 | A/D control register | ADCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80B2h | AD3 | ADDRy format select register | ADDPR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80B3h | AD3 | A/D sampling state register | ADSSTR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80C0h | D/A | D/A data register 0 | DADR0 | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80C2h | D/A | D/A data register 1 | DADR1 | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 80C4h | D/A | D/A control register | DACR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 80C5h | D/A | DADRy format select register | DADPR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8100h | TPU (unit 0) | Timer start register | TSTRA | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8101h | TPU (unit 0) | Timer synchronous register | TSYRA | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8110h | TPU0 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8111h | TPU0 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8112h | TPU0 | Timer I/O control register H | TIORH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8113h | TPU0 | Timer I/O control register L | TIORL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8114h | TPU0 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8115h | TPU0 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 8116h | TPU0 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8118h | TPU0 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 811Ah | TPU0 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 811Ch | TPU0 | Timer general register C | TGRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 811Eh | TPU0 | Timer general register D | TGRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8120h | TPU1 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8121h | TPU1 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8122h | TPU1 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8124h | TPU1 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8125h | TPU1 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8126h | TPU1 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8128h | TPU1 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 812Ah | TPU1 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8130h | TPU2 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8131h | TPU2 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8132h | TPU2 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8134h | TPU2 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8135h | TPU2 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8136h | TPU2 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8138h | TPU2 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 813Ah | TPU2 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8140h | TPU3 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8141h | TPU3 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8142h | TPU3 | Timer I/O control register H | TIORH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8143h | TPU3 | Timer I/O control register L | TIORL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8144h | TPU3 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8145h | TPU3 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8146h | TPU3 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8148h | TPU3 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 814Ah | TPU3 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 814Ch | TPU3 | Timer general register C | TGRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 814Eh | TPU3 | Timer general register D | TGRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8150h | TPU4 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8151h | TPU4 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8152h | TPU4 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8154h | TPU4 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8155h | TPU4 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8156h | TPU4 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8158h | TPU4 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 815Ah | TPU4 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8160h | TPU5 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8161h | TPU5 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8162h | TPU5 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8164h | TPU5 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8165h | TPU5 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 8166h | TPU5 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8168h | TPU5 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 816Ah | TPU5 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8170h | TPU (unit 1) | Timer start register | TSTRB | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8171h | TPU (unit 1) | Timer synchronous register | TSYRB | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8180h | TPU6 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8181h | TPU6 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8182h | TPU6 | Timer I/O control register H | TIORH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8183h | TPU6 | Timer I/O control register L | TIORL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8184h | TPU6 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8185h | TPU6 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8186h | TPU6 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8188h | TPU6 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 818Ah | TPU6 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 818Ch | TPU6 | Timer general register C | TGRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 818Eh | TPU6 | Timer general register D | TGRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8190h | TPU7 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8191h | TPU7 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8192h | TPU7 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8194h | TPU7 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8195h | TPU7 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8196h | TPU7 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8198h | TPU7 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 819Ah | TPU7 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81A0h | TPU8 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81A1h | TPU8 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81A2h | TPU8 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81A4h | TPU8 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81A5h | TPU8 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81A6h | TPU8 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81A8h | TPU8 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81AAh | TPU8 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81B0h | TPU9 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B1h | TPU9 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B2h | TPU9 | Timer I/O control register H | TIORH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B3h | TPU9 | Timer I/O control register L | TIORL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B4h | TPU9 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B5h | TPU9 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81B6h | TPU9 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81B8h | TPU9 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81BAh | TPU9 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81BCh | TPU9 | Timer general register C | TGRC | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81BEh | TPU9 | Timer general register D | TGRD | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81C0h | TPU10 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81C1h | TPU10 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 81C2h | TPU10 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81C4h | TPU10 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81C5h | TPU10 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81C6h | TPU10 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81C8h | TPU10 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81CAh | TPU10 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81D0h | TPU11 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81D1h | TPU11 | Timer mode register | TMDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81D2h | TPU11 | Timer I/O control register | TIOR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81D4h | TPU11 | Timer interrupt enable register | TIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81D5h | TPU11 | Timer status register | TSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81D6h | TPU11 | Timer counter | TCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81D8h | TPU11 | Timer general register A | TGRA | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81DAh | TPU11 | Timer general register B | TGRB | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 81E6h | PPG0 | PPG output control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81E7h | PPG0 | PPG output mode register | PMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81E8h | PPG0 | Next data enable register H | NDERH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81E9h | PPG0 | Next data enable register L | NDERL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81EAh | PPG0 | Output data register H | PODRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81EBh | PPG0 | Output data register L | PODRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81Ech*1 | PPG0 | Next data register H | NDRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81EDh*2 | PPG0 | Next data register L | NDRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81EEh*1 | PPG0 | Next data register H | NDRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81EFh*2 | PPG0 | Next data register L | NDRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81F0h | PPG1 | PPG trigger select register | PTRSLR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81F6h | PPG1 | PPG output control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81F7h | PPG1 | PPG output mode register | PMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81F8h | PPG1 | Next data enable register H | NDERH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81F9h | PPG1 | Next data enable register L | NDERL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FAh | PPG1 | Output data register H | PODRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FBh | PPG1 | Output data register L | PODRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FCh*3 | PPG1 | Next data register H | NDRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FDh*4 | PPG1 | Next data register L | NDRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FEh*3 | PPG1 | Next data register H | NDRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 81FFh*4 | PPG1 | Next data register L | NDRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8200h | TMR0 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8201h | TMR1 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8202h | TMR0 | Timer control/status register | TCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8203h | TMR1 | Timer control/status register | TCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8204h | TMR0 | Time constant register A | TCORA | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8205h | TMR1 | Time constant register A | TCORA | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |
| 0008 8206h | TMR0 | Time constant register B | TCORB | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8207h | TMR1 | Time constant register B | TCORB | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |
| 0008 8208h | TMR0 | Timer counter | TCNT | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8209h | TMR1 | Timer counter | TCNT | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 820Ah | TMR0 | Timer counter control register | TCCR | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 820Bh | TMR1 | Timer counter control register | TCCR | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8210h | TMR2 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8211h | TMR3 | Timer control register | TCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8212h | TMR2 | Timer control/status register | TCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8213h | TMR3 | Timer control/status register | TCSR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8214h | TMR2 | Time constant register A | TCORA | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8215h | TMR3 | Time constant register A | TCORA | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |
| 0008 8216h | TMR2 | Time constant register B | TCORB | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8217h | TMR3 | Time constant register B | TCORB | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |
| 0008 8218h | TMR2 | Timer counter | TCNT | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8219h | TMR3 | Timer counter | TCNT | 8 | 8 or 16*5 | 2 to 3 PCLK *7 |
| 0008 821Ah | TMR2 | Timer counter control register | TCCR | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 821Bh | TMR3 | Timer counter control register | TCCR | 8 | 8 or 16 | 2 to 3 PCLK *7 |
| 0008 8240h | SCI0 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8241h | SCI0 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8242h | SCI0 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8243h | SCI0 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8244h | SCI0 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8245h | SCI0 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8246h | SCI0 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8247h | SCI0 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8248h | SCI1 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8249h | SCI1 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Ah | SCI1 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Bh | SCI1 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Ch | SCI1 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Dh | SCI1 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Eh | SCI1 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 824Fh | SCI1 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8250h | SCI2 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8251h | SCI2 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8252h | SCI2 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8253h | SCI2 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8254h | SCI2 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8255h | SCI2 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8256h | SCI2 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8257h | SCI2 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8258h | SCI3 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8259h | SCI3 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 825Ah | SCI3 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 825Bh | SCI3 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 825Ch | SCI3 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 825Dh | SCI3 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 825Eh | SCI3 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 825Fh | SCI3 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8260h | SCI4 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8261h | SCI4 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8262h | SCI4 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8263h | SCI4 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8264h | SCI4 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8265h | SCI4 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8266h | SCI4 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8267h | SCI4 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8268h | SCI5 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8269h | SCI5 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Ah | SCI5 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Bh | SCI5 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Ch | SCI5 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Dh | SCI5 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Eh | SCI5 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 826Fh | SCI5 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8270h | SCI6 | Serial mode register | SMR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8271h | SCI6 | Bit rate register | BRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8272h | SCI6 | Serial control register | SCR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8273h | SCI6 | Transmit data register | TDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8274h | SCI6 | Serial status register | SSR*6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8275h | SCI6 | Receive data register | RDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8276h | SCI6 | Smart card mode register | SCMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8277h | SCI6 | Serial extended mode register | SEMR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8280h | CRC | CRC control register | CRCCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8281h | CRC | CRC data input register | CRCDIR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8282h | CRC | CRC data output register | CRCDOR | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 8300h | RIIC0 | I2C bus control register 1 | ICCR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8301h | RIIC0 | I2C bus control register 2 | ICCR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8302h | RIIC0 | I2C bus mode register 1 | ICMR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8303h | RIIC0 | I2C bus mode register 2 | ICMR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8304h | RIIC0 | I2C bus mode register 3 | ICMR3 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8305h | RIIC0 | I2C bus function enable register | ICFER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8306h | RIIC0 | I2C bus status enable register | ICSER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8307h | RIIC0 | I2C bus interrupt enable register | ICIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8308h | RIIC0 | I2C bus status register 1 | ICSR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8309h | RIIC0 | I2C bus status register 2 | ICSR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 830Ah | RIIC0 | Slave address register L0 | SARL0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 830Ah | RIIC0 | Internal control for timeout L | TMOCNTL | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 830Bh | RIIC0 | Slave address register U0 | SARU0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 830Bh | RIIC0 | Internal control for timeout U | TMOCNTU | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 830Ch | RIIC0 | Slave address register L1 | SARL1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 830Dh | RIIC0 | Slave address register U1 | SARU1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 830Eh | RIIC0 | Slave address register L2 | SARL2 | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 830Fh | RIIC0 | Slave address register U2 | SARU2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8310h | RIIC0 | I2C bus bit rate low-level register | ICBRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8311h | RIIC0 | I2C bus bit rate high-level register | ICBRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8312h | RIIC0 | I2C bus transmit data register | ICDRT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8313h | RIIC0 | I2C bus receive data register | ICDRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8320h | RIIC1 | I2C bus control register 1 | ICCR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8321h | RIIC1 | I2C bus control register 2 | ICCR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8322h | RIIC1 | I2C bus mode register 1 | ICMR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8323h | RIIC1 | I2C bus mode register 2 | ICMR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8324h | RIIC1 | I2C bus mode register 3 | ICMR3 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8325h | RIIC1 | I2C bus function enable register | ICFER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8326h | RIIC1 | I2C bus status enable register | ICSER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8327h | RIIC1 | I2C bus interrupt enable register | ICIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8328h | RIIC1 | I2C bus status register 1 | ICSR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8329h | RIIC1 | I2C bus status register 2 | ICSR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Ah | RIIC1 | Slave address register L0 | SARL0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Ah | RIIC1 | Internal control for timeout L | TMOCNTL | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 832Bh | RIIC1 | Slave address register U0 | SARU0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Bh | RIIC1 | Internal control for timeout U | TMOCNTU | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 832Ch | RIIC1 | Slave address register L1 | SARL1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Dh | RIIC1 | Slave address register U1 | SARU1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Eh | RIIC1 | Slave address register L2 | SARL2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 832Fh | RIIC1 | Slave address register U2 | SARU2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8330h | RIIC1 | I2C bus bit rate low-level register | ICBRL | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8331h | RIIC1 | I2C bus bit rate high-level register | ICBRH | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8332h | RIIC1 | I2C bus transmit data register | ICDRT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 8333h | RIIC1 | I2C bus receive data register | ICDRR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C000h | P0 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C001h | P1 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C002h | P2 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C003h | P3 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C004h | P4 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C005h | P5 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C006h | P6 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C007h | P7 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C008h | P8 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C009h | P9 | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Ah | PA | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Bh | PB | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Ch | PC | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Dh | PD | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Eh | PE | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C00Fh | PF | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C010h | PG | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C011h | PH | Data direction register | DDR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 C020h | P0 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C021h | P1 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C022h | P2 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C023h | P3 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C024h | P4 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C025h | P5 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C026h | P6 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C027h | P7 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C028h | P8 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C029h | P9 | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Ah | PA | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Bh | PB | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Ch | PC | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Dh | PD | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Eh | PE | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C02Fh | PF | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C030h | PG | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C031h | PH | Data register | DR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C040h | P0 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C041h | P1 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C042h | P2 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C043h | P3 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C044h | P4 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C045h | P5 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C046h | P6 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C047h | P7 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C048h | P8 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C049h | P9 | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Ah | PA | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Bh | PB | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Ch | PC | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Dh | PD | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Eh | PE | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C04Fh | PF | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C050h | PG | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C051h | PH | Port register | PORT | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C060h | P0 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C061h | P1 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C062h | P2 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C063h | P3 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C064h | P4 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C065h | P5 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C066h | P6 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C067h | P7 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C068h | P8 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 C069h | P9 | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Ah | PA | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Bh | PB | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Ch | PC | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Dh | PD | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Eh | PE | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C06Fh | PF | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C070h | PG | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C071h | PH | Input buffer control register | ICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C082h | P2 | Open drain control register | ODR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C08Ch | PC | Open drain control register | ODR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C0CAh | PA | Pull-Up resistor control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C0CBh | PB | Pull-Up resistor control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C0CCh | PC | Pull-Up resistor control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C0CDh | PD | Pull-Up resistor control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C0CEh | PE | Pull-Up resistor control register | PCR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C100h | I/O PORT | Port function control register 0 | PFCR0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C101h | I/O PORT | Port function control register 1 | PFCR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C102h | I/O PORT | Port function control register 2 | PFCR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C103h | I/O PORT | Port function control register 3 | PFCR3 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C104h | I/O PORT | Port function control register 4 | PFCR4 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C105h | I/O PORT | Port function control register 5 | PFCR5 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C106h | I/O PORT | Port function control register 6 | PFCR6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C107h | I/O PORT | Port function control register 7 | PFCR7 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C108h | I/O PORT | Port function control register 8 | PFCR8 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C109h | I/O PORT | Port function control register 9 | PFCR9 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C280h | SYSTEM | Deep standby control register | DPSBYCR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C281h | SYSTEM | Deep standby wait control register | DPSWCR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C282h | SYSTEM | Deep standby interrupt enable register | DPSIER | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C283h | SYSTEM | Deep standby interrupt flag register | DPSIFR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C284h | SYSTEM | Deep standby interrupt edge register | DPSIEGR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C285h | SYSTEM | Reset status register | RSTSR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C289h | FLASH | Flash write erase protection register | FWEPROR | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C290h | SYSTEM | Deep standby backup register 0 | DPSBKR0 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C291h | SYSTEM | Deep standby backup register 1 | DPSBKR1 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C292h | SYSTEM | Deep standby backup register 2 | DPSBKR2 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C293h | SYSTEM | Deep standby backup register 3 | DPSBKR3 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C294h | SYSTEM | Deep standby backup register 4 | DPSBKR4 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C295h | SYSTEM | Deep standby backup register 5 | DPSBKR5 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C296h | SYSTEM | Deep standby backup register 6 | DPSBKR6 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C297h | SYSTEM | Deep standby backup register 7 | DPSBKR7 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C298h | SYSTEM | Deep standby backup register 8 | DPSBKR8 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C299h | SYSTEM | Deep standby backup register 9 | DPSBKR9 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C29Ah | SYSTEM | Deep standby backup register 10 | DPSBKR10 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C29Bh | SYSTEM | Deep standby backup register 11 | DPSBKR11 | 8 | 8 | 4 to 5 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 C29Ch | SYSTEM | Deep standby backup register 12 | DPSBKR12 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C29Dh | SYSTEM | Deep standby backup register 13 | DPSBKR13 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C29Eh | SYSTEM | Deep standby backup register 14 | DPSBKR14 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C29Fh | SYSTEM | Deep standby backup register 15 | DPSBKR15 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A0h | SYSTEM | Deep standby backup register 16 | DPSBKR16 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A1h | SYSTEM | Deep standby backup register 17 | DPSBKR17 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A2h | SYSTEM | Deep standby backup register 18 | DPSBKR18 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A3h | SYSTEM | Deep standby backup register 19 | DPSBKR19 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A4h | SYSTEM | Deep standby backup register 20 | DPSBKR20 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A5h | SYSTEM | Deep standby backup register 21 | DPSBKR21 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A6h | SYSTEM | Deep standby backup register 22 | DPSBKR22 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A7h | SYSTEM | Deep standby backup register 23 | DPSBKR23 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A8h | SYSTEM | Deep standby backup register 24 | DPSBKR24 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2A9h | SYSTEM | Deep standby backup register 25 | DPSBKR25 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2AAh | SYSTEM | Deep standby backup register 26 | DPSBKR26 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2ABh | SYSTEM | Deep standby backup register 27 | DPSBKR27 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2ACh | SYSTEM | Deep standby backup register 28 | DPSBKR28 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2ADh | SYSTEM | Deep standby backup register 29 | DPSBKR29 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2AEh | SYSTEM | Deep standby backup register 30 | DPSBKR30 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C2AFh | SYSTEM | Deep standby backup register 31 | DPSBKR31 | 8 | 8 | 4 to 5 PCLK *7 |
| 0008 C300h | ICU | IRQ detection enable registrar 0 | IRQER0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C301h | ICU | IRQ detection enable registrar 1 | IRQER1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C302h | ICU | IRQ detection enable registrar 2 | IRQER2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C303h | ICU | IRQ detection enable registrar 3 | IRQER3 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C304h | ICU | IRQ detection enable registrar 4 | IRQER4 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C305h | ICU | IRQ detection enable registrar 5 | IRQER5 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C306h | ICU | IRQ detection enable registrar 6 | IRQER6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C307h | ICU | IRQ detection enable registrar 7 | IRQER7 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C308h | ICU | IRQ detection enable registrar 8 | IRQER8 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C309h | ICU | IRQ detection enable registrar 9 | IRQER9 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Ah | ICU | IRQ detection enable registrar 10 | IRQER10 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Bh | ICU | IRQ detection enable registrar 11 | IRQER11 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Ch | ICU | IRQ detection enable registrar 12 | IRQER12 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Dh | ICU | IRQ detection enable registrar 13 | IRQER13 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Eh | ICU | IRQ detection enable registrar 14 | IRQER14 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C30Fh | ICU | IRQ detection enable registrar 15 | IRQER15 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C320h | ICU | IRQ control register 0 | IRQCR0 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C321h | ICU | IRQ control register 1 | IRQCR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C322h | ICU | IRQ control register 2 | IRQCR2 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C323h | ICU | IRQ control register 3 | IRQCR3 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C324h | ICU | IRQ control register 4 | IRQCR4 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C325h | ICU | IRQ control register 5 | IRQCR5 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C326h | ICU | IRQ control register 6 | IRQCR6 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C327h | ICU | IRQ control register 7 | IRQCR7 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C328h | ICU | IRQ control register 8 | IRQCR8 | 8 | 8 | 2 to 3 PCLK *7 |



Downloaded from Arrow.com.

| Address | Module Abbreviation | Register Name | Register Abbreviation | Number of Bits | Access Size | Number of Access Cycles |
|---|---|---|---|---|---|---|
| 0008 C329h | ICU | IRQ control register 9 | IRQCR9 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Ah | ICU | IRQ control register 10 | IRQCR10 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Bh | ICU | IRQ control register 11 | IRQCR11 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Ch | ICU | IRQ control register 12 | IRQCR12 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Dh | ICU | IRQ control register 13 | IRQCR13 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Eh | ICU | IRQ control register 14 | IRQCR14 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C32Fh | ICU | IRQ control register 15 | IRQCR15 | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C340h | ICU | Software standby release IRQ enable register | SSIER | 16 | 16 | 2 to 3 PCLK *7 |
| 0008 C350h | ICU | Non-maskable interrupt enable register | NMIER | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C351h | ICU | NMI pin interrupt control register | NMICR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C352h | ICU | Non-maskable interrupt status register | NMISR | 8 | 8 | 2 to 3 PCLK *7 |
| 0008 C353h | ICU | Non-maskable interrupt clear register | NMICLR | 8 | 8 | 2 to 3 PCLK *7 |
| 007F C402h | FLASH | Flash mode register | FMODR | 8 | 8 | 2 to 3 PCLK *7 |
| 007F C410h | FLASH | Flash access status register | FASTAT | 8 | 8 | 2 to 3 PCLK *7 |
| 007F C411h | FLASH | Flash access error interrupt enable register | FAEINT | 8 | 8 | 2 to 3 PCLK *7 |
| 007F C412h | FLASH | Flash ready interrupt enable register | FRDYIE | 8 | 8 | 2 to 3 PCLK *7 |
| 007F C440h | FLASH | Data flash read enable register | DFLRE | 16 | 16 | 2 to 3 PCLK *7 |
| 007F C450h | FLASH | Data flash programming/erasure enable register | DFLWE | 16 | 16 | 2 to 3 PCLK *7 |
| 007F C454h | FLASH | FCU RAM enable register | FCURAME | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFB0h | FLASH | Flash status register 0 | FSTATR0 | 8 | 8 | 2 to 3 PCLK *7 |
| 007F FFB1h | FLASH | Flash status register 1 | FSTATR1 | 8 | 8 | 2 to 3 PCLK *7 |
| 007F FFB2h | FLASH | Flash P/E mode entry register | FENTRYR | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFB4h | FLASH | Flash protection register | FPROTR | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFB6h | FLASH | Flash reset register | FRESETR | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFBAh | FLASH | FCU command register | FCMDR | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFC8h | FLASH | FCU processing switching register | FCPSR | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFCAh | FLASH | Data flash blank check control register | DFLBCCNT | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFCCh | FLASH | Flash P/E status register | FPESTAT | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFCEh | FLASH | Data flash blank check status register | DFLBCSTAT | 16 | 16 | 2 to 3 PCLK *7 |
| 007F FFE8h | FLASH | Peripheral clock notification register | PCKAR | 16 | 16 | 2 to 3 PCLK *7 |

Notes:
1. When the same output trigger is specified for pulse output groups 2 and 3 by the PPG0.PCR setting, the PPG0.NDRH address is 000881ECh. When different output triggers are specified, the PPG0.NDRH addresses for pulse output groups 2 and 3 are 000881EEh and 000881ECh, respectively.
2. When the same output trigger is specified for pulse output groups 0 and 1 by the PPG0.PCR setting, the PPG0.NDRL address is 000881EDh. When different output triggers are specified, the PPG0.NDRL addresses for pulse output groups 0 and 1 are 000881EFh and 000881EDh, respectively.
3. When the same output trigger is specified for pulse output groups 6 and 7 by the PPG1.PCR setting, the PPG1.NDRH address is 000881FCh. When different output triggers are specified, the PPG1.NDRH addresses for pulse output groups 6 and 7 are 000881FEh and 000881FCh, respectively.
4. When the same output trigger is specified for pulse output groups 4 and 5 by the PPG1.PCR setting, the PPG1.NDRL address is 000881FDh. When different output triggers are specified, the PPG1.NDRL addresses for pulse output groups 4 and 5 are 000881FFh and 000881FDh, respectively.
5. 16-bit access to odd addresses is prohibited. When 16-bit access is required, access is at the address corresponding to TMR0 or TMR2.
6. For certain bits, functions differ according to whether the mode is serial communications or smart card interface.
7. The number of access cycles varies depending on the number of divided cycles for clock synchronization (0 to one PCLK).
8. The number of access cycles may be 5 ICLK if the register is accessed during the DMAC operation.



Downloaded from Arrow.com.

## 5.2 I/O Register Bits

Register addresses and bit names of the peripheral modules are described below.
Each line cover eight bits, and 16-bit and 32-bit registers are shown as 2 or 4 lines, respectively.

**Table 5.2 List of I/O Registers (Bit Order)**

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| SYSTEM | MDMONR | — | — | — | — | — | — | — | — |
| | | MDE | — | — | — | — | — | MD1 | MD0 |
| SYSTEM | MDSR | — | — | — | — | — | — | — | — |
| | | — | UBTS | — | BOTS | BSW[1:0] | | EXB | IROM |
| SYSTEM | SYSCR0 | KEY[7:0] | | | | | | | |
| | | — | — | — | — | — | — | EXBE | ROME |
| SYSTEM | SYSCR1 | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | — | RAME |
| SYSTEM | SBYCR | SSBY | OPE | — | STS[4:0] | | | | |
| | | — | — | — | — | — | — | — | — |
| SYSTEM | MSTPCRA | ACSE | — | — | MSTPA28 | MSTPA27 | — | — | — |
| | | MSTPA23 | MSTPA22 | MSTPA21 | MSTPA20 | MSTPA19 | — | — | — |
| | | MSTPA15 | MSTPA14 | MSTPA13 | MSTPA12 | MSTPA11 | MSTPA10 | — | — |
| | | — | — | MSTPA5 | MSTPA4 | — | — | — | — |
| SYSTEM | MSTPCRB | MSTPB31 | MSTPB30 | MSTPB29 | MSTPB28 | MSTPB27 | MSTPB26 | MSTPB25 | — |
| | | MSTPB23 | — | MSTPB21 | MSTPB20 | — | — | — | — |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | — | — |
| SYSTEM | MSTPCRC | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | MSTPC1 | MSTPC0 |
| SYSTEM | SCKCR | — | — | — | — | ICK[3:0] | | | |
| | | PSTOP1 | — | — | — | BCK[3:0] | | | |
| | | — | — | — | — | PCK[3:0] | | | |
| | | — | — | — | — | — | — | — | — |
| BSC | BERCLR | — | — | — | — | — | — | — | STSCLR |
| BSC | BEREN | — | — | — | — | — | — | TOEN | IGAEN |
| BSC | BERIE | — | — | — | — | — | — | — | CPEN |
| DMAC0 | DMCSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC0 | DMCDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC0 | DMCBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| DMAC0 | DMMOD | — | — | — | — | OPSEL[3:0] | | | |
| | | — | — | — | — | — | SZSEL[2:0] | | |
| | | — | SMOD[2:0] | | | — | DMOD[2:0] | | |
| | | — | — | — | — | — | — | — | — |
| DMAC1 | DMCSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMCDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMCBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMMOD | — | — | — | — | OPSEL[3:0] | | | |
| | | — | — | — | — | — | SZSEL[2:0] | | |
| | | — | SMOD[2:0] | | | — | DMOD[2:0] | | |
| | | — | — | — | — | — | — | — | — |
| DMAC2 | DMCSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMCDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMCBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMMOD | — | — | — | — | OPSEL[3:0] | | | |
| | | — | — | — | — | — | SZSEL[2:0] | | |
| | | — | SMOD[2:0] | | | — | DMOD[2:0] | | |
| | | — | — | — | — | — | — | — | — |
| DMAC3 | DMCSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC3 | DMCDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| DMAC3 | DMCBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC3 | DMMOD | — | — | — | — | OPSEL[3:0] | | | |
| | | — | — | — | — | — | SZSEL[2:0] | | |
| | | — | SMOD[2:0] | | | — | DMOD[2:0] | | |
| | | — | — | — | — | — | — | — | — |
| DMAC0 | DMRSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC0 | DMRDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC0 | DMRBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMRSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMRDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC1 | DMRBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMRSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMRDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC2 | DMRBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| DMAC3 | DMRSA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC3 | DMRDA | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC3 | DMRBC | — | — | — | — | — | — | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DMAC0 | DMCRA | — | — | — | — | — | — | DSEL[1:0] | |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | BRLOD | SRLOD | DRLOD |
| | | — | — | DCTG[5:0] | | | | | |
| DMAC0 | DMCRB | — | — | — | — | — | — | — | DSCLR |
| DMAC0 | DMCRC | — | — | — | — | — | — | — | ECLR |
| DMAC0 | DMCRD | — | — | — | — | — | — | — | DREQ |
| DMAC0 | DMCRE | — | — | — | — | — | — | — | DEN |
| DMAC1 | DMCRA | — | — | — | — | — | — | DSEL[1:0] | |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | BRLOD | SRLOD | DRLOD |
| | | — | — | DCTG[5:0] | | | | | |
| DMAC1 | DMCRB | — | — | — | — | — | — | — | DSCLR |
| DMAC1 | DMCRC | — | — | — | — | — | — | — | ECLR |
| DMAC1 | DMCRD | — | — | — | — | — | — | — | DREQ |
| DMAC1 | DMCRE | — | — | — | — | — | — | — | DEN |
| DMAC2 | DMCRA | — | — | — | — | — | — | DSEL[1:0] | |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | BRLOD | SRLOD | DRLOD |
| | | — | — | DCTG[5:0] | | | | | |
| DMAC2 | DMCRB | — | — | — | — | — | — | — | DSCLR |
| DMAC2 | DMCRC | — | — | — | — | — | — | — | ECLR |
| DMAC2 | DMCRD | — | — | — | — | — | — | — | DREQ |
| DMAC2 | DMCRE | — | — | — | — | — | — | — | DEN |
| DMAC3 | DMCRA | — | — | — | — | — | — | DSEL[1:0] | |
| | | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | BRLOD | SRLOD | DRLOD |
| | | — | — | DCTG[5:0] | | | | | |
| DMAC3 | DMCRB | — | — | — | — | — | — | — | DSCLR |
| DMAC3 | DMCRC | — | — | — | — | — | — | — | ECLR |
| DMAC3 | DMCRD | — | — | — | — | — | — | — | DREQ |
| DMAC3 | DMCRE | — | — | — | — | — | — | — | DEN |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| Common to all DMAC channels | DMSCNT | — | — | — | — | — | — | — | DMST |
| Common to all DMAC channels | DMICNT | DINTM0 | DINTM1 | DINTM2 | DINTM3 | — | — | — | — |
| Common to all DMAC channels | DMEDET | DEDET0 | DEDET1 | DEDET2 | DEDET3 | — | — | — | — |
| Common to all DMAC channels | DMASTS | DASTS0 | DASTS1 | DASTS2 | DASTS3 | — | — | — | — |
| BSC | CS0MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS0WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS0WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS1MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS1WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS1WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS2MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS2WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS2WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS3MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| BSC | CS3WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS3WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS4MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS4WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT [2:0] | | |
| BSC | CS4WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS5MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS5WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS5WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS6MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS6WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |
| BSC | CS6WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS7MOD | PRMOD | — | — | — | — | — | PWENB | PRENB |
| | | — | — | — | — | EWENB | — | — | WRMOD |
| BSC | CS7WCNT1 | — | — | — | CSRWAIT[4:0] | | | | |
| | | — | — | — | CSWWAIT[4 0] | | | | |
| | | — | — | — | — | — | CSPRWAIT[2:0] | | |
| | | — | — | — | — | — | CSPWWAIT[2:0] | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| BSC | CS7WCNT2 | — | CSON[2:0] | | | — | WDON[2:0] | | |
| | | — | WRON[2:0] | | | — | RDON[2:0] | | |
| | | — | — | — | — | — | WDOFF[2:0] | | |
| | | — | CSWOFF[2:0] | | | — | CSROFF[2:0] | | |
| BSC | CS0CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS0REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS1CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS1REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS2CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS2REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS3CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS3REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS4CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS4REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS5CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS5REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS6CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS6REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| BSC | CS7CNT | — | — | — | — | — | — | — | EMODE |
| | | — | — | BSIZE[1:0] | | — | — | — | EXENB |
| BSC | CS7REC | — | — | — | — | WRCV[3:0] | | | |
| | | — | — | — | — | RRCV[3:0] | | | |
| ICU | IR016 | — | — | — | — | — | — | — | IR |
| ICU | IR021 | — | — | — | — | — | — | — | IR |
| ICU | IR023 | — | — | — | — | — | — | — | IR |
| ICU | IR028 | — | — | — | — | — | — | — | IR |
| ICU | IR029 | — | — | — | — | — | — | — | IR |
| ICU | IR030 | — | — | — | — | — | — | — | IR |
| ICU | IR031 | — | — | — | — | — | — | — | IR |
| ICU | IR064 | — | — | — | — | — | — | — | IR |
| ICU | IR065 | — | — | — | — | — | — | — | IR |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IR066 | — | — | — | — | — | — | — | IR |
| ICU | IR067 | — | — | — | — | — | — | — | IR |
| ICU | IR068 | — | — | — | — | — | — | — | IR |
| ICU | IR069 | — | — | — | — | — | — | — | IR |
| ICU | IR070 | — | — | — | — | — | — | — | IR |
| ICU | IR071 | — | — | — | — | — | — | — | IR |
| ICU | IR072 | — | — | — | — | — | — | — | IR |
| ICU | IR073 | — | — | — | — | — | — | — | IR |
| ICU | IR074 | — | — | — | — | — | — | — | IR |
| ICU | IR075 | — | — | — | — | — | — | — | IR |
| ICU | IR076 | — | — | — | — | — | — | — | IR |
| ICU | IR077 | — | — | — | — | — | — | — | IR |
| ICU | IR078 | — | — | — | — | — | — | — | IR |
| ICU | IR079 | — | — | — | — | — | — | — | IR |
| ICU | IR096 | — | — | — | — | — | — | — | IR |
| ICU | IR098 | — | — | — | — | — | — | — | IR |
| ICU | IR099 | — | — | — | — | — | — | — | IR |
| ICU | IR100 | — | — | — | — | — | — | — | IR |
| ICU | IR101 | — | — | — | — | — | — | — | IR |
| ICU | IR104 | — | — | — | — | — | — | — | IR |
| ICU | IR105 | — | — | — | — | — | — | — | IR |
| ICU | IR106 | — | — | — | — | — | — | — | IR |
| ICU | IR107 | — | — | — | — | — | — | — | IR |
| ICU | IR108 | — | — | — | — | — | — | — | IR |
| ICU | IR111 | — | — | — | — | — | — | — | IR |
| ICU | IR112 | — | — | — | — | — | — | — | IR |
| ICU | IR115 | — | — | — | — | — | — | — | IR |
| ICU | IR116 | — | — | — | — | — | — | — | IR |
| ICU | IR117 | — | — | — | — | — | — | — | IR |
| ICU | IR118 | — | — | — | — | — | — | — | IR |
| ICU | IR120 | — | — | — | — | — | — | — | IR |
| ICU | IR121 | — | — | — | — | — | — | — | IR |
| ICU | IR122 | — | — | — | — | — | — | — | IR |
| ICU | IR123 | — | — | — | — | — | — | — | IR |
| ICU | IR124 | — | — | — | — | — | — | — | IR |
| ICU | IR125 | — | — | — | — | — | — | — | IR |
| ICU | IR126 | — | — | — | — | — | — | — | IR |
| ICU | IR127 | — | — | — | — | — | — | — | IR |
| ICU | IR128 | — | — | — | — | — | — | — | IR |
| ICU | IR131 | — | — | — | — | — | — | — | IR |
| ICU | IR132 | — | — | — | — | — | — | — | IR |
| ICU | IR133 | — | — | — | — | — | — | — | IR |
| ICU | IR134 | — | — | — | — | — | — | — | IR |
| ICU | IR136 | — | — | — | — | — | — | — | IR |
| ICU | IR137 | — | — | — | — | — | — | — | IR |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IR138 | — | — | — | — | — | — | — | IR |
| ICU | IR139 | — | — | — | — | — | — | — | IR |
| ICU | IR140 | — | — | — | — | — | — | — | IR |
| ICU | IR141 | — | — | — | — | — | — | — | IR |
| ICU | IR142 | — | — | — | — | — | — | — | IR |
| ICU | IR145 | — | — | — | — | — | — | — | IR |
| ICU | IR146 | — | — | — | — | — | — | — | IR |
| ICU | IR149 | — | — | — | — | — | — | — | IR |
| ICU | IR150 | — | — | — | — | — | — | — | IR |
| ICU | IR151 | — | — | — | — | — | — | — | IR |
| ICU | IR152 | — | — | — | — | — | — | — | IR |
| ICU | IR154 | — | — | — | — | — | — | — | IR |
| ICU | IR155 | — | — | — | — | — | — | — | IR |
| ICU | IR156 | — | — | — | — | — | — | — | IR |
| ICU | IR157 | — | — | — | — | — | — | — | IR |
| ICU | IR158 | — | — | — | — | — | — | — | IR |
| ICU | IR159 | — | — | — | — | — | — | — | IR |
| ICU | IR160 | — | — | — | — | — | — | — | IR |
| ICU | IR161 | — | — | — | — | — | — | — | IR |
| ICU | IR162 | — | — | — | — | — | — | — | IR |
| ICU | IR165 | — | — | — | — | — | — | — | IR |
| ICU | IR166 | — | — | — | — | — | — | — | IR |
| ICU | IR167 | — | — | — | — | — | — | — | IR |
| ICU | IR168 | — | — | — | — | — | — | — | IR |
| ICU | IR170 | — | — | — | — | — | — | — | IR |
| ICU | IR171 | — | — | — | — | — | — | — | IR |
| ICU | IR174 | — | — | — | — | — | — | — | IR |
| ICU | IR175 | — | — | — | — | — | — | — | IR |
| ICU | IR176 | — | — | — | — | — | — | — | IR |
| ICU | IR177 | — | — | — | — | — | — | — | IR |
| ICU | IR178 | — | — | — | — | — | — | — | IR |
| ICU | IR179 | — | — | — | — | — | — | — | IR |
| ICU | IR180 | — | — | — | — | — | — | — | IR |
| ICU | IR181 | — | — | — | — | — | — | — | IR |
| ICU | IR182 | — | — | — | — | — | — | — | IR |
| ICU | IR183 | — | — | — | — | — | — | — | IR |
| ICU | IR184 | — | — | — | — | — | — | — | IR |
| ICU | IR185 | — | — | — | — | — | — | — | IR |
| ICU | IR198 | — | — | — | — | — | — | — | IR |
| ICU | IR199 | — | — | — | — | — | — | — | IR |
| ICU | IR200 | — | — | — | — | — | — | — | IR |
| ICU | IR201 | — | — | — | — | — | — | — | IR |
| ICU | IR214 | — | — | — | — | — | — | — | IR |
| ICU | IR215 | — | — | — | — | — | — | — | IR |
| ICU | IR216 | — | — | — | — | — | — | — | IR |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IR217 | — | — | — | — | — | — | — | IR |
| ICU | IR218 | — | — | — | — | — | — | — | IR |
| ICU | IR219 | — | — | — | — | — | — | — | IR |
| ICU | IR220 | — | — | — | — | — | — | — | IR |
| ICU | IR221 | — | — | — | — | — | — | — | IR |
| ICU | IR222 | — | — | — | — | — | — | — | IR |
| ICU | IR223 | — | — | — | — | — | — | — | IR |
| ICU | IR224 | — | — | — | — | — | — | — | IR |
| ICU | IR225 | — | — | — | — | — | — | — | IR |
| ICU | IR226 | — | — | — | — | — | — | — | IR |
| ICU | IR227 | — | — | — | — | — | — | — | IR |
| ICU | IR228 | — | — | — | — | — | — | — | IR |
| ICU | IR229 | — | — | — | — | — | — | — | IR |
| ICU | IR230 | — | — | — | — | — | — | — | IR |
| ICU | IR231 | — | — | — | — | — | — | — | IR |
| ICU | IR232 | — | — | — | — | — | — | — | IR |
| ICU | IR233 | — | — | — | — | — | — | — | IR |
| ICU | IR234 | — | — | — | — | — | — | — | IR |
| ICU | IR235 | — | — | — | — | — | — | — | IR |
| ICU | IR236 | — | — | — | — | — | — | — | IR |
| ICU | IR237 | — | — | — | — | — | — | — | IR |
| ICU | IR238 | — | — | — | — | — | — | — | IR |
| ICU | IR239 | — | — | — | — | — | — | — | IR |
| ICU | IR240 | — | — | — | — | — | — | — | IR |
| ICU | IR241 | — | — | — | — | — | — | — | IR |
| ICU | IR246 | — | — | — | — | — | — | — | IR |
| ICU | IR247 | — | — | — | — | — | — | — | IR |
| ICU | IR248 | — | — | — | — | — | — | — | IR |
| ICU | IR249 | — | — | — | — | — | — | — | IR |
| ICU | IR250 | — | — | — | — | — | — | — | IR |
| ICU | IR251 | — | — | — | — | — | — | — | IR |
| ICU | IR252 | — | — | — | — | — | — | — | IR |
| ICU | IR253 | — | — | — | — | — | — | — | IR |
| ICU | ISELR028 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR029 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR030 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR031 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR064 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR065 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR066 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR067 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR068 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR069 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR070 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR071 | — | — | — | — | — | — | ISEL[1:0] | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | ISELR072 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR073 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR074 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR075 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR076 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR077 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR078 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR079 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR098 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR099 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR100 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR101 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR104 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR105 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR106 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR107 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR111 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR112 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR117 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR118 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR122 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR123 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR124 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR125 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR127 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR128 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR133 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR134 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR138 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR139 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR140 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR141 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR145 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR146 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR151 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR152 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR156 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR157 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR158 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR159 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR161 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR162 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR167 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR168 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR174 | — | — | — | — | — | — | ISEL[1:0] | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | ISELR175 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR177 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR178 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR180 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR181 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR183 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR184 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR198 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR199 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR200 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR201 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR215 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR216 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR219 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR220 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR223 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR224 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR227 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR228 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR231 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR232 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR235 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR236 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR239 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR240 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR247 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR248 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR251 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR252 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | ISELR253 | — | — | — | — | — | — | ISEL[1:0] | |
| ICU | IER02 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER03 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER08 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER09 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER0C | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER0D | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER0E | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER0F | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER10 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER11 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER12 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER13 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER14 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER15 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER16 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IER17 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER18 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER19 | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1A | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1B | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1C | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1D | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1E | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IER1F | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| ICU | IPR00 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR01 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR02 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR04 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR05 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR06 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR07 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR20 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR21 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR22 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR23 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR24 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR25 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR26 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR27 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR28 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR29 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2A | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2B | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2C | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2D | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2E | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR2F | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR40 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR44 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR45 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR46 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR47 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR4C | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR4D | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR4E | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR4F | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR50 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR51 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR52 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR53 | — | — | — | — | — | IPR[2:0] | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IPR54 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR55 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR56 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR57 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR58 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR59 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5A | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5B | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5C | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5D | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5E | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR5F | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR60 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR61 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR62 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR63 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR68 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR69 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR6A | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR6B | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR70 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR71 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR72 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR73 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR80 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR81 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR82 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR83 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR84 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR85 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR86 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR89 | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8A | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8B | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8C | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8D | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8E | — | — | — | — | — | IPR[2:0] | | |
| ICU | IPR8F | — | — | — | — | — | IPR[2:0] | | |
| ICU | FIR | FIEN | — | — | — | — | — | — | — |
| | | FVCT[7:0] | | | | | | | |
| DTC | DTCCR | — | — | — | RRS | RCHNE | — | — | ERR |
| DTC | DTCVBR | | | | | | | | |
| | | | | | | | | | |
| | | | | | | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| DTC | DTCADMOD | — | — | — | — | — | — | — | SHORT |
| DTC | DTCST | — | — | — | — | — | — | — | DTCST |
| CMT (unit 0) | CMSTR0 | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | STR1 | STR0 |
| CMT0 | CMCR | — | — | — | — | — | — | — | — |
| | | — | CMIE | — | — | — | — | CKS[1:0] | |
| CMT0 | CMCNT | | | | | | | | |
| | | | | | | | | | |
| CMT0 | CMCOR | | | | | | | | |
| | | | | | | | | | |
| CMT1 | CMCR | — | — | — | — | — | — | — | — |
| | | — | CMIE | — | — | — | — | CKS[1:0] | |
| CMT1 | CMCNT | | | | | | | | |
| | | | | | | | | | |
| CMT1 | CMCOR | | | | | | | | |
| | | | | | | | | | |
| CMT (unit 1) | CMSTR1 | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | STR3 | STR2 |
| CMT2 | CMCR | — | — | — | — | — | — | — | — |
| | | — | CMIE | — | — | — | — | CKS[1:0] | |
| CMT2 | CMCNT | | | | | | | | |
| | | | | | | | | | |
| CMT2 | CMCOR | | | | | | | | |
| | | | | | | | | | |
| CMT3 | CMCR | — | — | — | — | — | — | — | — |
| | | — | CMIE | — | — | — | — | CKS[1:0] | |
| CMT3 | CMCNT | | | | | | | | |
| | | | | | | | | | |
| CMT3 | CMCOR | | | | | | | | |
| | | | | | | | | | |
| WDT | TCSR | — | TMS | TME | — | — | CKS[2:0] | | |
| WDT | WINA | | | | | | | | |
| | | | | | | | | | |
| WDT | TCNT | | | | | | | | |
| WDT | WINB | | | | | | | | |
| | | | | | | | | | |
| WDT | RSTCSR | WOVF | RSTE | — | — | — | — | — | — |
| AD0 | ADDRA | | | | | | | | |
| | | | | | | | | | |
| AD0 | ADDRB | | | | | | | | |
| | | | | | | | | | |
| AD0 | ADDRC | | | | | | | | |
| | | | | | | | | | |
| AD0 | ADDRD | | | | | | | | |
| | | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| AD0 | ADCSR | — | ADIE | ADST | — | CH[3:0] | | | |
| AD0 | ADCR | TRGS[2:0] | | | — | CKS[1:0] | | MODE[1:0] | |
| AD0 | ADDPR | DPSEL | — | — | — | — | — | — | — |
| AD0 | ADSSTR | | | | | | | | |
| AD1 | ADDRA | | | | | | | | |
| | | | | | | | | | |
| AD1 | ADDRB | | | | | | | | |
| | | | | | | | | | |
| AD1 | ADDRC | | | | | | | | |
| | | | | | | | | | |
| AD1 | ADDRD | | | | | | | | |
| | | | | | | | | | |
| AD1 | ADCSR | — | ADIE | ADST | — | CH[3:0] | | | |
| AD1 | ADCR | TRGS[2:0] | | | — | CKS[1:0] | | MODE[1:0] | |
| AD1 | ADDPR | DPSEL | — | — | — | — | — | — | — |
| AD1 | ADSSTR | | | | | | | | |
| AD2 | ADDRA | | | | | | | | |
| | | | | | | | | | |
| AD2 | ADDRB | | | | | | | | |
| | | | | | | | | | |
| AD2 | ADDRC | | | | | | | | |
| | | | | | | | | | |
| AD2 | ADDRD | | | | | | | | |
| | | | | | | | | | |
| AD2 | ADCSR | — | ADIE | ADST | — | CH[3:0] | | | |
| AD2 | ADCR | TRGS[2:0] | | | — | CKS[1:0] | | MODE[1:0] | |
| AD2 | ADDPR | DPSEL | — | — | — | — | — | — | — |
| AD2 | ADSSTR | | | | | | | | |
| AD3 | ADDRA | | | | | | | | |
| | | | | | | | | | |
| AD3 | ADDRB | | | | | | | | |
| | | | | | | | | | |
| AD3 | ADDRC | | | | | | | | |
| | | | | | | | | | |
| AD3 | ADDRD | | | | | | | | |
| | | | | | | | | | |
| AD3 | ADCSR | — | ADIE | ADST | — | CH[3:0] | | | |
| AD3 | ADCR | TRGS[2:0] | | | — | CKS[1:0] | | MODE[1:0] | |
| AD3 | ADDPR | DPSEL | — | — | — | — | — | — | — |
| AD3 | ADSSTR | | | | | | | | |
| D/A | DADR0 | | | | | | | | |
| | | | | | | | | | |
| D/A | DADR1 | | | | | | | | |
| | | | | | | | | | |
| D/A | DACR | DAOE1 | DAOE0 | DAE | — | — | — | — | — |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| D/A | DADPR | DPSEL | — | — | — | — | — | — | — |
| TPU (unit 0) | TSTRA | — | — | CST5 | CST4 | CST3 | CST2 | CST1 | CST0 |
| TPU (unit 0) | TSYRA | — | — | SYNC5 | SYNC4 | SYNC3 | SYNC2 | SYNC1 | SYNC0 |
| TPU0 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU0 | TMDR | ICSELD | ICSELB | BFB | BFA | MD[3 0] | | | |
| TPU0 | TIORH | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU0 | TIORL | IOD[3:0] | | | | IOC[3:0] | | | |
| TPU0 | TIER | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |
| TPU0 | TSR | — | — | — | — | — | — | — | — |
| TPU0 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU0 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU0 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU0 | TGRC | | | | | | | | |
| | | | | | | | | | |
| TPU0 | TGRD | | | | | | | | |
| | | | | | | | | | |
| TPU1 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU1 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU1 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU1 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU1 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU1 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU1 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU1 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU2 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU2 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU2 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU2 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU2 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU2 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU2 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU2 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU3 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU3 | TMDR | ICSELD | ICSELB | BFB | BFA | MD[3 0] | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| TPU3 | TIORH | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU3 | TIORL | IOD[3:0] | | | | IOC[3:0] | | | |
| TPU3 | TIER | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |
| TPU3 | TSR | — | — | — | — | — | — | — | — |
| TPU3 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU3 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU3 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU3 | TGRC | | | | | | | | |
| | | | | | | | | | |
| TPU3 | TGRD | | | | | | | | |
| | | | | | | | | | |
| TPU4 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU4 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU4 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU4 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU4 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU4 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU4 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU4 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU5 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU5 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU5 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU5 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU5 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU5 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU5 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU5 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU (unit 1) | TSTRB | — | — | CST5 | CST4 | CST3 | CST2 | CST1 | CST0 |
| TPU (unit 1) | TSYRB | — | — | SYNC5 | SYNC4 | SYNC3 | SYNC2 | SYNC1 | SYNC0 |
| TPU6 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU6 | TMDR | ICSELD | ICSELB | BFB | BFA | MD[3 0] | | | |
| TPU6 | TIORH | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU6 | TIORL | IOD[3:0] | | | | IOC[3:0] | | | |
| TPU6 | TIER | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| TPU6 | TSR | — | — | — | — | — | — | — | — |
| TPU6 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU6 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU6 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU6 | TGRC | | | | | | | | |
| | | | | | | | | | |
| TPU6 | TGRD | | | | | | | | |
| | | | | | | | | | |
| TPU7 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU7 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU7 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU7 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU7 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU7 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU7 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU7 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU8 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU8 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU8 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU8 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU8 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU8 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU8 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU8 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU9 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU9 | TMDR | ICSELD | ICSELB | BFB | BFA | MD[3 0] | | | |
| TPU9 | TIORH | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU9 | TIORL | IOD[3:0] | | | | IOC[3:0] | | | |
| TPU9 | TIER | TTGE | — | — | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |
| TPU9 | TSR | — | — | — | — | — | — | — | — |
| TPU9 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU9 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU9 | TGRB | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| TPU9 | TGRC | | | | | | | | |
| | | | | | | | | | |
| TPU9 | TGRD | | | | | | | | |
| | | | | | | | | | |
| TPU10 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU10 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU10 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU10 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU10 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU10 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU10 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU10 | TGRB | | | | | | | | |
| | | | | | | | | | |
| TPU11 | TCR | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| TPU11 | TMDR | — | ICSELB | — | — | MD[3 0] | | | |
| TPU11 | TIOR | IOB[3:0] | | | | IOA[3:0] | | | |
| TPU11 | TIER | TTGE | — | TCIEU | TCIEV | — | — | TGIEB | TGIEA |
| TPU11 | TSR | TCFD | — | — | — | — | — | — | — |
| TPU11 | TCNT | | | | | | | | |
| | | | | | | | | | |
| TPU11 | TGRA | | | | | | | | |
| | | | | | | | | | |
| TPU11 | TGRB | | | | | | | | |
| | | | | | | | | | |
| PPG0 | PCR | G3CMS[1:0] | | G2CMS[1:0] | | G1CMS[1:0] | | G0CMS[1:0] | |
| PPG0 | PMR | G3INV | G2INV | G1INV | G0INV | G3NOV | G2NOV | G1NOV | G0NOV |
| PPG0 | NDERH | NDER15 | NDER14 | NDER13 | NDER12 | NDER11 | NDER10 | NDER9 | NDER8 |
| PPG0 | NDERL | NDER7 | NDER6 | NDER5 | NDER4 | NDER3 | NDER2 | NDER1 | NDER0 |
| PPG0 | PODRH | POD15 | POD14 | POD13 | POD12 | POD11 | POD10 | POD9 | POD8 |
| PPG0 | PODRL | POD7 | POD6 | POD5 | POD4 | POD3 | POD2 | POD1 | POD0 |
| PPG0 | NDRH*1 | NDR15 | NDR14 | NDR13 | NDR12 | NDR11 (—) | NDR10 (—) | NDR9 (—) | NDR8 (—) |
| PPG0 | NDRL*2 | NDR7 | NDR6 | NDR5 | NDR4 | NDR3 (—) | NDR2 (—) | NDR1 (—) | NDR0 (—) |
| PPG0 | NDRH*1 | — | — | — | — | NDR11 | NDR10 | NDR9 | NDR8 |
| PPG0 | NDRL*2 | — | — | — | — | NDR3 | NDR2 | NDR1 | NDR0 |
| PPG1 | PTRSLR | — | — | — | — | — | — | — | PTRSL |
| PPG1 | PCR | G3CMS[1:0] | | G2CMS[1:0] | | G1CMS[1:0] | | G0CMS[1:0] | |
| PPG1 | PMR | G3INV | G2INV | G1INV | G0INV | G3NOV | G2NOV | G1NOV | G0NOV |
| PPG1 | NDERH | NDER31 | NDER30 | NDER29 | NDER28 | NDER27 | NDER26 | NDER25 | NDER24 |
| PPG1 | NDERL | NDER23 | NDER22 | NDER21 | NDER20 | NDER19 | NDER18 | NDER17 | NDER16 |
| PPG1 | PODRH | POD31 | POD30 | POD29 | POD28 | POD27 | POD26 | POD25 | POD24 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| PPG1 | PODRL | POD23 | POD22 | POD21 | POD20 | POD19 | POD18 | POD17 | POD16 |
| PPG1 | NDRH*3 | NDR31 | NDR30 | NDR29 | NDR28 | NDR27 (—) | NDR26 (—) | NDR25 (—) | NDR24 (—) |
| PPG1 | NDRL*4 | NDR23 | NDR22 | NDR21 | NDR20 | NDR19 (—) | NDR18 (—) | NDR17 (—) | NDR16 (—) |
| PPG1 | NDRH*3 | — | — | — | — | NDR27 | NDR26 | NDR25 | NDR24 |
| PPG1 | NDRL*4 | — | — | — | — | NDR19 | NDR18 | NDR17 | NDR16 |
| TMR0 | TCR | CMIEB | CMIEA | OVIE | CCLR[1:0] | | — | — | — |
| TMR1 | TCR | CMIEB | CMIEA | OVIE | CCLR[1:0] | | — | — | — |
| TMR0 | TCSR | — | — | — | ADTE | OSB[1:0] | | OSA[1:0] | |
| TMR1 | TCSR | — | — | — | — | OSB[1:0] | | OSA[1:0] | |
| TMR0 | TCORA | | | | | | | | |
| TMR1 | TCORA | | | | | | | | |
| TMR0 | TCORB | | | | | | | | |
| TMR1 | TCORB | | | | | | | | |
| TMR0 | TCNT | | | | | | | | |
| TMR1 | TCNT | | | | | | | | |
| TMR0 | TCCR | TMRIS | — | — | CSS[1:0] | | CKS[2:0] | | |
| TMR1 | TCCR | TMRIS | — | — | CSS[1:0] | | CKS[2:0] | | |
| TMR2 | TCR | CMIEB | CMIEA | OVIE | CCLR[1:0] | | — | — | — |
| TMR3 | TCR | CMIEB | CMIEA | OVIE | CCLR[1:0] | | — | — | — |
| TMR2 | TCSR | — | — | — | ADTE | OSB[1:0] | | OSA[1:0] | |
| TMR3 | TCSR | — | — | — | — | OSB[1:0] | | OSA[1:0] | |
| TMR2 | TCORA | | | | | | | | |
| TMR3 | TCORA | | | | | | | | |
| TMR2 | TCORB | | | | | | | | |
| TMR3 | TCORB | | | | | | | | |
| TMR2 | TCNT | | | | | | | | |
| TMR3 | TCNT | | | | | | | | |
| TMR2 | TCCR | TMRIS | — | — | CSS[1:0] | | CKS[2:0] | | |
| TMR3 | TCCR | TMRIS | — | — | CSS[1:0] | | CKS[2:0] | | |
| SCI0 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI0 | BRR | | | | | | | | |
| SCI0 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI0 | TDR | | | | | | | | |
| SCI0 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI0 | RDR | | | | | | | | |
| SCI0 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI0 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI1 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI1 | BRR | | | | | | | | |
| SCI1 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI1 | TDR | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| SCI1 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI1 | RDR | | | | | | | | |
| SCI1 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI1 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI2 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI2 | BRR | | | | | | | | |
| SCI2 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI2 | TDR | | | | | | | | |
| SCI2 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI2 | RDR | | | | | | | | |
| SCI2 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI2 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI3 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI3 | BRR | | | | | | | | |
| SCI3 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI3 | TDR | | | | | | | | |
| SCI3 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI3 | RDR | | | | | | | | |
| SCI3 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI3 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI4 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI4 | BRR | | | | | | | | |
| SCI4 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI4 | TDR | | | | | | | | |
| SCI4 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI4 | RDR | | | | | | | | |
| SCI4 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI4 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI5 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI5 | BRR | | | | | | | | |
| SCI5 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI5 | TDR | | | | | | | | |
| SCI5 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI5 | RDR | | | | | | | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| SCI5 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI5 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| SCI6 | SMR*5 | CM (GM) | CHR (BLK) | PE (PE) | PM (PM) | STOP (BCP[1:0]) | — | CKS[1:0] (CKS[1:0]) | |
| SCI6 | BRR | | | | | | | | |
| SCI6 | SCR*5 | TIE | RIE | TE | RE | — | TEIE | CKE[1:0] | |
| SCI6 | TDR | | | | | | | | |
| SCI6 | SSR*5 | TDRE (TDRE) | RDRF (RDRF) | ORER (ORER) | FER (ERS) | PER (PER) | TEND (TEND) | — (—) | — (—) |
| SCI6 | RDR | | | | | | | | |
| SCI6 | SCMR | BCP2 | — | — | — | SDIR | SINV | — | SMIF |
| SCI6 | SEMR | — | — | — | ABCS | — | — | — | ACS0 |
| CRC | CRCCR | DORCLR | — | — | — | — | LMS | GPS[1:0] | |
| CRC | CRCDIR | | | | | | | | |
| CRC | CRCDOR | | | | | | | | |
| | | | | | | | | | |
| RIIC0 | ICCR1 | ICE | IICRST | CLO | SOWP | SCLO | SDAO | SCLI | SDAI |
| RIIC0 | ICCR2 | BBSY | MST | TRS | — | SP | RS | ST | — |
| RIIC0 | ICMR1 | MTWP | CKS[2:0] | | | BCWP | BC[2 0] | | |
| RIIC0 | ICMR2 | DLCS | SDDL[2:0] | | | TMWE | TMOH | TMOL | TMOS |
| RIIC0 | ICMR3 | SMBS | WAIT | RDRFS | ACKWP | ACKBT | ACKBR | NF[1:0] | |
| RIIC0 | ICFER | FMPE | SCLE | NFE | NACKE | SALE | NALE | MALE | TMOE |
| RIIC0 | ICSER | HOAE | — | DIDE | — | GCAE | SAR2E | SAR1E | SAR0E |
| RIIC0 | ICIER | TIE | TEIE | RIE | NAKIE | SPIE | STIE | ALIE | TMOIE |
| RIIC0 | ICSR1 | HOA | — | DID | — | GCA | AAS2 | AAS1 | AAS0 |
| RIIC0 | ICSR2 | TDRE | TEND | RDRF | NACKF | STOP | START | AL | TMOF |
| RIIC0 | SARL0 | SVA[7:1] | | | | | | | SVA0 |
| RIIC0 | SARU0 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC0 | TMOCNTL | | | | | | | | |
| | | | | | | | | | |
| RIIC0 | SARL1 | SVA[7:1] | | | | | | | SVA0 |
| RIIC0 | TMOCNTU | | | | | | | | |
| | | | | | | | | | |
| RIIC0 | SARU1 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC0 | SARL2 | SVA[7:1] | | | | | | | SVA0 |
| RIIC0 | SARU2 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC0 | ICBRL | — | — | — | BRL[4:0] | | | | |
| RIIC0 | ICBRH | — | — | — | BRH[4:0] | | | | |
| RIIC0 | ICDRT | | | | | | | | |
| RIIC0 | ICDRR | | | | | | | | |
| RIIC1 | ICCR1 | ICE | IICRST | CLO | SOWP | SCLO | SDAO | SCLI | SDAI |
| RIIC1 | ICCR2 | BBSY | MST | TRS | — | SP | RS | ST | — |
| RIIC1 | ICMR1 | MTWP | CKS[2:0] | | | BCWP | BC[2 0] | | |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| RIIC1 | ICMR2 | DLCS | SDDL[2:0] | | | TMWE | TMOH | TMOL | TMOS |
| RIIC1 | ICMR3 | SMBS | WAIT | RDRFS | ACKWP | ACKBT | ACKBR | NF[1:0] | |
| RIIC1 | ICFER | FMPE | SCLE | NFE | NACKE | SALE | NALE | MALE | TMOE |
| RIIC1 | ICSER | HOAE | — | DIDE | — | GCAE | SAR2E | SAR1E | SAR0E |
| RIIC1 | ICIER | TIE | TEIE | RIE | NAKIE | SPIE | STIE | ALIE | TMOIE |
| RIIC1 | ICSR1 | HOA | — | DID | — | GCA | AAS2 | AAS1 | AAS0 |
| RIIC1 | ICSR2 | TDRE | TEND | RDRF | NACKF | STOP | START | AL | TMOF |
| RIIC1 | SARL0 | SVA[7:1] | | | | | | | SVA0 |
| RIIC1 | TMOCNTU | | | | | | | | |
| | | | | | | | | | |
| RIIC1 | SARU0 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC1 | TMOCNTL | | | | | | | | |
| | | | | | | | | | |
| RIIC1 | SARL1 | SVA[7:1] | | | | | | | SVA0 |
| RIIC1 | SARU1 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC1 | SARL2 | SVA[7:1] | | | | | | | SVA0 |
| RIIC1 | SARU2 | — | — | — | — | — | SVA[9:8] | | FS |
| RIIC1 | ICBRL | — | — | — | BRL[4:0] | | | | |
| RIIC1 | ICBRH | — | — | — | BRH[4:0] | | | | |
| RIIC1 | ICDRT | | | | | | | | |
| RIIC1 | ICDRR | | | | | | | | |
| P0 | DDR | — | — | B5 | B4 | B3 | B2 | B1 | B0 |
| P1 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P2 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P3 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P4 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P5 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P6 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P7 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P8 | DDR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P9 | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PA | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PB | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PD | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PE | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PF | DDR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PG | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PH | DDR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P0 | DR | — | — | B5 | B4 | B3 | B2 | B1 | B0 |
| P1 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P2 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P3 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P4 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| P5 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P6 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P7 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P8 | DR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P9 | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PA | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PB | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PD | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PE | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PF | DR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PG | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PH | DR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P0 | PORT | — | — | B5 | B4 | B3 | B2 | B1 | B0 |
| P1 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P2 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P3 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P4 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P5 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P6 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P7 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P8 | PORT | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P9 | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PA | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PB | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PD | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PE | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PF | PORT | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PG | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PH | PORT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P0 | ICR | — | — | B5 | B4 | B3 | B2 | B1 | B0 |
| P1 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P2 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P3 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P4 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P5 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P6 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P7 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P8 | ICR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P9 | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PA | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PB | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PD | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| PE | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PF | ICR | — | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PG | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PH | ICR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| P2 | ODR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | ODR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PA | PCR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PB | PCR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PC | PCR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PD | PCR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PE | PCR | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| I/O PORT | PFCR0 | CS7E | CS6E | CS5E | CS4E | CS3E | CS2E | CS1E | CS0E |
| I/O PORT | PFCR1 | CS7S[1:0] | | CS6S[1:0] | | CS5S[1:0] | | CS4S[1:0] | |
| I/O PORT | PFCR2 | CS3S | CS2S | — | — | — | — | — | — |
| I/O PORT | PFCR3 | A23E | A22E | A21E | A20E | A19E | A18E | A17E | A16E |
| I/O PORT | PFCR4 | A15E | A14E | A13E | A12E | A11E | A10E | A9E | A8E |
| I/O PORT | PFCR5 | — | WR1BC1E | — | DHE | TCLKS | — | — | — |
| I/O PORT | PFCR6 | TPUMS5 | TPUMS4 | TPUMS3A | TPUMS3B | TPUMS2 | TPUMS1 | TPUMS0A | TPUMS0B |
| I/O PORT | PFCR7 | TPUMS11 | TPUMS10 | TPUMS9A | TPUMS9B | TPUMS8 | TPUMS7 | TPUMS6A | TPUMS6B |
| I/O PORT | PFCR8 | ITS15 | ITS14 | ITS13 | ITS12 | ITS11 | ITS10 | ITS9 | ITS8 |
| I/O PORT | PFCR9 | ITS7 | ITS6 | ITS5 | ITS4 | ITS3 | ITS2 | ITS1 | ITS0 |
| SYSTEM | DPSBYCR | DPSBY | IOKEEP | RAMCUT2 | RAMCUT1 | — | — | — | RAMCUT0 |
| SYSTEM | DPSWCR | — | — | WTSTS[5:0] | | | | | |
| SYSTEM | DPSIER | DNMIE | — | — | — | DIRQ3E | DIRQ2E | DIRQ1E | DIRQ0E |
| SYSTEM | DPSIFR | DNMIF | — | — | — | DIRQ3F | DIRQ2F | DIRQ1F | DIRQ0F |
| SYSTEM | DPSIEGR | DNMIEG | — | — | — | DIRQ3EG | DIRQ2EG | DIRQ1EG | DIRQ0EG |
| SYSTEM | RSTSR | DPSRSTF | — | — | — | — | — | — | — |
| FLASH | FWEPROR | — | — | — | — | — | — | FLWE[1:0] | |
| SYSTEM | DPSBKR0 | BKUP07 | BKUP06 | BKUP05 | BKUP04 | BKUP03 | BKUP02 | BKUP01 | BKUP00 |
| SYSTEM | DPSBKR1 | BKUP17 | BKUP16 | BKUP15 | BKUP14 | BKUP13 | BKUP12 | BKUP11 | BKUP10 |
| SYSTEM | DPSBKR2 | BKUP27 | BKUP26 | BKUP25 | BKUP24 | BKUP23 | BKUP22 | BKUP21 | BKUP20 |
| SYSTEM | DPSBKR3 | BKUP37 | BKUP36 | BKUP35 | BKUP34 | BKUP33 | BKUP32 | BKUP31 | BKUP30 |
| SYSTEM | DPSBKR4 | BKUP47 | BKUP46 | BKUP45 | BKUP44 | BKUP43 | BKUP42 | BKUP41 | BKUP40 |
| SYSTEM | DPSBKR5 | BKUP57 | BKUP56 | BKUP55 | BKUP54 | BKUP53 | BKUP52 | BKUP51 | BKUP50 |
| SYSTEM | DPSBKR6 | BKUP67 | BKUP66 | BKUP65 | BKUP64 | BKUP63 | BKUP62 | BKUP61 | BKUP60 |
| SYSTEM | DPSBKR7 | BKUP77 | BKUP76 | BKUP75 | BKUP74 | BKUP73 | BKUP72 | BKUP71 | BKUP70 |
| SYSTEM | DPSBKR8 | BKUP87 | BKUP86 | BKUP85 | BKUP84 | BKUP83 | BKUP82 | BKUP81 | BKUP80 |
| SYSTEM | DPSBKR9 | BKUP97 | BKUP96 | BKUP95 | BKUP94 | BKUP93 | BKUP92 | BKUP91 | BKUP90 |
| SYSTEM | DPSBKR10 | BKUP107 | BKUP106 | BKUP105 | BKUP104 | BKUP103 | BKUP102 | BKUP101 | BKUP100 |
| SYSTEM | DPSBKR11 | BKUP117 | BKUP116 | BKUP115 | BKUP114 | BKUP113 | BKUP112 | BKUP111 | BKUP110 |
| SYSTEM | DPSBKR12 | BKUP127 | BKUP126 | BKUP125 | BKUP124 | BKUP123 | BKUP122 | BKUP121 | BKUP120 |
| SYSTEM | DPSBKR13 | BKUP137 | BKUP136 | BKUP135 | BKUP134 | BKUP133 | BKUP132 | BKUP131 | BKUP130 |
| SYSTEM | DPSBKR14 | BKUP147 | BKUP146 | BKUP145 | BKUP144 | BKUP143 | BKUP142 | BKUP141 | BKUP140 |
| SYSTEM | DPSBKR15 | BKUP157 | BKUP156 | BKUP155 | BKUP154 | BKUP153 | BKUP152 | BKUP151 | BKUP150 |
| SYSTEM | DPSBKR16 | BKUP167 | BKUP166 | BKUP165 | BKUP164 | BKUP163 | BKUP162 | BKUP161 | BKUP160 |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| SYSTEM | DPSBKR17 | BKUP177 | BKUP176 | BKUP175 | BKUP174 | BKUP173 | BKUP172 | BKUP171 | BKUP170 |
| SYSTEM | DPSBKR18 | BKUP187 | BKUP186 | BKUP185 | BKUP184 | BKUP183 | BKUP182 | BKUP181 | BKUP180 |
| SYSTEM | DPSBKR19 | BKUP197 | BKUP196 | BKUP195 | BKUP194 | BKUP193 | BKUP192 | BKUP191 | BKUP190 |
| SYSTEM | DPSBKR20 | BKUP207 | BKUP206 | BKUP205 | BKUP204 | BKUP203 | BKUP202 | BKUP201 | BKUP200 |
| SYSTEM | DPSBKR21 | BKUP217 | BKUP216 | BKUP215 | BKUP214 | BKUP213 | BKUP212 | BKUP211 | BKUP210 |
| SYSTEM | DPSBKR22 | BKUP227 | BKUP226 | BKUP225 | BKUP224 | BKUP223 | BKUP222 | BKUP221 | BKUP220 |
| SYSTEM | DPSBKR23 | BKUP237 | BKUP236 | BKUP235 | BKUP234 | BKUP233 | BKUP232 | BKUP231 | BKUP230 |
| SYSTEM | DPSBKR24 | BKUP247 | BKUP246 | BKUP245 | BKUP244 | BKUP243 | BKUP242 | BKUP241 | BKUP240 |
| SYSTEM | DPSBKR25 | BKUP257 | BKUP256 | BKUP255 | BKUP254 | BKUP253 | BKUP252 | BKUP251 | BKUP250 |
| SYSTEM | DPSBKR26 | BKUP267 | BKUP266 | BKUP265 | BKUP264 | BKUP263 | BKUP262 | BKUP261 | BKUP260 |
| SYSTEM | DPSBKR27 | BKUP277 | BKUP276 | BKUP275 | BKUP274 | BKUP273 | BKUP272 | BKUP271 | BKUP270 |
| SYSTEM | DPSBKR28 | BKUP287 | BKUP286 | BKUP285 | BKUP284 | BKUP283 | BKUP282 | BKUP281 | BKUP280 |
| SYSTEM | DPSBKR29 | BKUP297 | BKUP296 | BKUP295 | BKUP294 | BKUP293 | BKUP292 | BKUP291 | BKUP290 |
| SYSTEM | DPSBKR30 | BKUP307 | BKUP306 | BKUP305 | BKUP304 | BKUP303 | BKUP302 | BKUP301 | BKUP300 |
| SYSTEM | DPSBKR31 | BKUP317 | BKUP316 | BKUP315 | BKUP314 | BKUP313 | BKUP312 | BKUP311 | BKUP310 |
| ICU | IRQER0 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER1 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER2 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER3 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER4 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER5 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER6 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER7 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER8 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER9 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER10 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER11 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER12 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER13 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER14 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQER15 | — | — | — | — | — | — | — | IRQEN |
| ICU | IRQCR0 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR1 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR2 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR3 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR4 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR5 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR6 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR7 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR8 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR9 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR10 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR11 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR12 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR13 | — | — | — | — | IRQMD[1:0] | | — | — |



Downloaded from Arrow.com.

| Module Abbreviation | Register Abbreviation | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|
| ICU | IRQCR14 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | IRQCR15 | — | — | — | — | IRQMD[1:0] | | — | — |
| ICU | SSIER | SSI15 | SSI14 | SSI13 | SSI12 | SSI11 | SSI10 | SSI9 | SSI8 |
| | | SSI7 | SSI6 | SSI5 | SSI4 | SSI3 | SSI2 | SSI1 | SS 0 |
| ICU | NMIER | — | — | — | — | — | — | — | NMIEN |
| ICU | NMICR | — | — | — | — | NMIMD | — | — | — |
| ICU | NMISR | — | — | — | — | — | — | — | NMIST |
| ICU | NMICLR | — | — | — | — | — | — | — | NMICLR |
| FLASH | FMODR | — | — | — | FRDMD | — | — | — | — |
| FLASH | FASTAT | ROMAE | — | — | CMDLK | DFLAE | — | DFLRPE | DFLWPE |
| FLASH | FAEINT | ROMAEIE | — | — | CMDLKIE | DFLAEIE | — | DFLRPEIE | DFLWPEIE |
| FLASH | FRDYIE | — | — | — | — | — | — | — | FRDYIE |
| FLASH | DFLRE | KEY[7 0] | | | | | | | |
| | | — | — | — | — | DBRE3 | DBRE2 | DBRE1 | DBRE0 |
| FLASH | DFLWE | KEY[7 0] | | | | | | | |
| | | — | — | — | — | DBWE3 | DBWE2 | DBWE1 | DBWE0 |
| FLASH | FCURAME | KEY[7 0] | | | | | | | |
| | | — | — | — | — | — | — | — | FCRME |
| FLASH | FSTATR0 | FRDY | ILGLERR | ERSERR | PRGERR | SUSRDY | — | ERSSPD | PRGSPD |
| FLASH | FSTATR1 | FCUERR | — | — | FLOCKST | — | — | — | — |
| FLASH | FENTRYR | FEKEY[7:0] | | | | | | | |
| | | FENTRYD | — | — | — | — | — | FENTRY1 | FENTRY0 |
| FLASH | FPROTR | FPKEY[7:0] | | | | | | | |
| | | — | — | — | — | — | — | — | FPROTCN |
| FLASH | FRESETR | FRKEY[7:0] | | | | | | | |
| | | — | — | — | — | — | — | — | FRESET |
| FLASH | FCMDR | CMDR[7:0] | | | | | | | |
| | | PCMDR[7:0] | | | | | | | |
| FLASH | FCPSR | — | — | — | — | — | — | — | — |
| | | — | — | — | — | — | — | — | ESUSPMD |
| FLASH | DFLBCCNT | — | — | — | BCADR[9:0] | | | | |
| | | BCADR[9:0] | | | | | — | — | BCSIZE |
| FLASH | FPESTAT | — | — | — | — | — | — | — | — |
| | | PEERRST[7:0] | | | | | | | |
| FLASH | DFLBCSTA | — | — | — | — | — | — | — | — |
| | T | — | — | — | — | — | — | — | BCST |
| FLASH | PCKAR | — | — | — | — | — | — | — | — |
| | | PCKA[7:0] | | | | | | | |

Notes: 1. When the same output trigger is specified for pulse output groups 2 and 3 by the PPG0.PCR setting, the PPG0.NDRH address is 000881ECh. When different output triggers are specified, the PPG0.NDRH addresses for pulse output groups 2 and 3 are 000881EEh and 000881ECh, respectively.

2. When the same output trigger is specified for pulse output groups 0 and 1 by the PPG0.PCR setting, the PPG0.NDRL address is 000881EDh. When different output triggers are specified, the PPG0.NDRL addresses for pulse output groups 0 and 1 are 000881EFh and 000881EDh, respectively.

3. When the same output trigger is specified for pulse output groups 6 and 7 by the PPG1.PCR setting, the PPG1.NDRH address is 000881FCh. When different output triggers are specified, the PPG1.NDRH addresses for pulse output groups 6 and 7 are 000881FEh and 000881FCh, respectively.



Downloaded from Arrow.com.

4. When the same output trigger is specified for pulse output groups 4 and 5 by the PPG1.PCR setting, the PPG1.NDRL address is 000881FDh. When different output triggers are specified, the PPG1.NDRL addresses for pulse output groups 4 and 5 are 000881FFh and 000881FDh, respectively.
5. For certain bits, functions differ according to whether the mode is serial communications or smart card interface.



Downloaded from Arrow.com.

# 6. Resets

## 6.1 Overview

There are three types of reset: pin reset, deep software standby reset, and watchdog timer reset. Table 6.1 lists the reset names and sources.

**Table 6.1 Reset Names and Sources**

| Reset Name | Source |
|---|---|
| Pin reset | The RES# pin input voltage is driven low. |
| Deep software standby reset | Deep software standby mode is canceled by an interrupt. |
| Watchdog timer reset | The watchdog timer overflows. |

The internal state and pins of the LSI are initialized by a reset. Figure 6.1 shows the targets to be initialized by each reset.

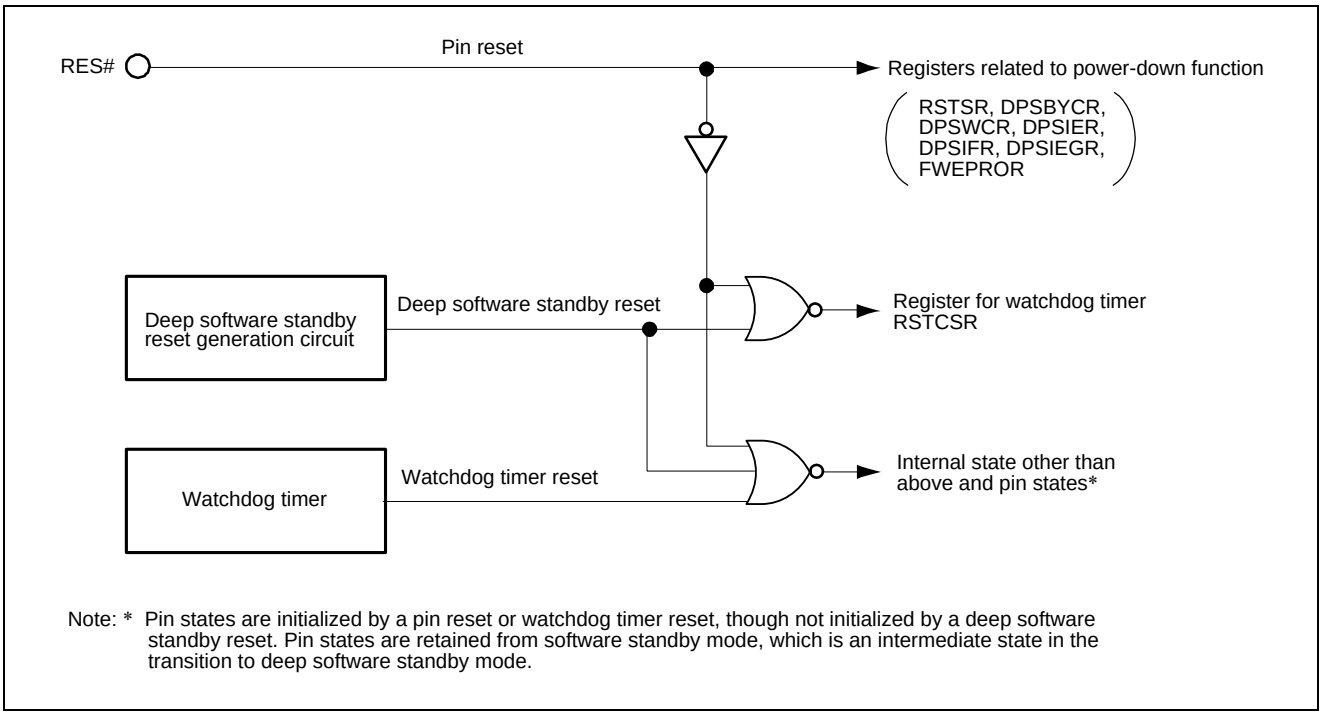



**Figure 6.1 Block Diagram of Reset Circuit**



Downloaded from Arrow.com.

**Table 6.2 Targets to be Initialized by Each Reset Type**

| | Reset Type | | |
|---|---|---|---|
| **Reset Target** | **Pin Reset** | **Deep Software Standby Reset** | **Watchdog Timer Reset** |
| Registers related to the power-down function (RSTSR, DPSBYCR, DPSWCR, DPSIER, DPSIFR, DPSIEGR, FWEPROR) | Reset | — | — |
| Register for the watchdog timer RSTCSR | Reset | Reset | — |
| Registers other than above and internal state | Reset | Reset | Reset |
| Pin states | Reset | — | Reset |

When a reset is released, the reset exception handling is started. For the reset exception handling, see section 9, Exceptions.

Table 6.3 shows the pin related to reset.

**Table 6.3 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| RES# | Input | Reset pin |



Downloaded from Arrow.com.

## 6.2 Register Descriptions

Table 6.4 lists registers related to reset.

**Table 6.4 Registers Related to Reset**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Reset status register | RSTSR | 00h | 0008 C285h | 8 |
| Reset control/status register | RSTCSR | 1Fh | 0008 802Bh | 8 |

### 6.2.1 Reset Status Register (RSTSR)

Address: 0008 C285h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DPSRSTF | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b6 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DPSRSTF | Deep Software Standby Reset Flag | 0: An external interrupt source to cancel the deep software standby reset is not generated<br>1: An external interrupt source to cancel the deep software standby reset is generated | R/(W)* |

Note: * Only 0 can be written to this bit.

RSTSR indicates a source for generating an internal reset.

**DPSRSTF Flag (Deep Software Standby Reset Flag)**

The DPSRSTF flag indicates that deep software standby mode is canceled by an external interrupt source specified with the deep standby interrupt enable register (DPSIER) or the deep standby interrupt flag register (DPSIFR) and an internal reset is generated.

The DPSRSTF flag is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that is used to cancel deep software standby mode.

[Setting condition]

- When deep software standby mode is canceled by an external interrupt source

[Clearing condition]

- When this bit is read as 1 and then written by 0



Downloaded from Arrow.com.

### 6.2.2 Reset Control/Status Register (RSTCSR)

Address: 0008 802Bh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | WOVF | RSTE | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b4 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b5 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b6 | RSTE | Reset Enable | 0: The LSI is not reset internally when TCNT overflows in watchdog timer mode. (TCNT and TCSR of the WDT are reset.)<br>1: The LSI is internally reset when TCNT overflows in watchdog timer mode. | R/W |
| b7 | WOVF | Watchdog Timer Overflow Flag | 0: TCNT has not overflowed in watchdog timer mode.<br>1: TCNT has overflowed in watchdog timer mode. | R/(W)* |

Note: * Only 0 can be written to this bit.

RSTCSR controls the generation of the internal reset signal when TCNT overflows, and selects the type of internal reset signal.
RSTCSR is initialized to 1Fh by a reset signal from the RES# pin or a deep software standby reset, but not by the WDT internal reset signal caused by a WDT overflow.
To read this register, use 8-bit access.
To write to this register, write data in WINB in 16 bits.
For details, see section 19.5.1, Notes on Register Access.

#### RSTE Bit (Reset Enable)

Selects whether or not this LSI is internally reset when TCNT overflows in watchdog timer mode.

#### WOVF Flag (Watchdog Timer Overflow)

Indicates that TCNT overflows in watchdog timer mode. This bit cannot be set to 1 in interval timer mode.

[Setting condition]

- When TCNT overflows (changed form FFh to 00h) in watchdog timer mode

[Clearing condition]

- Reading RSTCSR when WOVF = 1, and then writing 0 to WOVF



Downloaded from Arrow.com.

## 6.3 Operation

### 6.3.1 Pin Reset

This is a reset generated by the RES# pin.

When the RES# pin is driven low, all the processing in progress is terminated and this LSI enters a reset state.

To reset this LSI without fail, the specified oscillation settling time should be observed at power-on and then the RES# pin should be held low. While this LSI is working, the RES# pin should be held low while observing the specified reset pulse width. For details, see section 28, Electrical Characteristics.

### 6.3.2 Deep Software Standby Reset

This is an internal reset generated when deep software standby mode is canceled by an interrupt.

When deep software standby mode is canceled, clock oscillation starts and a deep software standby reset is generated at the same time. After the time specified with the deep software standby wait time bits (WTSTS[5:0] in DPSWCR) has passed, the deep software standby reset is released.

For details on the deep software standby reset, see section 8, Low Power Consumption.

### 6.3.3 Watchdog Timer Reset

This is an internal reset generated by the watchdog timer.

While the RSTE bit in RSTCSR is set to 1, a watchdog timer reset is generated by a watchdog timer overflow. After a predetermined time has passed, the watchdog timer reset is released.

For details on the watchdog timer reset, see section 19, Watchdog Timer (WDT).



Downloaded from Arrow.com.

## 6.4 Determining Reset Generation Source

Reading RSTCSR and RSTSR allows the LSI to determine which reset was used to execute the reset exception handling. Figure 6.2 shows an example of flow to identify a reset generation source.

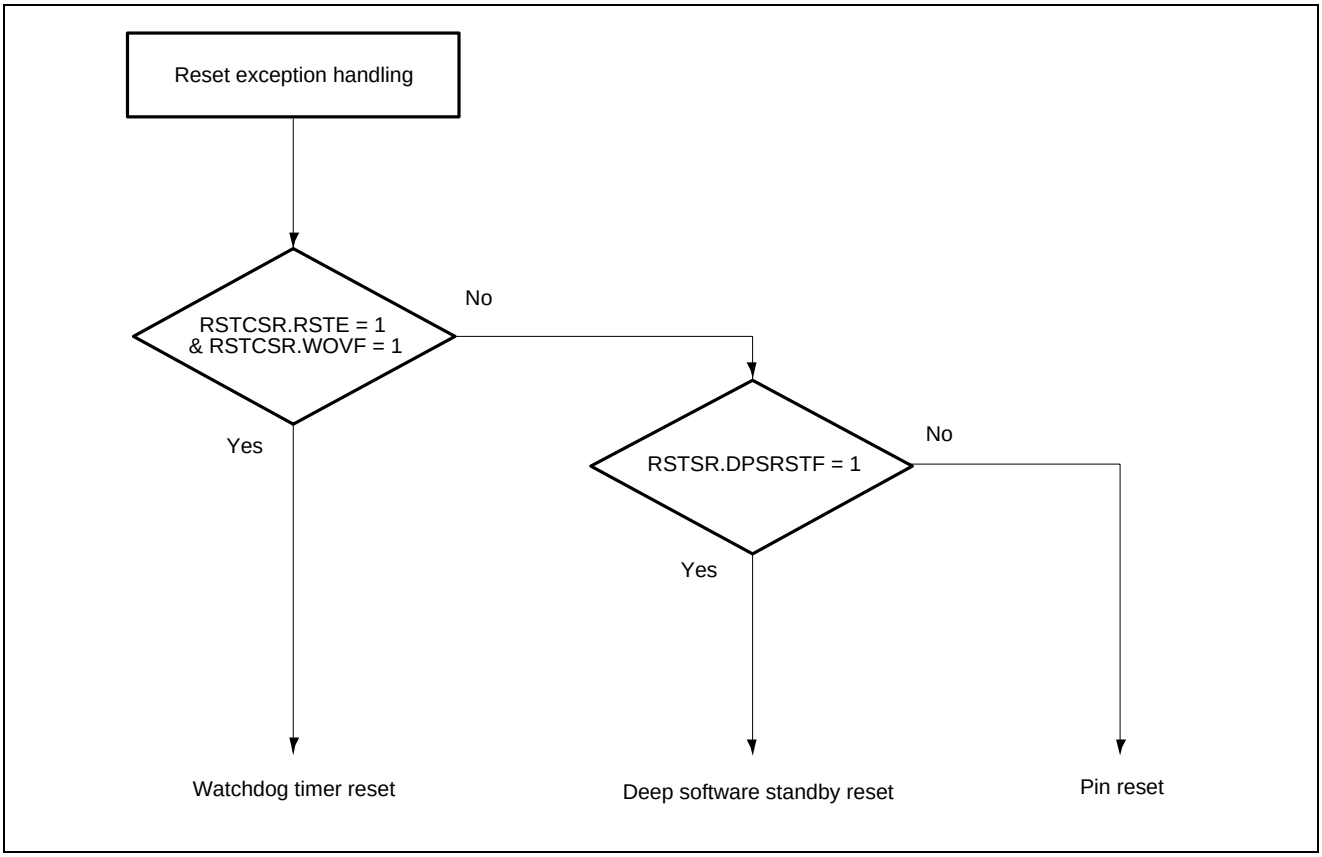



**Figure 6.2 Example of Flow to Identify a Reset Generation Source**

## 6.5 Usage Notes

### 6.5.1 Notes on Design of Board

The XTAL pin and the reset pin are crossly arranged on the RX610 Group. Therefore, to avoid the reference from the clock signal, the reset signal should be guarded by GND.



Downloaded from Arrow.com.

# 7. Clock Generation Circuit

## 7.1 Overview

The RX610 Group has a clock generation circuit that generates the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK).

The clock generation circuit consists of a main clock oscillator, phase-locked loop (PLL) circuit, frequency divider, and selector circuit.

Table 7.1 lists the specifications of the clock generation circuit. Figure 7.1 shows a block diagram of the clock generation circuit.

**Table 7.1 Specifications of Clock Generation Circuit**

| Item | Specification |
|---|---|
| Use | • Generates the system clock (ICLK) to be supplied to the CPU, DTC, DMAC, ROM, and RAM.<br>• Generates the peripheral module clock (PCLK) to be supplied to peripheral modules.<br>• Generates the external bus clock (BCLK) to be supplied to the external bus. |
| Input clock (EXTAL) frequency | 8 to 14 MHz |
| Selection of ICLK, PCLK, or BCLK | The ICLK, PCLK, or BCLK is selectable independently from EXTAL ×8, ×4, ×2, and ×1. |
| Operating frequency | ICLK: 8 to 100 MHz<br>PCLK: 8 to 50 MHz<br>BCLK: 8 to 25 MHz<br>Restrictions for setting clock frequencies: ICLK ≥ PCLK and ICLK ≥ BCLK |
| Connectable resonator or additional circuit | Crystal resonator |
| Pins for connection to the resonator or additional circuit | EXTAL and XTAL |
| BCLK output control function | BCLK output or high-level output is selectable. |

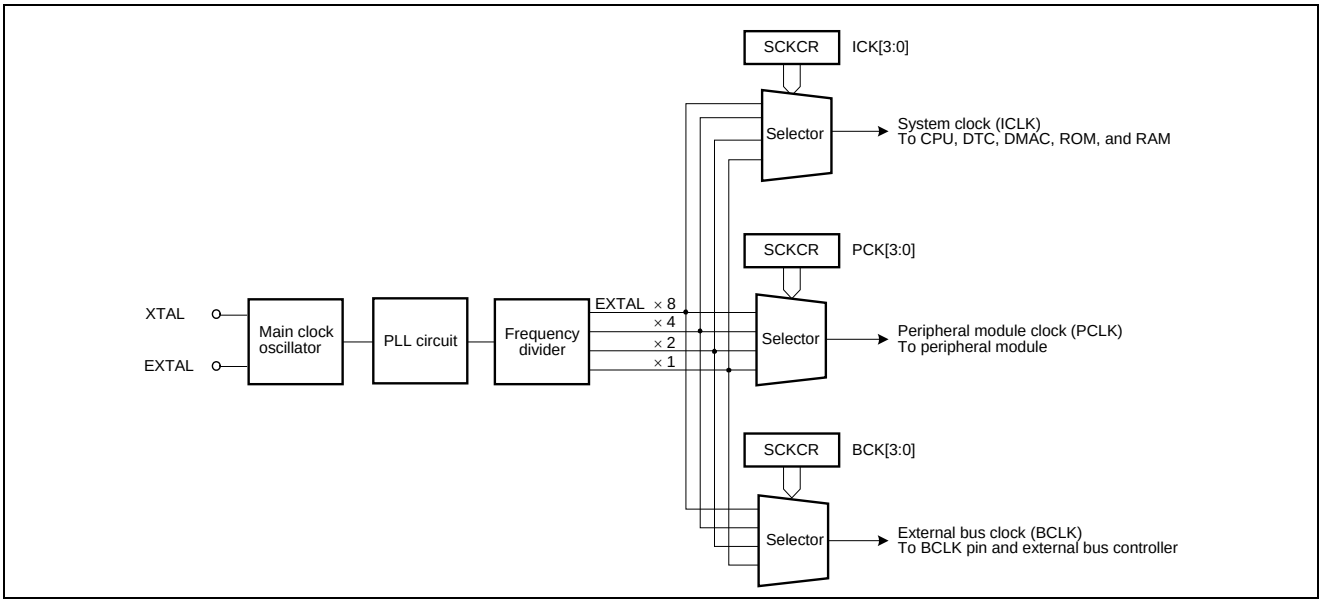



**Figure 7.1 Block Diagram of Clock Generation Circuit**



Downloaded from Arrow.com.

Table 7.2 lists the input/output pins of the clock generation circuit.

**Table 7.2 Pin Configuration**

| Pin Name | I/O | Description |
|---|---|---|
| XTAL | Input | These pins are used to connect a crystal resonator. The EXTAL pin can also be used to input an external clock. For details, see section 7.3.2, External Clock Input. |
| EXTAL | Input | |
| BCLK | Output | This pin is used to supply external devices with the external bus clock (BCLK). |

## 7.2 Register Descriptions

Table 7.3 shows the register of the clock generation circuit.

**Table 7.3 Register of Clock Generation Circuit**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| System clock control register | SCKCR | 0202 0200h | 0008 0020h | 32 |



Downloaded from Arrow.com.

### 7.2.1 System Clock Control Register (SCKCR)

Address: 0008 0020h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | ICK[3:0] | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|
| | PSTOP1 | – | – | – | BCK[3:0] | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | PCK[3:0] | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b11 to b8 | PCK[3:0]*1 | Peripheral Module Clock (PCLK) Select | b11 b8<br>0 0 0 0: ×8<br>0 0 0 1: ×4<br>0 0 1 0: ×2<br>0 0 1 1: ×1<br>Settings other than above are prohibited. | R/W |
| b15 to b12 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b19 to b16 | BCK[3:0]*1 | External Bus Clock (BCLK) Select | b19 b16<br>0 0 0 0: ×8<br>0 0 0 1: ×4<br>0 0 1 0: ×2<br>0 0 1 1: ×1<br>Settings other than above are prohibited. | R/W |
| b22 to b20 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b23 | PSTOP1 | BCLK Output Stop | 0: BCLK output<br>1: Fixed high | R/W |
| b27 to b24 | ICK[3:0]*2 | System Clock (ICLK) Select | b27 b24<br>0 0 0 0: ×8<br>0 0 0 1: ×4<br>0 0 1 0: ×2<br>0 0 1 1: ×1<br>Settings other than above are prohibited. | R/W |
| b31 to b28 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

Notes: 1. Do not set a frequency higher than the system clock (ICLK). If such a frequency is set, the clock frequency will be the same as the ICLK.

2. Do not set a frequency lower than the peripheral module clock (PCLK) and external bus clock (BCLK). If such a frequency is set, the frequency of the PCLK and BCLK will change to the system clock (ICLK) frequency.

The SCKCR register is used to control the BCLK output and select the frequencies of the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK).

### PCK[3:0] Bits (Peripheral Module Clock (PCLK) Select)

These bits select the PCLK frequency.

The value of these bits indicates a multiplication factor of the input clock (EXTAL).

### BCK[3:0] Bits (External Bus Clock (BCLK) Select)

These bits select the BCLK frequency.

The value of these bits indicates a multiplication factor of the input clock (EXTAL).

### PSTOP1 Bit (BCLK Output Stop)

This bit controls the BCLK output from P53.

### ICK[3:0] Bits (System Clock (ICLK) Select)

These bits select the frequency of the CPU, DMAC, DTC, and system clock (ICLK).

The value of these bits indicates a multiplication factor of the input clock (EXTAL).



Downloaded from Arrow.com.

## 7.3 Main Clock Oscillator

Clock pulses can be supplied by connecting a crystal resonator or by inputting an external clock.

### 7.3.1 Connecting a Crystal Resonator

Figure 7.2 shows an example of connecting a crystal resonator. Table 7.4 shows reference values of the damping resistance (Rd).

When supplying the clock from the crystal resonator, the frequency of the resonator should be in the range of 8 to 14 MHz.

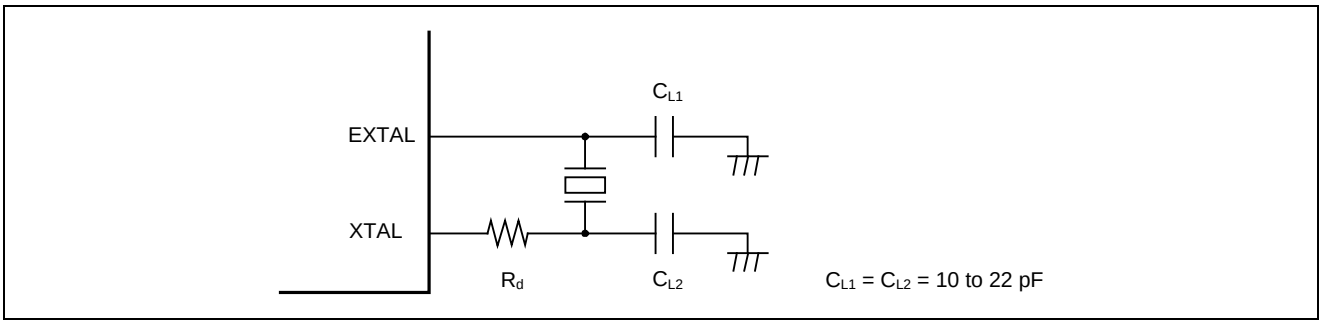



**Figure 7.2 Example of Crystal Resonator Connection**

**Table 7.4 Damping Resistance (Reference Values)**

| Frequency (MHz) | 8 | 10 | 12 | 14 |
|---|---|---|---|---|
| Rd (Ω) | 200 | 100 | 0 | 0 |

Figure 7.3 shows an equivalent circuit of the crystal resonator. Use a crystal resonator that has the characteristics shown in table 7.5.

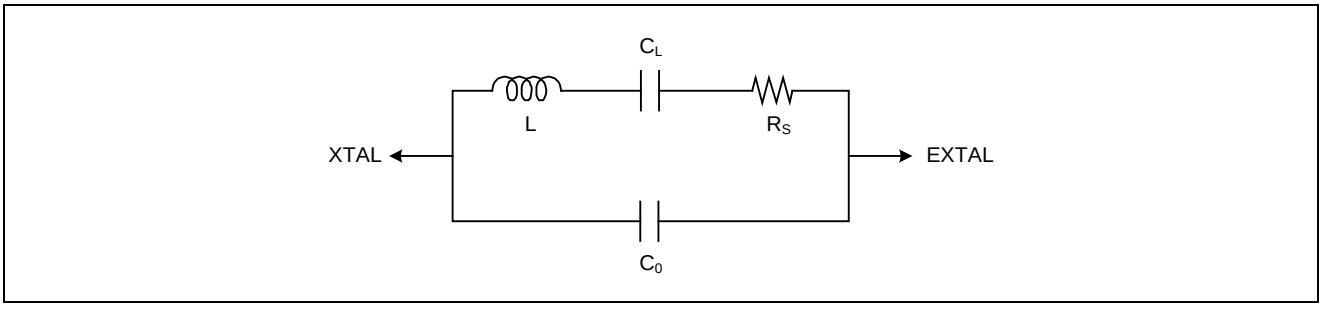



**Figure 7.3 Equivalent Circuit of Crystal Resonator**

**Table 7.5 Crystal Resonator Characteristics (Reference Values)**

| Frequency (MHz) | 8 | 10 | 12 | 14 |
|---|---|---|---|---|
| Rs max (Ω) | 80 | 70 | 60 | 50 |
| C0 max (pF) | 7 | | | |



Downloaded from Arrow.com.

### 7.3.2 External Clock Input

Figure 7.4 shows examples of external clock input. To leave the XTAL pin open, make the parasitic capacitance less than 10 pF. When the counter-phase clock is input to the XTAL pin, hold the external clock in high level during standby mode.

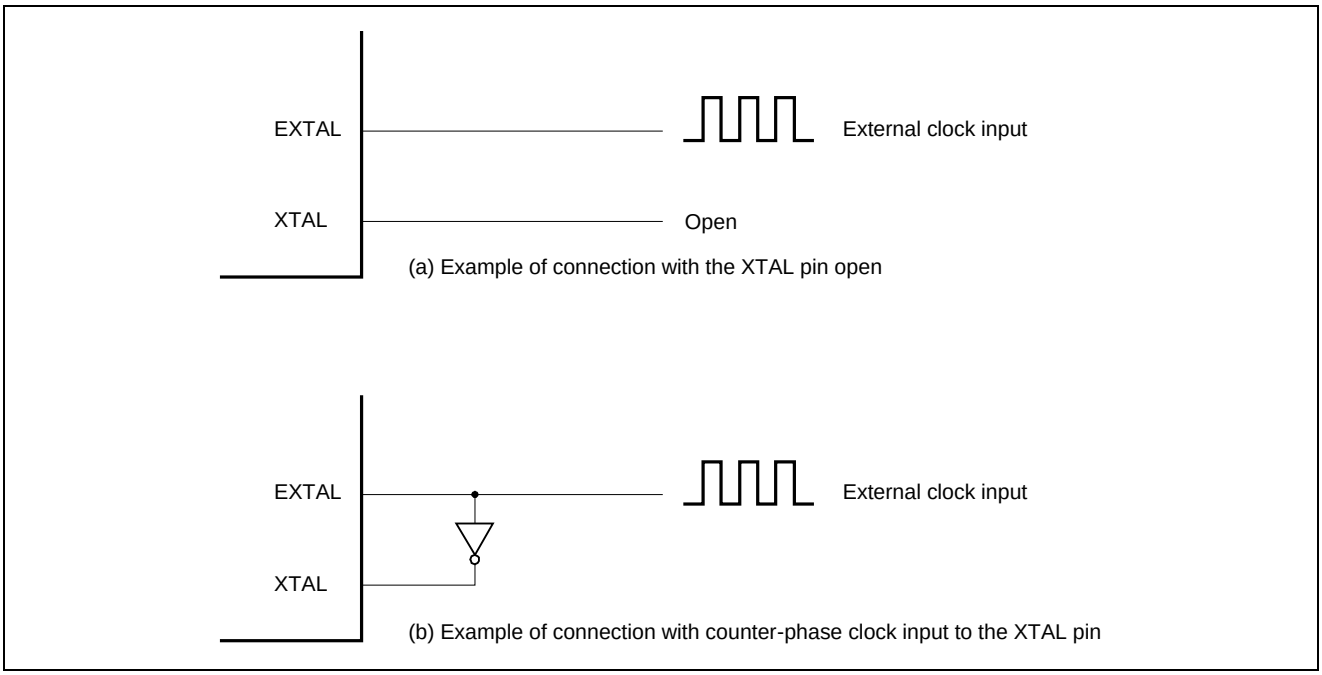



**Figure 7.4 Examples of External Clock Input**

## 7.4 PLL Circuit

The PLL circuit has a function to multiply the frequency from the oscillator by a factor of up to 8.

## 7.5 Frequency Divider

The frequency divider divides the PLL clock to generate 1/2, 1/4, or 1/8 clock. After the ICK[3:0], PCK[3:0], and BCK[3:0] bits in SCKCR are updated, this LSI operates with the updated frequencies.



Downloaded from Arrow.com.

## 7.6 Internal Clock

The internal clock is generated by multiplying the external input clock (EXTAL) by 8 with the PLL circuit, and then by dividing the multiplied clock by 1, 2, 4, or 8 with the frequency divider.

There are following three types of internal clock.

- Operating clock of the CPU, DMAC, and DTC: System clock (ICLK)
- Operating clock of peripheral modules: Peripheral module clock (PCLK)
- Clock for the external bus controller and external pin output: External bus clock (BCLK)

The frequencies are set by bits ICK[3:0], PCK[3:0], and BCK[3:0] in SCKCR, respectively.

### 7.6.1 System Clock (ICLK)

The system clock (ICLK) is used as the operating clock of the CPU, DMAC, DTC, ROM, and RAM.

The ICLK frequency is specified by the ICK[3:0] bits in SCKCR.

A frequency lower than the peripheral module clock (PCLK) and external bus clock (BCLK) should not be set for the ICLK. If such a frequency is set, the frequency of ICLS will be the same as that of the PCLK or BCLK.

### 7.6.2 Peripheral Module Clock (PCLK)

The peripheral module clock (PCLK) is the operating clock for peripheral modules.

The PCLK frequency is specified by the PCK[3:0] bits in SCKCR.

A frequency higher than the system clock (ICLK) should not be set for the PCLK. If such a frequency is set, the clock frequency will be the same as the ICLK.

### 7.6.3 External Bus Clock (BCLK)

The external bus clock (BCLK) is output on an external pin as the clock signal for the external connection bus.

Setting the PSTOP1 bit in SCKCR to 0 and the B3 bit in DDR for port 5 (P5.DDR) to 1 selects output of the BCLK on the BCLK output pin. However, note that the order of setting for these bits requires care. Specifically, change the value of the B3 bit in P5.DDR while the value of the PSTOP1 bit in SCKCR is 1.

The BCLK frequency is specified by the BCK[3:0] bits in SCKCR.

A frequency higher than the system clock (ICLK) should not be set for the BCLK. If such a frequency is set, the clock frequency will be the same as the ICLK.



Downloaded from Arrow.com.

## 7.7 Usage Notes

### 7.7.1 Notes on the Clock Generation Circuit

1. The frequencies of the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK) supplied to each module are selected according to the setting of SCKCR. Select each frequency that is within the operation guaranteed range of clock cycle time (tcyc) specified in AC characteristics of electrical characteristics. Each frequency should meet the following:

   ICLK = 8 to 100 MHz, PCLK = 8 to 50 MHz, BCLK = 8 to 25 MHz

   All peripheral modules (except for the DMAC and DTC) operate on the PCLK. Note therefore that the operating speed of modules such as the timer and SCI varies before and after the frequency is changed.
   In addition, the waiting time for canceling software standby mode varies with the frequency change. For details, see section 8.5.3.3, Setting Oscillation Settling Time after Software Standby Mode is Canceled.

2. The relationship among the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK) is ICLK ≥ PCLK and ICLK ≥ BCLK, and the ICLK has the highest priority. For this reason, if a setting that does not meet these conditions is made, the PCLK and BCLK may have the clock frequency set by the ICK[3:0] bits in SCKCR regardless of the settings of the PCK[3:0] and BCK[3:0] bits in SCKCR.

3. Note that when changing a clock frequency, it may change during an access to the external bus.

4. After writing to the SCKCR, further writing to the same register before completion of the change in frequency is ignored. In the case of continued writing to the SCKCR, confirm that values read from the SCKCR are actually the most recently written values.

5. After writing to the SCKCR, transitions to software standby mode are prohibited until completion of the change in frequency. Subsequent operation is not guaranteed if a transition to software standby mode is attempted while the frequency is being changed. The interval between writing to the SCKCR and issuing of the WAIT instruction must take up at least 11 cycles of the system clock.



Downloaded from Arrow.com.

### 7.7.2 Notes on Resonator

Since various resonator characteristics relate closely to the user's board design, adequate evaluation is required on the user side before use, referencing the resonator connection example shown in this section. The circuit constants for the resonator depend on the resonator to be used and the stray capacitance of the mounting circuit. Therefore, the circuit constants should be determined in full consultation with the resonator manufacturer. The voltage to be applied between the resonator pins must be within the absolute maximum rating.

### 7.7.3 Notes on Board Design

When using a crystal resonator, place the resonator and its capacitors as close to the XTAL and EXTAL pins as possible. Other signal lines should be routed away from the oscillation circuit as shown in figure 7.5 to prevent induction from interfering with correct oscillation.

The XTAL pin and the reset pin are crossly arranged on the RX610 Group. Therefore, to avoid the reference from the clock signal, the reset signal should be guarded by GND.

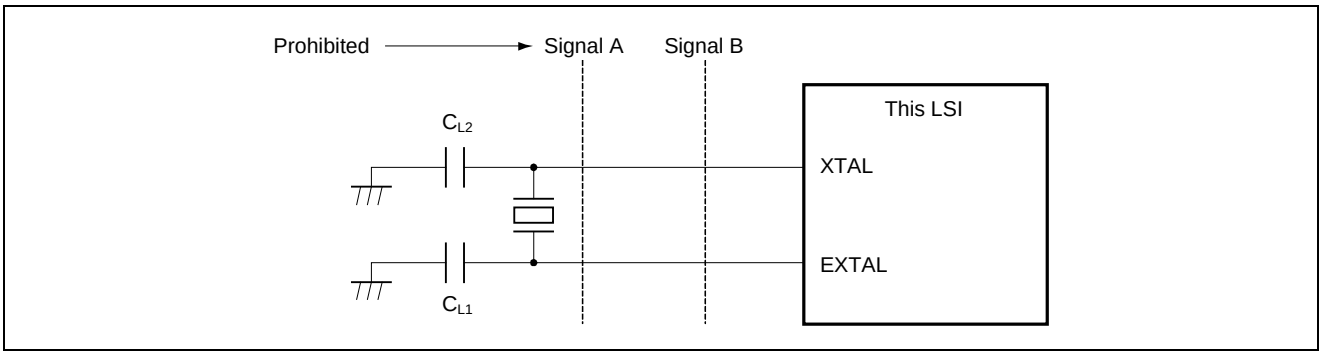



**Figure 7.5 Notes on Board Design for Oscillation Circuit**

Figure 7.6 shows a recommended external circuit for the PLL circuit. Separate pins PLLVcc, PLLVss, Vcc, and Vss from the board power supply source, and be sure to insert bypass capacitors CPB and CB close to the pins.

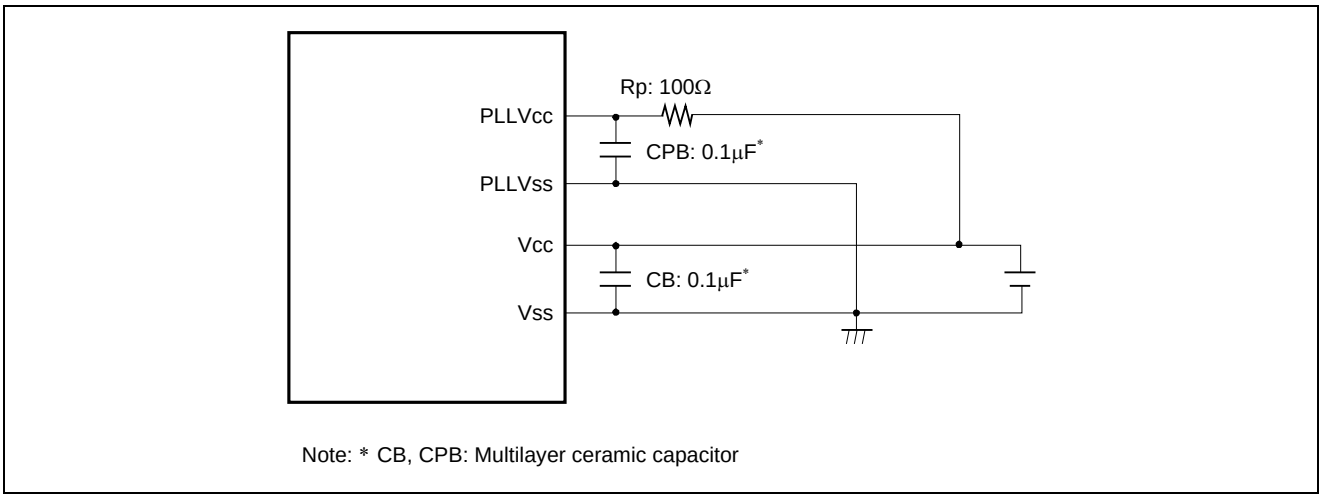



**Figure 7.6 Recommended External Circuit for PLL Circuit**



Downloaded from Arrow.com.

# 8. Low Power Consumption

## 8.1 Overview

The RX610 Group has functions to reduce power consumption, including a multi-clock function, BCLK output stop function, module stop function, and a function for transition to low power consumption mode.

Table 8.1 lists the specifications of low power consumption. Table 8.2 shows the conditions to shift to low power consumption mode, states of the CPU and peripheral modules, and mode canceling method.
After the reset state, this LSI enters the normal program execution state, but modules except the DTC and DMAC do not operate.

**Table 8.1 Specifications of Low Power Consumption**

| Item | Description |
|---|---|
| Multi-clock function | The frequency division ratio is settable independently for the system clock (ICLK), peripheral module clock (PCLK), and external bus clock (BCLK). |
| BCLK output stop function | BCLK output and high-output are selectable. |
| Module stop function | Functions can be stopped for each peripheral module. |
| Function for transition to low power consumption mode | Transition to low power consumption mode is enabled to stop the CPU, peripheral modules, and oscillator. |
| Four low power consumption modes | Sleep mode<br>All-module clock stop mode<br>Software standby mode<br>Deep software standby mode |



Downloaded from Arrow.com.

**Table 8.2 Transition and Cancellation of the Mode and the State of Operation**

| Transition and Cancellation of the Mode and the State of Operation | Sleep Mode | All-Module Clock Stop Mode | Software Standby Mode | Deep Software Standby Mode |
|---|---|---|---|---|
| Transition method | Control register + instruction | Control register + instruction | Control register + instruction | Control register + instruction |
| Canceling method | Interrupt | Interrupt*1 | External interrupt | External interrupt*2 |
| State after cancellation*3 | Program execution state (interrupt processing) | Program execution state (interrupt processing) | Program execution state (interrupt processing) | Program execution state (reset processing) |
| Oscillator | Operating | Operating | Stopped | Stopped |
| CPU | Stopped (Retained) | Stopped (Retained) | Stopped (Retained) | Stopped (Undefined) |
| On-chip RAM 1 (0001 0000h to 0001 FFFFh) | Operating (Retained) | Stopped (Retained) | Stopped (Retained) | Stopped (Undefined) |
| On-chip RAM 0 (0000 0000h to 0000 FFFFh) | Operating (Retained) | Stopped (Retained) | Stopped (Retained) | Stopped (Retained/ Undefined)*4 |
| Watchdog timer | Operating | Operating | Stopped (Retained) | Stopped (Undefined) |
| 8-bit timer (unit 0, unit 1) | Operating | Operating*5 | Stopped (Retained) | Stopped (Undefined) |
| Peripheral modules | Operating | Stopped*6 | Stopped*6 | Stopped (Undefined) |
| I/O ports | Operating | Retained*7 | Retained*8 | Retained*8 |

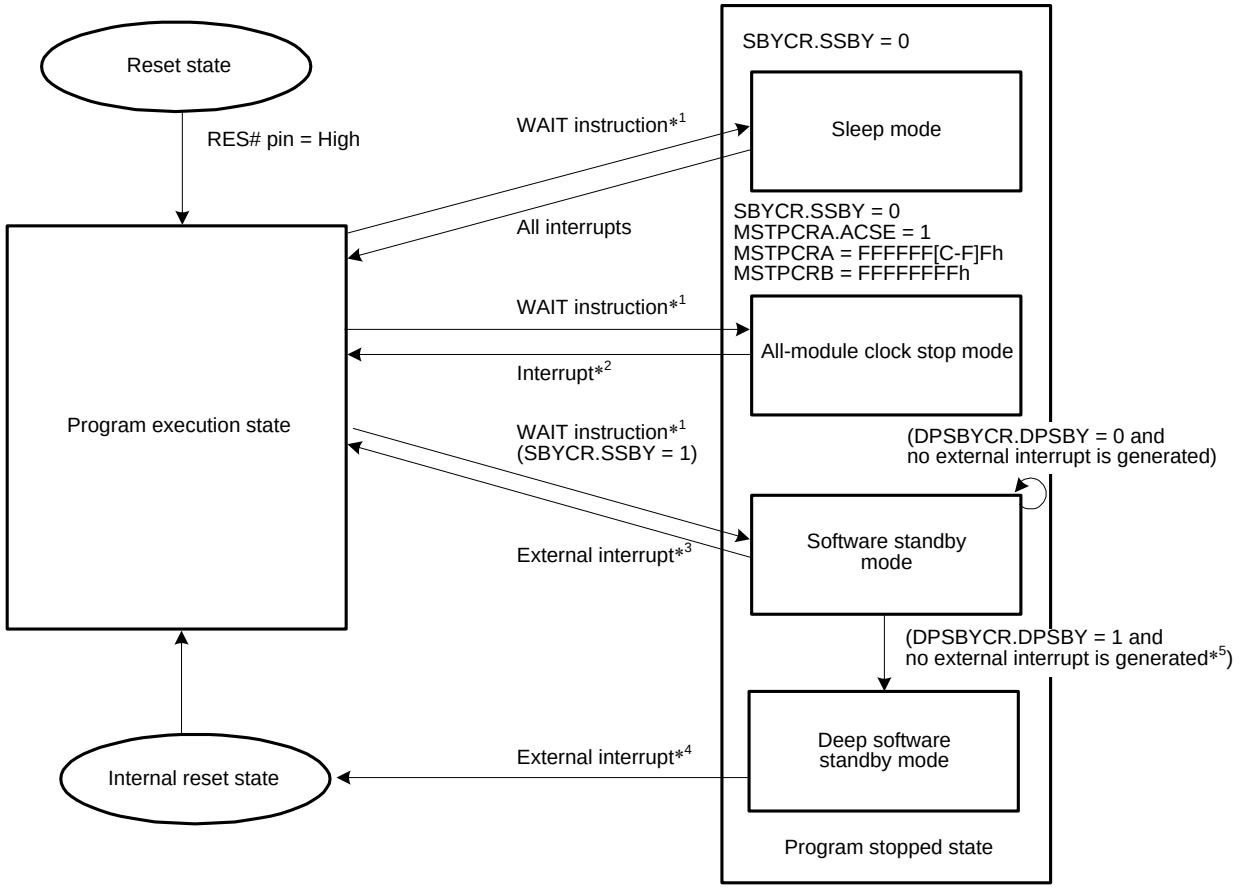
Notes: "Stopped (Retained)" means that internal register values are retained and internal operations are suspended.

"Stopped (Undefined)" means that internal register values are undefined and power is not supplied to the internal circuit.

1. External interrupt and some internal interrupts (8-bit timer and watchdog timer)
2. NMI and only side A of IRQ0 to IRQ3. However, NMI and IRQ are enabled only when the corresponding bit in DPSIER is set to 1.
3. Cancellation by the RES# pin is excluded. When canceled by the RES# pin, this LSI enters the reset state.
4. "Retained" or "Undefined" can be selected by the settings of the on-chip RAM Off 2, on-chip RAM Off 1, and on-chip RAM Off 0 bits (RAMCUT2/RAMCUT1/RAMCUT0) in DPSBYC.
5. "Run" or "stop" can be selected by the settings of the 8-bit timer 3/2 (unit 1) module stop and 8-bit timer 1/0 (unit 0) module stop bits (MSTPA5/MSTPA4) in MSTPCRA.
6. Peripheral modules retain the state.
7. When P53 is set as the BCLK output, the I/O port continues to operate as the BCLK output. For details, see section 8.6, BCLK Output Control.
8. "Retained" or "High impedance" for the address bus and bus control signals (CS0# to CS7#, RD#, WR0#, WR1#, WR#, BC0#, and BC1#) can be selected by the setting of the output port enable bit (OPE) in SBYCR.



Downloaded from Arrow.com.




: Transition after exception handling

Notes:
1. When a WAIT instruction is executed, if a canceling source interrupt is received during a transition to the program stopped state, the processing does not shift to the program stopped state and the interrupt exception handling starts.
2. NMI, IRQ0 to IRQ15, 8-bit timer interrupts, and watchdog timer interrupts.
   However, 8-bit timer interrupts are enabled when the MSTPA5 or MSTPA4 bit in MSTPCRA is 0.
3. NMI and IRQ0 to IRQ15. However, IRQ is enabled only when the corresponding bit in SSIER of the ICU is 1.
4. NMI and side A of IRQ0 to IRQ3
   However, NMI and IRQ are enabled only when the corresponding bit in DPSIER is 1.
5. If a conflict between a transition to deep software standby mode and generation of software standby mode canceling source occurs, a mode transition may be made from software standby mode to program execution state through the interrupt exception handling. In this case, transition to deep software standby mode does not occur.
   For details, see section 8.7, Usage Notes.
6. When an interrupt source listed in note 4. is generated, an internal reset (deep software standby reset) is generated for the time period specified with the DPSWCR.WTSTS[5:0] bits. Along with cancellation of an internal reset, deep software standby mode is cancelled. When a deep software standby mode is cancelled, the program execution state is entered, and reset exception handling is started.

When the RES# pin is driven low in any state, a transition to the reset state is made.

**Figure 8.1 Mode Transitions**



Downloaded from Arrow.com.

## 8.2 Register Descriptions

Table 8.3 is the list of low power consumption registers. For details on the system clock control register (SCKCR), see section 7.2.1, System Clock Control Register (SCKCR).

**Table 8.3 List of Low Power Consumption Registers**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Standby control register | SBYCR | 4F00h | 0008 000Ch | 16 |
| Module stop control register A | MSTPCRA | 67FF FFFFh | 0008 0010h | 32 |
| Module stop control register B | MSTPCRB | FFFF FFFFh | 0008 0014h | 32 |
| Module stop control register C | MSTPCRC | FFFF 0000h | 0008 0018h | 32 |
| Deep standby control register | DPSBYCR | 31h | 0008 C280h | 8 |
| Deep standby wait control register | DPSWCR | 0Fh | 0008 C281h | 8 |
| Deep standby interrupt enable register | DPSIER | 00h | 0008 C282h | 8 |
| Deep standby interrupt flag register | DPSIFR | 00h | 0008 C283h | 8 |
| Deep standby interrupt edge register | DPSIEGR | 00h | 0008 C284h | 8 |
| Reset status register | RSTSR | 00h | 0008 C285h | 8 |
| Deep standby backup register 0 | DPSBKR0 | xxh* | 0008 C290h | 8 |
| Deep standby backup register 1 | DPSBKR1 | xxh* | 0008 C291h | 8 |
| Deep standby backup register 2 | DPSBKR2 | xxh* | 0008 C292h | 8 |
| Deep standby backup register 3 | DPSBKR3 | xxh* | 0008 C293h | 8 |
| Deep standby backup register 4 | DPSBKR4 | xxh* | 0008 C294h | 8 |
| Deep standby backup register 5 | DPSBKR5 | xxh* | 0008 C295h | 8 |
| Deep standby backup register 6 | DPSBKR6 | xxh* | 0008 C296h | 8 |
| Deep standby backup register 7 | DPSBKR7 | xxh* | 0008 C297h | 8 |
| Deep standby backup register 8 | DPSBKR8 | xxh* | 0008 C298h | 8 |
| Deep standby backup register 9 | DPSBKR9 | xxh* | 0008 C299h | 8 |
| Deep standby backup register 10 | DPSBKR10 | xxh* | 0008 C29Ah | 8 |
| Deep standby backup register 11 | DPSBKR11 | xxh* | 0008 C29Bh | 8 |
| Deep standby backup register 12 | DPSBKR12 | xxh* | 0008 C29Ch | 8 |
| Deep standby backup register 13 | DPSBKR13 | xxh* | 0008 C29Dh | 8 |
| Deep standby backup register 14 | DPSBKR14 | xxh* | 0008 C29Eh | 8 |
| Deep standby backup register 15 | DPSBKR15 | xxh* | 0008 C29Fh | 8 |
| Deep standby backup register 16 | DPSBKR16 | xxh* | 0008 C2A0h | 8 |
| Deep standby backup register 17 | DPSBKR17 | xxh* | 0008 C2A1h | 8 |
| Deep standby backup register 18 | DPSBKR18 | xxh* | 0008 C2A2h | 8 |
| Deep standby backup register 19 | DPSBKR19 | xxh* | 0008 C2A3h | 8 |
| Deep standby backup register 20 | DPSBKR20 | xxh* | 0008 C2A4h | 8 |
| Deep standby backup register 21 | DPSBKR21 | xxh* | 0008 C2A5h | 8 |
| Deep standby backup register 22 | DPSBKR22 | xxh* | 0008 C2A6h | 8 |
| Deep standby backup register 23 | DPSBKR23 | xxh* | 0008 C2A7h | 8 |
| Deep standby backup register 24 | DPSBKR24 | xxh* | 0008 C2A8h | 8 |
| Deep standby backup register 25 | DPSBKR25 | xxh* | 0008 C2A9h | 8 |
| Deep standby backup register 26 | DPSBKR26 | xxh* | 0008 C2AAh | 8 |
| Deep standby backup register 27 | DPSBKR27 | xxh* | 0008 C2ABh | 8 |
| Deep standby backup register 28 | DPSBKR28 | xxh* | 0008 C2ACh | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Deep standby backup register 29 | DPSBKR29 | xxh* | 0008 C2ADh | 8 |
| Deep standby backup register 30 | DPSBKR30 | xxh* | 0008 C2AEh | 8 |
| Deep standby backup register 31 | DPSBKR31 | xxh* | 0008 C2AFh | 8 |

Note: * DPSBKR0 to DPSBKR31 are not initialized and their values are undefined immediately after power-on.



Downloaded from Arrow.com.

### 8.2.1 Standby Control Register (SBYCR)

Address: 0008 000Ch

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | OPE | — | | | STS[4:0] | | |
| Value after reset: | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b12 to b8 | STS[4:0] | Standby Timer Select | b12 b8<br>0 0 1 0 1: Waiting time = 64 states<br>0 0 1 1 0: Waiting time = 512 states<br>0 0 1 1 1: Waiting time = 1024 states<br>0 1 0 0 0: Waiting time = 2048 states<br>0 1 0 0 1: Waiting time = 4096 states<br>0 1 0 1 0: Waiting time = 16384 states<br>0 1 0 1 1: Waiting time = 32768 states<br>0 1 1 0 0: Waiting time = 65536 states<br>0 1 1 0 1: Waiting time = 131072 states<br>0 1 1 1 0: Waiting time = 262144 states<br>0 1 1 1 1: Waiting time = 524288 states<br>Settings other than above are prohibited. | R/W |
| b13 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b14 | OPE | Output Port Enable | 0: In software standby mode or deep software standby mode, the address bus and bus control signals are set to the high-impedance state.<br>1: In software standby mode or deep software standby mode, the address bus and bus control signals retain the output state. | R/W |
| b15 | SSBY | Software Standby | 0: Shifts to sleep mode or all-module clock stop mode after the WAIT instruction is executed<br>1: Shifts to software standby mode after the WAIT instruction is executed | R/W |

SBYCR is used to control software standby mode.



Downloaded from Arrow.com.

### STS[4:0] Bits (Standby Timer Select)

These bits select the time for this LSI to wait until the clock is stabilized when software standby mode is canceled by an external interrupt.

In the case of crystal oscillation, see table 8.4 and make a selection according to the operating frequency so that the waiting time is no less than the oscillation settling time. When an external clock is used, the PLL circuit settling time is necessary. Select a waiting time referring to table 8.4.

During the oscillation settling time, the standby timer is counted on the peripheral module clock (PCLK) frequency. Note this in multi-clock mode.

### OPE Bit (Output Port Enable)

The OPE bit specifies whether to retain the output of the address bus and bus control signals (CS0# to CS7#, RD#, WR0#, WR1#, WR#, BC0#, and BC1#) in software standby mode or deep software standby mode, or to set the output to the high-impedance state.

### SSBY Bit (Software Standby)

The SSBY bit specifies the transition destination after the WAIT instruction is executed.

When the SSBY bit is set to 0, the LSI enters either sleep mode or all-module clock stop mode after execution of the WAIT instruction, according to the setting of the MSTPCRA and MSTPCRB registers. When the SSBY bit is set to 1, the LSI enters software standby mode after execution of the WAIT instruction. In this case, when the DPSBY bit in DPSBYCR is 1, the LSI enters deep software standby mode after software standby mode. For details, see section 8.5, Low Power Consumption Modes.

This bit is not cleared to 0 when software standby mode is canceled by an external interrupt and the LSI enters normal mode. Write 0 to this bit to clear.

When the WDT is used in watchdog timer mode, the setting of this bit is invalid and the LSI always enters sleep mode or all-module clock stop mode after the WAIT instruction is executed.



Downloaded from Arrow.com.

### 8.2.2 Module Stop Control Register A (MSTPCRA)

Address: 0008 0010h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 |
|---|---|---|---|---|---|---|---|---|
| | ACSE | — | — | MSTPA28 | MSTPA27 | — | — | — |
| Value after reset: | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

| | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|
| | MSTPA23 | MSTPA22 | MSTPA21 | MSTPA20 | MSTPA19 | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | MSTPA15 | MSTPA14 | MSTPA13 | MSTPA12 | MSTPA11 | MSTPA10 | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | MSTPA5 | MSTPA4 | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b4 | MSTPA4 | 8-Bit Timer 3/2 (Unit 1) Module Stop | Target module: TMR3/TMR2<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b5 | MSTPA5 | 8-Bit Timer 1/0 (Unit 0) Module Stop | Target module: TMR1/TMR0<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b9 to b6 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b10 | MSTPA10 | Programmable Pulse Generator 1 (Unit 1) Module Stop | Target module: PPG1<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b11 | MSTPA11 | Programmable Pulse Generator 0 (Unit 0) Module Stop | Target module: PPG0<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b12 | MSTPA12 | 16-Bit Timer Pulse Unit 1 (Unit 1) Module Stop | Target module: TPU unit 1 (TPU6 to TPU11)<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b13 | MSTPA13 | 16-Bit Timer Pulse Unit 0 (Unit 0) Module Stop | Target module: TPU unit 0 (TPU0 to TPU5)<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b14 | MSTPA14 | Compare Match Timer 1 (Unit 1) Module Stop | Target module: CMT unit 1 (CMT2, CMT3)<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b15 | MSTPA15 | Compare Match Timer 0 (Unit 0) Module Stop | Target module: CMT unit 0 (CMT0, CMT1)<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b18 to b16 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b19 | MSTPA19 | D/A Converter Module Stop | Target module: DA<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b20 | MSTPA20 | A/D Converter (Unit 3) Module Stop | Target module: AD3<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b21 | MSTPA21 | A/D Converter (Unit 2) Module Stop | Target module: AD2<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b22 | MSTPA22 | A/D Converter (Unit 1) Module Stop | Target module: AD1<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b23 | MSTPA23 | A/D Converter (Unit 0) Module Stop | Target module: AD0<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b26 to b24 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b27 | MSTPA27 | Data Transfer Controller Module Stop | Target module: DTC<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b28 | MSTPA28 | DMA Controller Module Stop | Target module: DMAC<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b30, b29 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b31 | ACSE[*1] | All-Module Clock Stop Mode Enable | 0: All-module clock stop mode is disabled<br>1: All-module clock stop mode is enabled | R/W |

MSTPCRA is used to control the module stop state.

### ACSE Bit (All-Module Clock Stop Mode Enable)

The ACSE bit enables or disables all-module clock stop mode for reducing supply current by stopping the bus controller and I/O ports when the CPU executes the WAIT instruction after the module stop state has been specified for all modules[*2] controlled by MSTPCRA and MSTPCRB.

Notes: 1. When the SBYCR.SSBY and MSTPCRA.ACSE bits are both 0, sleep mode is entered after WAIT instruction execution.

2. Whether to stop the 8-bit timers or not can be selected by the MSTPA5 and MSTPA 4 bits.



Downloaded from Arrow.com.

### 8.2.3 Module Stop Control Register B (MSTPCRB)

Address: 0008 0014h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 |
|---|---|---|---|---|---|---|---|---|
| | MSTPB31 | MSTPB30 | MSTPB29 | MSTPB28 | MSTPB27 | MSTPB26 | MSTPB25 | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|
| | MSTPB23 | — | MSTPB21 | MSTPB20 | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b19 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b20 | MSTPB20 | $I^2C$ Bus Interface 1 (Unit 1) Module Stop | Target module: RIIC1<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b21 | MSTPB21 | $I^2C$ Bus Interface 0 (Unit 0) Module Stop | Target module: RIIC0<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b22 | — | Reserved | This bit is always read as 1. The write value should always be 1. | R/W |
| b23 | MSTPB23 | CRC Calculator Module Stop | Target module: CRC<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b24 | — | Reserved | This bit is always read as 1. The write value should always be 1. | R/W |
| b25 | MSTPB25 | Serial Communication Interface 6 Module Stop | Target module: SCI6<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b26 | MSTPB26 | Serial Communication Interface 5 Module Stop | Target module: SCI5<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b27 | MSTPB27 | Serial Communication Interface 4 Module Stop | Target module: SCI4<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b28 | MSTPB28 | Serial Communication Interface 3 Module Stop | Target module: SCI3<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b29 | MSTPB29 | Serial Communication Interface 2 Module Stop | Target module: SCI2<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b30 | MSTPB30 | Serial Communication Interface 1 Module Stop | Target module: SCI1<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |
| b31 | MSTPB31 | Serial Communication Interface 0 Module Stop | Target module: SCI0<br>0: The module stop state is canceled<br>1: Transition to the module stop state is made | R/W |

MSTPCRB is used to control the module stop state.



Downloaded from Arrow.com.

### 8.2.4 Module Stop Control Register C (MSTPCRC)

Address: 0008 0018h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MSTPC1 | MSTPC0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | MSTPC0* | RAM0 Module Stop | Target module: RAM0 (0000 0000h to 0000 FFFFh)<br>0: RAM0 run<br>1: RAM0 stop | R/W |
| b1 | MSTPC1* | RAM1 Module Stop | Target module: RAM1 (0001 0000h to 0001 FFFFh)<br>0: RAM1 run<br>1: RAM1 stop | R/W |
| b15 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b31 to b16 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |

Note: * The MSTPC0 or MSTPC1 bit should not be set to 1 during access to the on-chip RAM0 or RAM1. The on-chip RAM 0 or RAM 1 should not be accessed with the MSTPC0 or MSTPC1 bit set to 1.

MSTPCRC is used to control the module stop state.



Downloaded from Arrow.com.

### 8.2.5 Deep Standby Control Register (DPSBYCR)

Address: 0008 C280h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DPSBY | IOKEEP | RAMCUT2 | RAMCUT1 | – | – | – | RAMCUT0 |
| Value after reset: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | RAMCUT0 | On-Chip RAM Off 0 | b5 b4 b0<br>0 0 0: Power is supplied to the on-chip RAM (RAM0*) in deep software standby mode<br>1 1 1: Power is not supplied to the on-chip RAM (RAM0*) in deep software standby mode<br>Settings other than above are prohibited. | R/W |
| b3 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4 | RAMCUT1 | On-Chip RAM Off 1 | See the description of the RAMCUT0 bit | R/W |
| b5 | RAMCUT2 | On-Chip RAM Off 2 | See the description of the RAMCUT0 bit | R/W |
| b6 | IOKEEP | I/O Port Retention | 0: Deep software standby mode and I/O port retention are canceled simultaneously<br>1: I/O port retention is canceled when 0 is written to the IOKEEP bit after deep software standby mode is canceled | R/W |
| b7 | DPSBY | Deep Software Standby | SSBY b7<br>0 0: Transition to sleep mode is made after the WAIT instruction is executed<br>0 1: Transition to sleep mode is made after the WAIT instruction is executed<br>1 0: Transition to software standby mode is made after the WAIT instruction is executed<br>1 1: Transition to deep software standby mode is made after the WAIT instruction is executed | R/W |

Note: * For the on-chip RAM address space, see table 8.2.

DPSBYCR is used to control deep software standby mode.

DPSBYCR is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.

#### RAMCUTj Bits (On-Chip RAM Off j) (j = 0 to 2)

These bits control the internal power supply to the on-chip RAM modules in deep software standby mode.

The on-chip RAM address space is divided into the RAM 0 area and RAM 1 area. For the on-chip RAM address space, see table 8.2.

Only the internal power supply of RAM0 can be controlled by the setting of bits RAMCUT0, RAMCUT1, and RAMCUT2.

The internal power supply of RAM1 is stopped in deep software standby mode regardless of the setting of bits RAMCUT0, RAMCUT1, and RAMCUT2.



Downloaded from Arrow.com.

### IOKEEP Bit (I/O Port Retention)

In deep software standby mode, I/O ports keep retaining the same states from software standby mode. The IOKEEP bit specifies whether to keep retaining the I/O port states from deep software standby mode even after deep software standby mode is canceled, or to cancel retaining the I/O port states.

### DPSBY Bit (Deep Software Standby)

The DPSBY bit controls transitions to deep software standby mode.

When the WAIT instruction is executed while the SSBY and DPSBY bits in SBYCR are 1, the LSI enters deep software standby mode through software standby mode.

This bit is not cleared to 0 when deep software standby mode is canceled by the external interrupt pin. Write 0 to this bit to clear.

When the WDT is used in watchdog timer mode, the setting of this bit is invalid. In this case, even when the SSBY and DPSBY bits in SBYCR are set to 1, the LSI always enters sleep mode or all-module clock stop mode after the WAIT instruction is executed.



Downloaded from Arrow.com.

### 8.2.6 Deep Standby Wait Control Register (DPSWCR)

Address: 0008 C281h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | WTSTS[5:0] | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b5 to b0 | WTSTS[5:0] | Deep Software Standby Waiting Time | b5 b0<br>0 0 0 1 0 1: Waiting time = 64 states<br>0 0 0 1 1 0: Waiting time = 512 states<br>0 0 0 1 1 1: Waiting time = 1024 states<br>0 0 1 0 0 0: Waiting time = 2048 states<br>0 0 1 0 0 1: Waiting time = 4096 states<br>0 0 1 0 1 0: Waiting time = 16384 states<br>0 0 1 0 1 1: Waiting time = 32768 states<br>0 0 1 1 0 0: Waiting time = 65536 states<br>0 0 1 1 0 1: Waiting time = 131072 states<br>0 0 1 1 1 0: Waiting time = 262144 states<br>0 0 1 1 1 1: Waiting time = 524288 states | R/W |
| b7, b6 | — | Reserved | These bits are always read as 0.The write value should always be 0. | R/W |

DPSWCR is used to select the time for this LSI to wait until the clock is stabilized when deep software standby mode is canceled by the external interrupt pin.

DPSWCR is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.

**WTSTS[5:0] Bits (Deep Software Standby Waiting Time)**

These bits select the time for this LSI to wait until the clock is stabilized when deep software standby mode is canceled by the external interrupt pin. When using deep software standby mode, set the WTSTS[5:0] bits before a transition to deep software standby mode is made.

In the case of crystal oscillation, see table 8.5 and make a selection according to the operating frequency so that the waiting time is no less than the oscillation settling time. When an external clock is used, the PLL circuit settling time is necessary. Select a waiting time referring to table 8.5.

During the oscillation settling time, the counter is counted on the EXTAL input clock frequency.



Downloaded from Arrow.com.

### 8.2.7 Deep Standby Interrupt Enable Register (DPSIER)

Address: 0008 C282h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DNMIE | — | — | — | DIRQ3E | DIRQ2E | DIRQ1E | DIRQ0E |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DIRQ0E | IRQ0 Pin Enable | 0: Canceling deep software standby mode by the IRQ0 pin is disabled<br>1: Canceling deep software standby mode by the IRQ0 pin is enabled | R/W |
| b1 | DIRQ1E | IRQ1 Pin Enable | 0: Canceling deep software standby mode by the IRQ1 pin is disabled<br>1: Canceling deep software standby mode by the IRQ1 pin is enabled | R/W |
| b2 | DIRQ2E | IRQ2 Pin Enable | 0: Canceling deep software standby mode by the IRQ2 pin is disabled<br>1: Canceling deep software standby mode by the IRQ2 pin is enabled | R/W |
| b3 | DIRQ3E | IRQ3 Pin Enable | 0: Canceling deep software standby mode by the IRQ3 pin is disabled<br>1: Canceling deep software standby mode by the IRQ3 pin is enabled | R/W |
| b6 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DNMIE | NMI Pin Enable | 0: Canceling deep software standby mode by the NMI pin is disabled<br>1: Canceling deep software standby mode by the NMI pin is enabled | R/(W)* |

Note: * A 1 can be written only once. Once 1 is written to the DNMIE bit, subsequent write accesses are disabled.

DPSIER is used to enable or disable the external interrupt pin that cancels deep software standby mode.

DPSIER is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.

Note that modifying the DPSIER register setting causes a change in the internal state of control over the input buffer of the corresponding pin. At that time, an edge might be generated internally according to the pin state, and if this happens, it leads to setting of the DPSIFR register value to 1. Before causing a shift to deep software standby mode, set the DPSIFR register value to 0.

Also note that the input buffer is disabled on a transition to deep software standby mode if the value of the DPSIER register is 0. At that time, an edge might be generated internally according to the pin state, and if this happens, it leads to setting of the DPSIFR register value to 1. In this case, however, if the DPSIEGR register value is 0, no rising edges are detected and thus the DPSIFR register value will not be 1.



Downloaded from Arrow.com.

## 8.2.8 Deep Standby Interrupt Flag Register (DPSIFR)

Address: 0008 C283h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DNMIF | — | — | — | DIRQ3F | DIRQ2F | DIRQ1F | DIRQ0F |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DIRQ0F | IRQ0 Deep Standby Cancel Flag | 0: No cancel request by the IRQ0 pin is generated<br>1: A cancel request by the IRQ0 pin is generated | R/(W)* |
| b1 | DIRQ1F | IRQ1 Deep Standby Cancel Flag | 0: No cancel request by the IRQ1 pin is generated<br>1: A cancel request by the IRQ1 pin is generated | R/(W)* |
| b2 | DIRQ2F | IRQ2 Deep Standby Cancel Flag | 0: No cancel request by the IRQ2 pin is generated<br>1: A cancel request by the IRQ2 pin is generated | R/(W)* |
| b3 | DIRQ3F | IRQ3 Deep Standby Cancel Flag | 0: No cancel request by the IRQ3 pin is generated<br>1: A cancel request by the IRQ3 pin is generated | R/(W)* |
| b6 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DNMIF | NMI Deep Standby Cancel Flag | 0: No cancel request by the NMI pin is generated<br>1: A cancel request by the NMI pin is generated | R/(W)* |

Note: * Only 0 can be written to this bit.

DPSIFR is used to hold the request for canceling deep software standby mode.

Each flag is set to 1 when a cancel request specified by the deep standby interrupt edge register (DPSIEGR) is generated. Since each flag is set to 1 when a cancel request is generated in any mode, a transition to deep software standby mode should be made after DPSIFR is cleared to 0. Furthermore, changing the corresponding P3.ICR or DPSIER setting may lead to a flag being set to 1. When clearing DPSIFR to 0 after changing the P3.ICR or DPSIER setting, wait for at least 6 cycles of the PCLK before reading DPSIFR and then writing 0 to DPSIFR. For example, reading DPSIER secures at least 6 cycles of the PCLK.

DPSIFR is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.

### DIRQnF Flags (IRQn Deep Standby Cancel Flag) (n = 0 to 3)

These flags indicate that a cancel request by the IRQn pin has been generated.

[Setting condition]

- When a cancel request by the IRQn pin specified by DPSIEGR is generated

[Clearing condition]

- When each bit is read as 1 and then written by 0

### DNMIF Flag (NMI Deep Standby Cancel Flag)

This flag indicates that a cancel request by the NMI pin has been generated.

[Setting condition]

- When a cancel request by the NMI pin specified by DPSIEGR is generated

[Clearing condition]

- When this bit is read as 1 and then written by 0



Downloaded from Arrow.com.

### 8.2.9 Deep Standby Interrupt Edge Register (DPSIEGR)

Address: 0008 C284h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DNMIEG | – | – | – | DIRQ3EG | DIRQ2EG | DIRQ1EG | DIRQ0EG |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DIRQ0EG | IRQ0 Edge Select | 0: A cancel request is generated at a falling edge<br>1: A cancel request is generated at a rising edge | R/W |
| b1 | DIRQ1EG | IRQ1 Edge Select | 0: A cancel request is generated at a falling edge<br>1: A cancel request is generated at a rising edge | R/W |
| b2 | DIRQ2EG | IRQ2 Edge Select | 0: A cancel request is generated at a falling edge<br>1: A cancel request is generated at a rising edge | R/W |
| b3 | DIRQ3EG | IRQ3 Edge Select | 0: A cancel request is generated at a falling edge<br>1: A cancel request is generated at a rising edge | R/W |
| b6 to b4 | — | Reserved | These bits are always read as 0.The write value should always be 0. | R/W |
| b7 | DNMIEG | NMI Edge Select | 0: A cancel request is generated at a falling edge<br>1: A cancel request is generated at a rising edge | R/W |

DPSIEGR is used to select an edge of the cancel signal for canceling deep software standby mode.

DPSIEGR is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.



Downloaded from Arrow.com.

## 8.2.10 Reset Status Register (RSTSR)

Address: 0008 C285h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DPSRSTF | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b6 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DPSRSTF | Deep Software Standby Reset Flag | 0: No deep software standby mode canceling source by an external interrupt is generated<br>1: A deep software standby mode canceling source by an external interrupt is generated | R/(W)* |

Note: * Only 0 can be written to clear the flag.

RSTSR indicates an internal reset generation source.

**DPSRSTF Flag (Deep Software Standby Reset Flag)**

The DPSRSTF flag indicates that deep software standby mode has been canceled by an external interrupt source specified by DPSIER and DPSIEGR and an internal reset has been generated.

This flag is initialized by the reset signal from the RES# pin, but is not initialized by the internal reset signal that cancels deep software standby mode.

[Setting condition]

- When deep software standby mode is canceled by an external interrupt source

[Clearing condition]

- When this bit is read as 1 and then written by 0

## 8.2.11 Deep Standby Backup Register (DPSBKRy) (y = 0 to 31)

Address: 0008 C290h to 0008 C2AFh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | BKUPm7 | BKUPm6 | BKUPm5 | BKUPm4 | BKUPm3 | BKUPm2 | BKUPm1 | BKUPm0 |
| Value after reset: | x | x | x | x | x | x | x | x |

DPSBKRy is an 8-bit readable/writable register to store data during deep software standby mode.

The value of this register is retained even in deep software standby mode where on-chip RAM data is not retained.

DPSBKRy is not initialized and the register value is undefined immediately after power-on.



Downloaded from Arrow.com.

## 8.3 Multi-Clock Function

When the ICK[3:0], BCK[3:0], and PCK[3:0] bits in SCKCR are set, the clock frequency changes.

The CPU and bus masters operate on the operating clock specified by the ICK[3:0] bits. Peripheral modules operate on the operating clock specified by the PCK[3:0] bits. The external bus operates on the operating clock specified by the BCK[3:0] bits. For details, see section 7, Clock Generation Circuit.

## 8.4 Module Stop Function

The module stop function can be set for each on-chip peripheral module.

When the MSTPyj bit (y = A to C, j = 0 to 31) in MSTPCRA to MSTPCRC is set to 1, the specified module stops operating and enters the module stop state, but the CPU continues to operate independently. Clearing the MSTPmi bit to 0 cancels the module stop state, allowing the module to restart operating at the end of the bus cycle.

The internal states of modules are retained in the module stop state.

After a reset, all modules other than the DMAC, DTC, and on-chip RAM are placed in the module stop state. No read/write access can be made to the registers of the module that are in the module stop state.



Downloaded from Arrow.com.

## 8.5 Low Power Consumption Modes

### 8.5.1 Sleep Mode

#### 8.5.1.1 Transition to Sleep Mode

When the WAIT instruction is executed while the SSBY bit in SBYCR is 0, the CPU enters sleep mode.

In sleep mode, the CPU stops operating but the contents of its internal registers are retained. Other peripheral functions do not stop.

To use sleep mode, make the following settings and then execute a WAIT instruction.

1. Clear the PSW.I bit*1 of the CPU to 0.
2. Set the interrupt destination to be used for recovery from sleep mode to the CPU.
3. Set the priority*2 of the interrupt to be used for recovery from sleep mode to a level higher than the setting of the PSW.IPL[2:0] bits*1 of the CPU.
4. Set the IERm.IENj bit*2 for the interrupt to be used for recovery from sleep mode to 1.
5. For the last I/O register to which writing proceeded, read the register to confirm that the value written has been reflected.
6. Execute the WAIT instruction (this automatically sets the PSW.I bit*1 of the CPU to 1).

Notes: 1. For details, see section 2, CPU.
2. For details, see section 10, Interrupt Control Unit (ICU).

#### 8.5.1.2 Canceling Sleep Mode

Sleep mode is canceled by any interrupt, reset signal from the RES# pin, or a reset caused by a watchdog timer overflow.

- Canceling by an interrupt

  When an interrupt occurs, sleep mode is canceled and the interrupt exception handling starts. If a maskable interrupt has been masked by the CPU (the priority level*1 of the interrupt has been set to a value lower than that of the IPL[2:0] bits*2 in PSW of the CPU), sleep mode is not canceled.

Notes: 1. For details, see section 10, Interrupt Control Unit (ICU).
2. For details, see section 2, CPU.

- Canceling by the RES# pin

  When the RES# pin is driven low, the LSI enters the reset state. When the RES# pin is driven high after the reset signal is input for a predetermined time period, the CPU starts the reset exception handling.
- Canceling by a watchdog timer overflow reset

  Sleep mode is canceled by an internal reset generated by a watchdog timer overflow.



Downloaded from Arrow.com.

### 8.5.2 All-Module Clock Stop Mode

#### 8.5.2.1 Transitions to All-Module Clock Stop Mode

When the following two conditions are satisfied, executing the WAIT instruction with the SSBY bit in SBYCR cleared to 0 will cause the transition to all-module clock stop mode at the end of the bus cycle.*[1]

- The ACSE bit in MSTPCRA is set to 1.
- All the modules controlled by the MSTPCRA and MSTPCRB registers except for the 8-bit timers (units 0 and 1) are set in the module stop state (MSTPCRA = FFFFFF[C to F]Fh, MSTPCRB = FFFFFFFFh).

In all-module clock stop mode, the CPU, the bus controller, the I/O ports, and all the peripheral modules except for the 8-bit timers*[2] and the watchdog timers are stopped.

If a further reduction in supply current is required beyond that in all-module clock stop mode, stop the target modules for which operation or stopping is controlled by MSTPCRC.

When all-module clock stop mode is in use, issue a WAIT instruction after making the following settings.

1. Clear the I bit*[3] in PSW of the CPU to 0.
2. Set the priority*[4] of the interrupt to be used for recovery from all-module clock stop mode to a level higher than the setting of the IPL[2:0] bits*[3] in PSW.
3. Set the IENj bit*[4] in IERm for the interrupt to be used for recovery from all-module clock stop mode to 1.
4. Make either of the following settings for interrupts that are not to be used for recovery from all-module clock stop mode.
   - Set the priority*[4] of interrupts*[5] that are not to be used for recovery from all-module clock stop mode to a level lower than the setting of the IPL[2:0] bits*[3] in PSW of the CPU.
   - Set the IENj bit*[4] in IERm for the interrupt*[5] that is not to be used for recovery from all-module clock stop mode to 0.
5. Execute a WAIT instruction (executing a WAIT instruction causes automatic setting of the I bit*[3] in PSW of the CPU to 1).

Notes:
1. The state of DTC or DMAC operations can make transitions to module stop mode impossible. Before setting the MSTPA28 or MSTPA27 bits in MSTPCRA to 1, clear the DMST bit in DMSCNT of the DMAC and the DTCST bit in DTCST of the DTC to 0 so that the DTC or DMAC is not initiated.
2. The MSTPA4 and MSTPA5 bits of the MSTPCRA register select operation or stopping of these modules.
3. For details, see section 2, CPU.
4. For details, see section 10, Interrupt Control Unit (ICU).
5. Executing a WAIT instruction while a peripheral module is operating creates a possibility of recovery from all-module clock-stop mode being triggered by an interrupt that could not normally act as a trigger for recovery. Interrupts that can act as triggers for recovery thus include all interrupts which can be set by the various IERm.IENj bits and the PSW.IPL[2:0] bits, as well as the interrupts that are intended to act as triggers for recovery.



Downloaded from Arrow.com.

#### 8.5.2.2 Release from All-Module Clock Stop Mode

Release from all-module clock stop mode is triggered by an external interrupt (the NMI pin or any pin from among IRQ0 to IRQ15), the signal on the RES# pin, or an internal interrupt (from an 8-bit timer*[1] or the watchdog timer), and normal program execution resumes once handling of the given exception is complete. However, note that in cases where a maskable interrupt has been masked by the CPU (the priority level*[2] of the interrupt has been set to a value lower than that of the IPL[2:0] bits*[3] in PSW of the CPU) or a maskable interrupt has been set up as a trigger for transfer by the DTC or DMAC, the interrupt will not trigger release from all-module clock stop mode.

Notes: 1. The MSTPA4 and MSTPA5 bits of register MSTPCRA select operation or stopping of these modules.
2. For details, see section 10, Interrupt Control Unit (ICU).
3. For details, see section 2, CPU.



Downloaded from Arrow.com.

### 8.5.3 Software Standby Mode

#### 8.5.3.1 Transition to Software Standby Mode

When the WAIT instruction is executed with the SSBY bit in SBYCR set to 1 and the DPSBY bit in DPSBYCR cleared to 0, a transition to software standby mode is made. In this mode, the CPU, on-chip peripheral functions, and all the oscillator functions stop. However, the contents of the CPU internal registers, on-chip RAM data, on-chip peripheral functions, and the states of the I/O ports are retained. Whether the address bus and bus control signals are placed in the high-impedance state or the output state is retained can be specified by the OPE bit in SBYCR. This mode allows significant reduction in power consumption because the oscillator stops in this mode.

When a transition to software standby mode is made, the system clock (ICLK) and peripheral module clock (PCLK) should be operating at the same frequency. If not, be sure to change the system clock (ICLK) or peripheral module clock (PCLK) setting, before executing a WAIT instruction.

In addition, before executing a WAIT instruction, be sure to clear both the DMST bit in DMSCNT of the DMAC and the DTCST bit in DTCST in the DTC to 0.

When the WDT is used in watchdog timer mode, no transition to software standby mode is made. Stop the WDT before executing the WAIT instruction.

When software standby mode is in use, issue a WAIT instruction after making the following settings.

1. Clear the I bit*1 in PSW of the CPU to 0.
2. Set the priority*2 of the interrupt to be used for recovery from software standby mode to a level higher than the setting of the IPL[2:0] bits*1 in PSW.
3. Set the IENj bit*2 in IERm for the interrupt to be used for recovery from software standby mode to 1.
4. Make either of the following settings for interrupts that are not to be used for recovery from software standby mode.
   - Set the priority*2 of interrupts*3 that are not to be used for recovery from software standby mode to a level lower than the setting of the IPL[2:0] bits*1 in PSW of the CPU.
   - Set the IENj bit*2 in IERm for the interrupt*3 that is not to be used for recovery from software standby mode to 0.
5. Execute a WAIT instruction (executing a WAIT instruction causes automatic setting of the I bit*1 in PSW of the CPU to 1).

Notes: 1. For details, see section 2, CPU.
2. For details, see section 10, Interrupt Control Unit (ICU).
3. Executing a WAIT instruction while a peripheral module is operating creates a possibility of recovery from software standby mode being triggered by an interrupt that could not normally act as a trigger for recovery. In addition, the interrupt status flag of an interrupt IRQ0 to IRQ15, which is not specified as a trigger for recovery, might be set in software standby mode, and thus an unexpected exception handling that is not the trigger for recovery will be started.
   Interrupts that can act as triggers for recovery thus includes all interrupts which can be set by the various IERi.IENj bits and the PSW.IPL[2:0] bits, as well as the interrupts that are intended to act as triggers for recovery.



Downloaded from Arrow.com.

### 8.5.3.2 Canceling Software Standby Mode

Software standby mode is canceled by an external interrupt (NMI or IRQ0 to IRQ15*[1]) or the reset signal from the RES# pin.

1. Canceling by an interrupt

   When an NMI or IRQ0 to IRQ15*[1] interrupt request signal is input, clock oscillation starts, and stable clocks are supplied to the entire LSI after the time selected by the STS[4:0] bits in SBYCR has passed, software standby mode is canceled, and the interrupt exception handling is started.

   To cancel software standby mode by an interrupt IRQ0 to IRQ15*[1], set the corresponding enable bit (IENj bit in IERm)*[2] to 1 and mask interrupts with a higher priority than interrupts IRQ0 to IRQ15*[1].

   To cancel software standby mode by an interrupt IRQ0 to IRQ15 for which edge detection is set, 0 should be written to the status flag (the IR bit in IRn of the ICU) of the corresponding interrupt at the beginning of the exception handler for the interrupt.

   In addition, the interrupt status flag of an interrupt IRQ0 to IRQ15, which is not specified as a trigger for recovery, might be set in software standby mode*[3]. Therefore, clear the IR flag after the recovery from software standby mode.

Notes: 1 An interrupt from among IRQ0 to IRQ15 cannot be used as a trigger for release from software standby mode if the corresponding SSIj bit (j = 15 to 0) in SSIER of the ICU has been set to 0. For details, see section 10, Interrupt Control Unit (ICU).

2 For details, see section 10, Interrupt Control Unit (ICU).

3. For details, see section 10.6.2, Return from Software Standby Mode.

2. Canceling by the RES# pin

   When the RES# pin is driven low, clock oscillation starts and at the same time the clocks are supplied to the entire LSI. Be sure to hold the RES# pin low until the clock oscillation settles. When the RES# pin is driven high, the CPU begins the reset exception handling.



Downloaded from Arrow.com.

### 8.5.3.3 Setting Oscillation Settling Time after Software Standby Mode is Canceled

Set the STS[4:0] bits in SBYCR as follows:

1. When using a crystal resonator
   Set the STS[4:0] bits so that the waiting time is no less than the oscillation settling time.
   Table 8.4 shows operating frequencies and waiting time corresponding to each setting of the STS[4:0] bits.
2. When using an external clock
   The PLL circuit settling time is necessary. Set the waiting time referring to table 8.4.

**Table 8.4 Oscillation Settling Time Setting**

| | | | | | Waiting Time | PCLK* (MHz) | | | |
|---|---|---|---|---|---|---|---|---|---|
| STS4 | STS3 | STS2 | STS1 | STS0 | (States) | 50 | 25 | 8 | Unit |
| 0 | 0 | 0 | 0 | 0 | Reserved | — | — | — | μs |
| | | | | 1 | Reserved | — | — | — | |
| | | | 1 | 0 | Reserved | — | — | — | |
| | | | | 1 | Reserved | — | — | — | |
| | | 1 | 0 | 0 | Reserved | — | — | — | |
| | | | | 1 | 64 | 1.3 | 2.6 | 8.0 | |
| | | | 1 | 0 | 512 | 10.25 | 20.5 | 64.0 | |
| | | | | 1 | 1024 | 20.5 | 41.0 | 128.0 | |
| | 1 | 0 | 0 | 0 | 2048 | 40.95 | 81.9 | 256.0 | |
| | | | | 1 | 4096 | 0.08 | 0.16 | 0.51 | ms |
| | | | 1 | 0 | 16384 | 0.33 | 0.66 | 2.05 | |
| | | | | 1 | 32768 | 0.655 | 1.31 | 4.10 | |
| | | 1 | 0 | 0 | 65536 | 1.31 | 2.62 | 8.19 | |
| | | | | 1 | 131072 | 2.62 | 5.24 | 16.38 | |
| | | | 1 | 0 | 262144 | 5.25 | 10.49 | 32.77 | |
| | | | | 1 | 524288 | 10.49 | 20.97 | 65.54 | |
| 1 | x | x | x | x | Reserved | — | — | — | |

: Recommended time setting when an external clock is used

: Recommended time setting when a crystal resonator is used

Note: * The PCLK is the output of the peripheral module frequency divider.
The oscillation settling time (including oscillator's unstable oscillation time) depends on the resonator characteristics.
The PCLK values in this table are reference values.



Downloaded from Arrow.com.

### 8.5.3.4 Example of Software Standby Mode Application

Figure 8.2 shows an example where a transition to software standby mode is made at the falling edge on the IRQ pin, and software standby mode is canceled at the rising edge on the IRQ pin.

In this example, an IRQ interrupt is accepted with the IRQMD[1:0] bits in IRQCRi of the ICU set to 01b (falling edge), and then the IRQMD[1:0] bits are set to 10b (rising edge). After that, the SSIi bit in SSIER of the ICU and the SSBY bit in SBYCR are set to 1, and then the WAIT instruction is executed. Thus a transition to software standby mode is made.

After that, software standby mode is canceled at the rising edge on the IRQ pin.

To return from the software standby mode, settings of the interrupt control unit (ICU) are also necessary. For details, see section 10, Interrupt Control Unit (ICU).

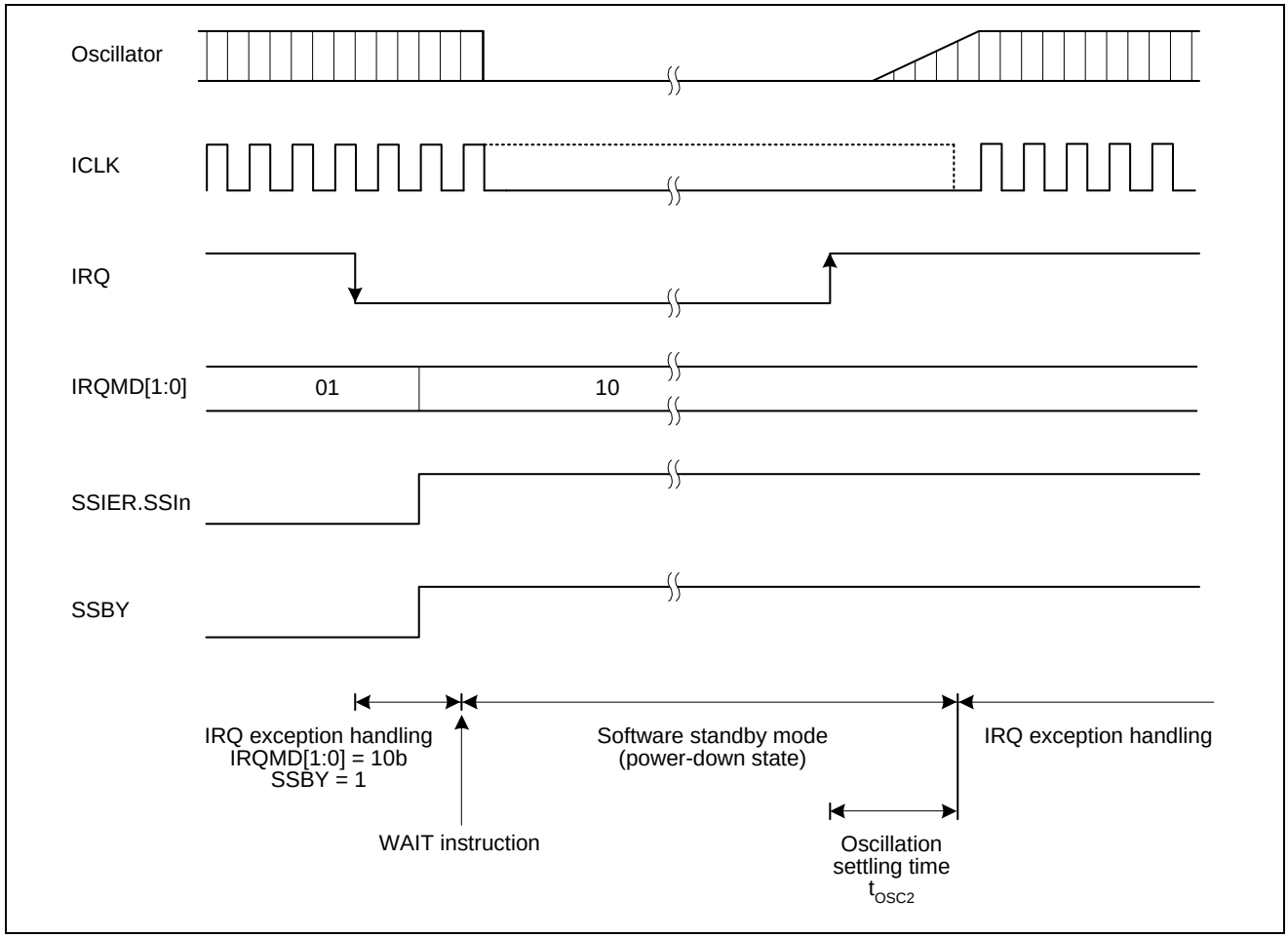



**Figure 8.2 Example of Software Standby Mode Application**



Downloaded from Arrow.com.

### 8.5.4 Deep Software Standby Mode

#### 8.5.4.1 Transition to Deep Software Standby Mode

When the WAIT instruction is executed with the SSBY bit in SBYCR set to 1, a transition to software standby mode*[1] is made. At this time, when the DPSBY bit in DPSBYCR is set to 1, a transition to deep software standby mode is made. However, if a software standby mode canceling source (NMI or IRQ0 to IRQ15) is generated concurrently when a transition to software standby mode is made, software standby mode is canceled regardless of the DPSBY setting, and the interrupt exception handling starts after the oscillation settling time for software standby mode specified by the STS[4:0] bits in SBYCR has passed*[2].

When the SSBY and DPSBY bits are set to 1 and no software standby mode canceling source is generated, a transition to deep software standby mode is made immediately after transition to software standby mode.

In deep software standby mode, the CPU, on-chip peripheral functions, on-chip RAM 1*[3], and all the oscillator functions stop. Furthermore, the internal power supply to these modules stops, allowing significant reduction in power consumption. At this time, the contents of all the registers of the CPU and on-chip peripheral functions become undefined. All the on-chip RAM 1*[3] data becomes undefined regardless of the setting of the RAMCUT2 to RAMCUT0 bits in DPSBYCR.

The on-chip RAM 0*[3] data can be retained by clearing the RAMCUT2 to RAMCUT0 bits to all 0 beforehand. Setting all of these bits to 1 will stop the internal power supply to the on-chip RAM 0*[2], allowing further reduction in power consumption. At this time, the on-chip RAM 0*[3] data is undefined.

The I/O port states remain unchanged from software standby mode.

Notes: 1. Conditions on the DTC, DMAC, and WDT for transition to software standby mode should be met before the WAIT instruction is executed. For details, see section 8.5.3, Software Standby Mode.
2. To cancel software standby mode by an interrupt IRQ0 to IRQ15 for which edge detection is set, 0 should be written to the status flag (the IR bit in IRi of the ICU) of the corresponding interrupt at the beginning of the exception handling routine for the interrupt.
3. The on-chip RAM address space is divided into the RAM 0 area and RAM 1 area. For the on-chip RAM address space, see table 8.2.



Downloaded from Arrow.com.

### 8.5.4.2 Canceling Deep Software Standby Mode

Deep software standby mode is canceled by an external interrupt (NMI or IRQ0-A to IRQ3-A pins) or the reset signal from the RES# pin.

1. Canceling by an external interrupt
   The DPSIFR holds the cancelation cause of deep software standby mode and the bits in this register are set to 1 when the corresponding cancelation requests are generated. When a bit is set to 1 and the corresponding cancelation cause is enabled in the DPSIER register, deep software standby mode is canceled.
   Deep software standby mode is canceled when any of the DNMIF flag in DPSIFR and the DIRQnF flag (n = 3 to 0) in DPSIFR is set to 1. The DNMIF or DIRQnF flag is set to 1 when an edge is generated on the NMI pin or IRQ0-A to IRQ3-A pins enabled by the DNMIE bit in DPSIER or the DIRQnE bit (n = 3 to 0) in DPSIER. Rising edge or falling edge is selectable with DPSIEGR for each pin.
   When a deep software standby mode canceling source is generated, clock oscillation starts and the internal power supply begins at the same time, and then the internal reset signal is generated throughout the LSI. After the time specified by the WTSTS[5:0] bits in DPSWCR has passed, stable clocks are supplied to the entire LSI and the internal reset is released. At the same time, deep software standby mode is canceled and the reset exception handling starts.
   When deep software standby mode is canceled by an external interrupt, the DPSRSTF flag in RSTSR is set to 1.
2. Canceling by the RES# pin
   When the RES# pin is driven low, clock oscillation starts and the internal power supply begins at the same time. Clocks are supplied to the LSI simultaneously with the start of clock oscillation. Be sure to hold the RES# pin low until the clock oscillation settles. When the RES# pin is driven high, the CPU begins the reset exception handling.

### 8.5.4.3 Pin States when Deep Software Standby Mode is Canceled

In deep software standby mode, I/O ports retain the same states from software standby mode. The inside of the LSI is initialized by an internal reset generated when deep software standby mode is canceled. Upon cancellation of deep software standby mode, the reset exception handling starts. The following shows the states of I/O ports at this time.

Whether to initialize the I/O ports or to keep retaining the I/O port states at the time of software standby mode can be selected by the IOKEEP bit in DPSBYCR.

- When IOKEEP = 0
  I/O ports are initialized by an internal reset generated when deep software standby mode is canceled.
- When IOKEEP = 1
  Although the inside of the LSI is initialized by an internal reset generated when deep software standby mode is canceled, I/O ports keep retaining their states from software standby mode regardless of the LSI internal state. At this time, the I/O port states remain unchanged from software standby mode even if settings of I/O ports or peripheral modules are made. Then the retained I/O port states are released by clearing the IOKEEP bit to 0, and the LSI operates according to the internal state.

The IOKEEP bit is not initialized by an internal reset generated when deep software standby mode is canceled.



Downloaded from Arrow.com.

### 8.5.4.4 Setting Oscillation Settling Time after Deep Software Standby Mode is Canceled

Set the WTSTS[5:0] bits in DPSWCR as follows:

1. When using a crystal resonator
   Set the WTSTS[5:0] bits so that the waiting time is no less than the oscillation settling time.
   Table 8.5 shows EXTAL input clock frequencies and waiting time corresponding to each setting of the WTSTS[5:0] bits.
2. When using an external clock
   The PLL circuit settling time is necessary. Set the waiting time referring to table 8.5.

**Table 8.5 Oscillation Settling Time Setting**

| | | | | | | | EXTAL Input Clock Frequency* (MHz) | | |
|---|---|---|---|---|---|---|---|---|---|
| WTSTS5 | WTSTS4 | WTSTS3 | WTSTS2 | WTSTS1 | WTSTS0 | Waiting Time (States) | 12 | 8 | Unit |
| 0 | 0 | 0 | 0 | 0 | 0 | Reserved | — | — | µs |
| | | | | | 1 | Reserved | — | — | |
| | | | | 1 | 0 | Reserved | — | — | |
| | | | | | 1 | Reserved | — | — | |
| | | | 1 | 0 | 0 | Reserved | — | — | |
| | | | | | 1 | 64 | 5.3 | 8.0 | |
| | | | | 1 | 0 | 512 | 42.7 | 64.0 | |
| | | | | | 1 | 1024 | 85.3 | 128.0 | |
| | | 1 | 0 | 0 | 0 | 2048 | 170.7 | 256.0 | |
| | | | | | 1 | 4096 | 0.34 | 0.51 | ms |
| | | | | 1 | 0 | 16384 | 1.37 | 2.05 | |
| | | | | | 1 | 32768 | 2.73 | 4.10 | |
| | | | 1 | 0 | 0 | 65536 | 5.46 | 8.19 | |
| | | | | | 1 | 131072 | 10.92 | 16.38 | |
| | | | | 1 | 0 | 262144 | 21.85 | 32.77 | |
| | | | | | 1 | 524288 | 43.69 | 65.54 | |
| | 1 | × | × | × | × | Reserved | — | — | |
| 1 | × | × | × | × | × | Reserved | — | — | |

: Recommended time setting when an external clock is used

: Recommended time setting when a crystal resonator is used

Note: * The oscillation settling time (including oscillator's unstable oscillation time) depends on the resonator characteristics.
The values of the EXTAL input clock frequency in this table are reference values.



Downloaded from Arrow.com.

### 8.5.4.5 Example of Deep Software Standby Mode Application

Figure 8.3 shows an example where a transition to deep software standby mode is made at the falling edge on the IRQ pin, and deep software standby mode is canceled at the rising edge on the IRQ pin.

In this example, an IRQ interrupt is accepted with the IRQMD[1:0] bits in IRQCRi of the ICU set to 01 (falling edge), and then the DIRQnEG bit in DPSIEGR is set to 1 (rising edge). After that, the SSBY bit in SBYCR and the DPSBY bit in DPSBYCR are set to 1, and then the WAIT instruction is executed. Thus a transition to deep software standby mode is made.

After that, deep software standby mode is canceled at the rising edge on the IRQ pin.

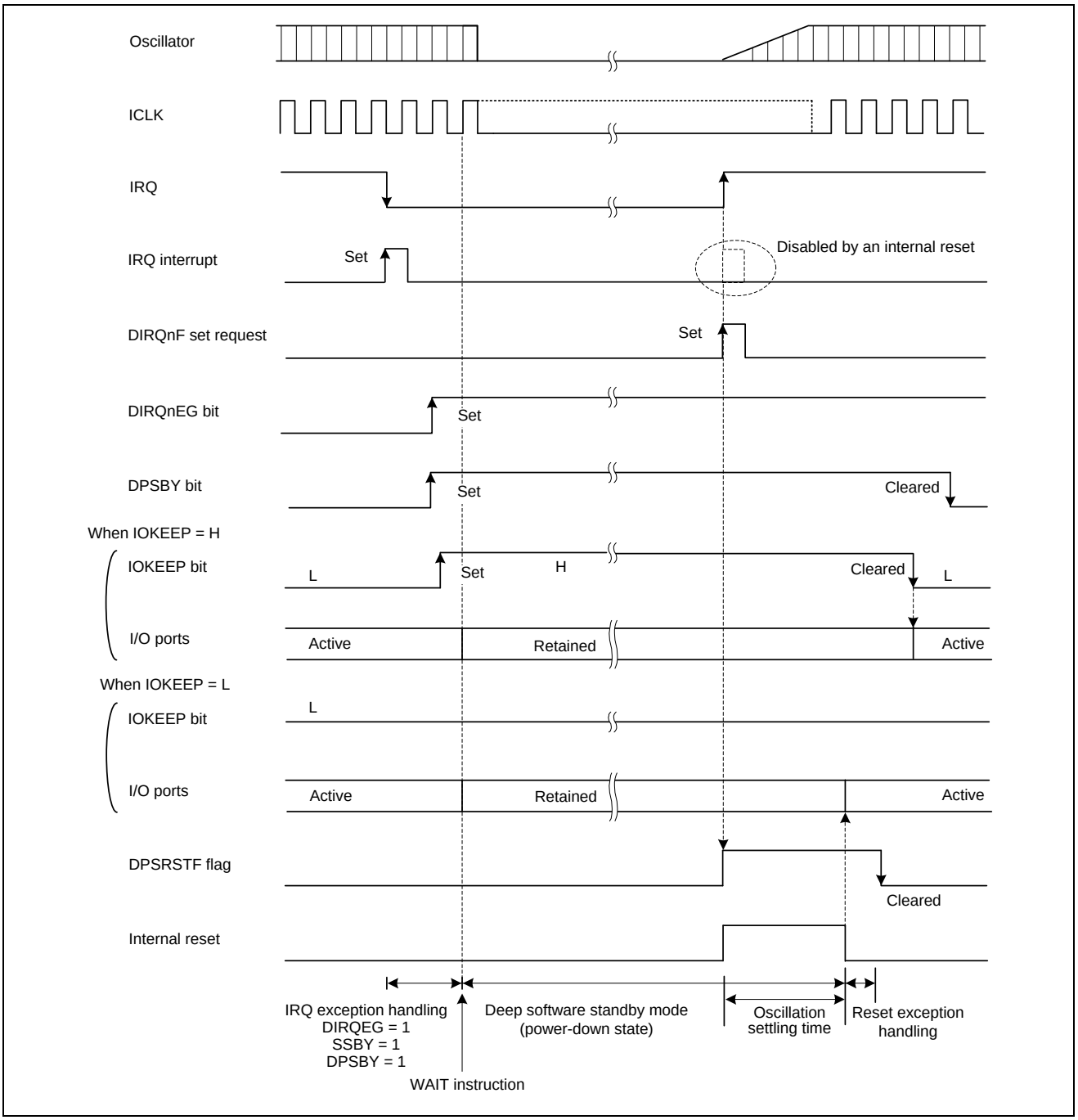



**Figure 8.3 Example of Deep Software Standby Mode Application**



Downloaded from Arrow.com.

### 8.5.4.6 Flowchart to Use Deep Software Standby Mode

Figure 8.4 shows an example of flowchart to use deep software standby mode.

In this example, the DPSRSTF flag in RSTSR of the reset function is read after the reset exception handling to determine whether a reset was generated by the RES# pin or by the cancellation of deep software standby mode.

In the case of a reset by the RES# pin, a transition to deep software standby mode is made after required register settings.

In the case of a reset by the cancellation of deep software standby mode, the IOKEEP bit in DPSBYCR is cleared to 0 after the I/O port settings.

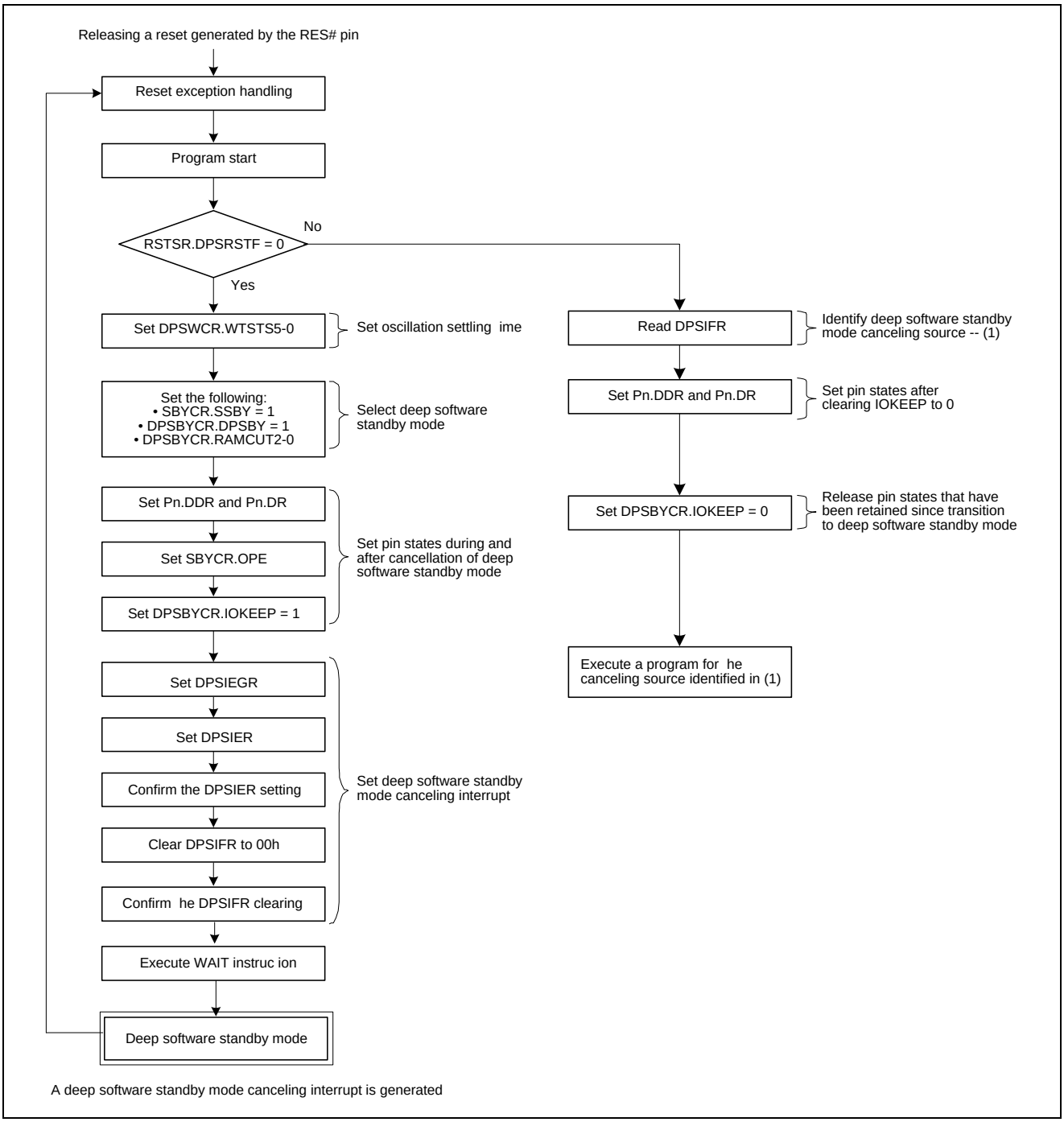



**Figure 8.4 Example of Flowchart to Use Deep Software Standby Mode**



Downloaded from Arrow.com.

## 8.6 BCLK Output Control

The BCLK output can be controlled with the PSTOP1 bit in SCKCR and the B3 bit in P5.DDR of corresponding P53.

When the PSTOP1 bit is cleared to 0, P53 functions as the BCLK output. When the PSTOP1 bit is set to 1, the BCLK output stops at the end of the bus cycle and goes high. When the B3 bit in P5.DDR of P53 is cleared to 0, the BCLK output is disabled and the pin functions as an input port.

Table 8.6 shows the BCLK pin state in each operating mode.

**Table 8.6 BCLK Pin (P53) State in Each Operating Mode**

| Register Settings | | Normal Operating Mode | Sleep Mode | All-Module Clock Stop Mode | Software Standby Mode | | Deep Software Standby Mode | |
|---|---|---|---|---|---|---|---|---|
| DDR | PSTOP1 | | | | OPE = 0 | OPE = 1 | IOKEE P= 0 | IOKEEP = 1 |
| 0 | × | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| 1 | 0 | BCLK output | BCLK output | BCLK output | High | High | High | High |
| 1 | 1 | High | High | High | High | High | High | High |



Downloaded from Arrow.com.

## 8.7 Usage Notes

### 8.7.1 I/O Port States

I/O port states are retained in software standby mode and deep software standby mode. Therefore, supply current is not reduced while output signals are held high.

### 8.7.2 Module Stop State of the DMAC and DTC

Before setting the MSTPA28 or MSTPA27 bits in MSTPCRA to 1, clear the DMST bit in DMSCNT of the DMAC and the DTCST bit in DTCST of the DTC to 0 so that initiating transfer by the DTC or DMAC is not possible. For details, see section 12, DMA Controller (DMAC) and section 13, Data Transfer Controller (DTC).

### 8.7.3 On-Chip Peripheral Module Interrupts

Operation of relevant interrupt is disabled in the module stop state. Therefore, if the module stop state is made with an interrupt request pending, a CPU interrupt source or a DMAC (or DTC) startup source cannot be cleared. For this reason, disable interrupts before entering the module stop state.

### 8.7.4 Write-Access to MSTPCRA, MSTPCRB, and MSTPCRC

Write-accesses to MSTPCRA, MSTPCRB, and MSTPCRC should be made only by the CPU.

### 8.7.5 Input Buffer Control by DIRQnE Bit (n = 3 to 0)

The input buffers for the P30/IRQ0-A to P33/IRQ3-A pins are enabled by setting the DIRQnE bit (n = 3 to 0) in DPSIER to 1. Therefore, note that inputs to these pins are reflected in the DIRQnF flag (n = 3 to 0) in DPSIFR, but are not transferred to the interrupt control unit, peripheral modules, and I/O ports. Inputs to the interrupt control unit, peripheral modules, and I/O ports should be controlled by each Pn.ICR.

### 8.7.6 Conflict between Transition to Deep Software Standby Mode and Interrupt

If a conflict occurs between a transition to deep software standby mode and generation of a software standby mode canceling source, the transition to deep software standby mode is not realized but the software standby mode canceling sequence is started. After the oscillation settling time specified by the STS[4:0] bits in SBYCR for software standby mode has passed, the interrupt exception handling is started.

Note that if a conflict occurs between a transition to deep software standby mode and generation of an NMI interrupt, the NMI interrupt exception handling routine is required.

If a conflict occurs between a transition to deep software standby mode and generation of an interrupt of IRQ0 to IRQ15, a transition to deep software standby mode can be made by clearing the SSIj bit (j = 15 to 0) in SSIER of the ICU to 0 beforehand without starting the interrupt exception handling.

### 8.7.7 Timing of Wait Instructions

The WAIT instruction is executed before completion of the preceding register write; it may be executed before the register modification is reflected, causing unintended operation. To avoid this, execute the WAIT instruction after confirming that the last write to the register has been completed.



Downloaded from Arrow.com.

# 9. Exceptions

## 9.1 Types of Exceptions

During the execution of a program by the CPU, the occurrence of certain events may necessitate suspending execution of the main flow of the program and starting the execution of another flow. Such events are called exceptions.

The RX CPU supports the seven types of exceptions listed in figure 9.1.

The occurrence of an exception causes the processor mode to switch to supervisor mode.

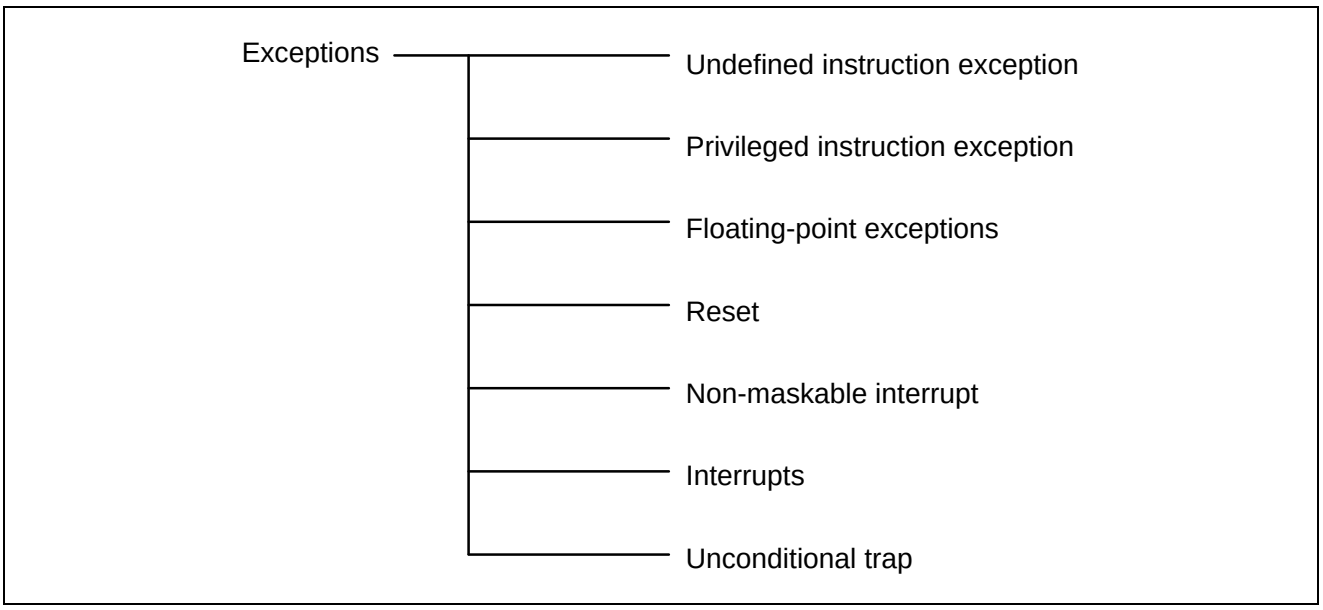



**Figure 9.1 Types of Exception**



Downloaded from Arrow.com.

### 9.1.1 Undefined Instruction Exception

An undefined instruction exception occurs when execution of an undefined instruction (an instruction not implemented) is detected.

### 9.1.2 Privileged Instruction Exception

A privileged instruction exception occurs when execution of a privileged instruction is detected while operation is in user mode. Privileged instructions can only be executed in supervisor mode.

### 9.1.3 Floating-Point Exceptions

Floating-point exceptions include the five specified in the IEEE754 standard, namely overflow, underflow, inexact, division-by-zero, and invalid operation, and a further floating-point exception that is generated on the detection of unimplemented processing. The exception processing of floating-point exceptions is masked when the EX, EU, EZ, EO, or EV bit in FPSW is 0.

### 9.1.4 Reset

A reset through input of the reset signal to the CPU causes the exception request. This has the highest priority of any exception and is always accepted.

### 9.1.5 Non-Maskable Interrupt

The non-maskable interrupt is generated by input of the non-maskable interrupt signal to the CPU and is only used when a fatal fault is considered to have occurred in the system. Never end the exception handling routine for the non-maskable interrupt with an attempt to return to the program that was being executed at the time of interrupt generation.

### 9.1.6 Interrupts

Interrupts are generated by the input of interrupt signals to the CPU. The interrupt with the highest priority can be selected for handling as a fast interrupt. The exception processing of interrupts is masked when the I bit in the PSW is 0.

### 9.1.7 Unconditional Trap

An unconditional trap is generated when the INT or BRK instruction is executed.



Downloaded from Arrow.com.

## 9.2 Exception Handling Procedure

For exception handling, part of the processing is handled automatically by hardware and part is handled by a program (the exception handling routine) that has been written by the user. Figure 9.2 shows the handling procedure when an exception other than a reset is accepted.

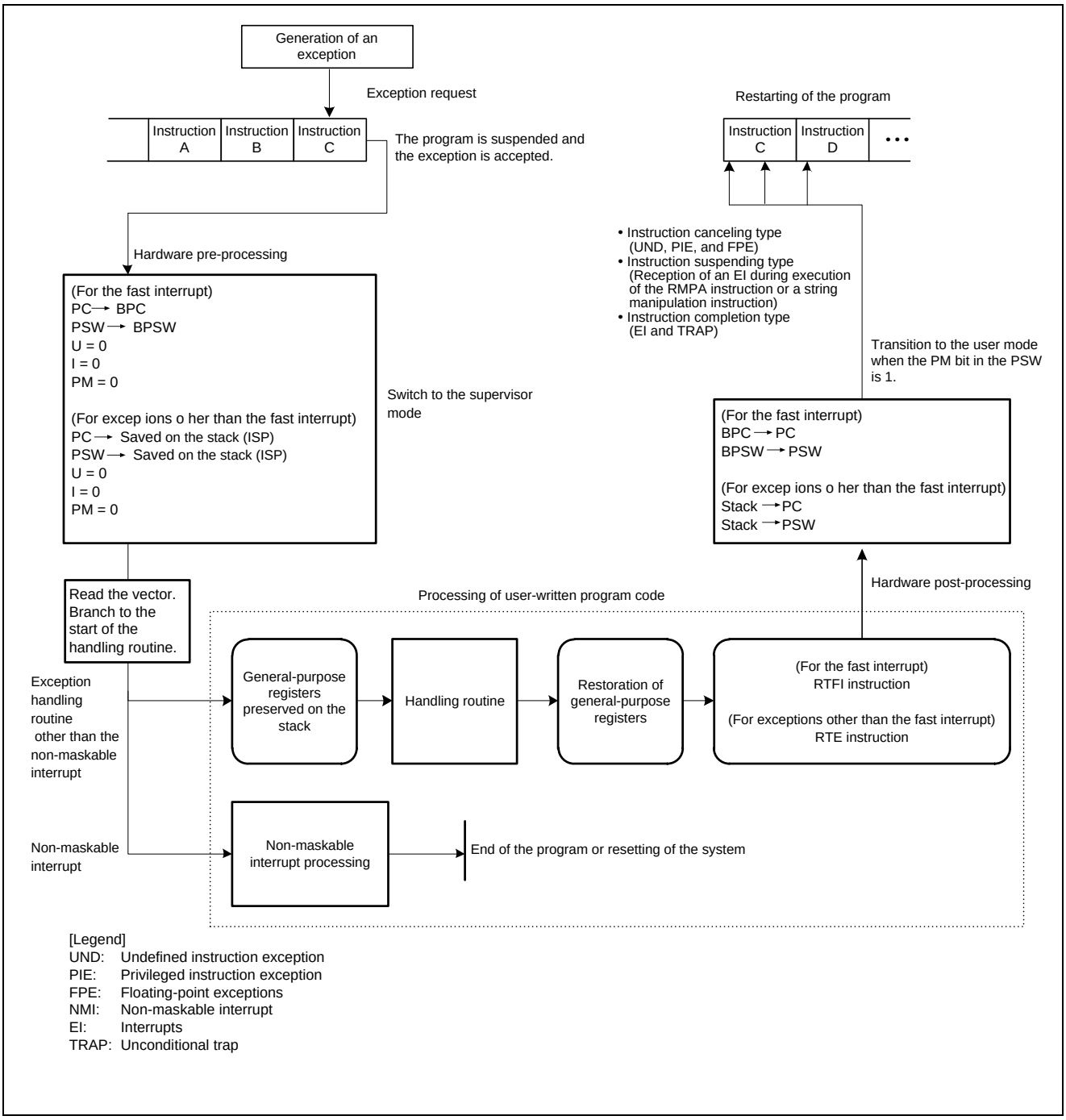



**Figure 9.2 Outline of the Exception Handling Procedure**



Downloaded from Arrow.com.

When an exception is accepted, hardware processing by the RX CPU is followed by vector access to acquire the address of the branch destination. A vector address is allocated to each exception. The branch destination address of the handler for the given exception is written to each vector address. The combination is referred to as a vector.

Hardware pre-processing by the RX CPU handles saving of the contents of the program counter (PC) and processor status word (PSW). In the case of the fast interrupt, the contents are saved in the backup PC (BPC) and the backup PSW (BPSW), respectively. In the case of other exceptions, the contents are preserved in the stack area.

General purpose registers and control registers other than the PC and PSW that are to be used within the exception handling routine must be preserved on the stack by user program code at the start of the exception handling routine.

On completion of processing by most exception processing handlers, registers preserved under program control are restored and the RTE instruction is executed to restore execution from the exception handling routine to the original program. For return from the fast interrupt, the RTFI instruction is used instead. In the case of the non-maskable interrupt, however, end the program or reset the system without returning to the original program.

Hardware post-processing by the RX CPU handles restoration of the pre-exception contents of the PC and PSW. In the case of the fast interrupt, the contents of the BPC and BPSW are restored to the PC and PSW, respectively. In the case of other exceptions, the contents are restored from the stack area to the PC and PSW.



Downloaded from Arrow.com.

## 9.3 Acceptance of Exceptions

When an exception occurs, the CPU suspends the execution of the program and processing branches to the start of the exception handling routine.

### 9.3.1 Timing of Acceptance and Saved PC Values

Table 9.1 lists the timing of acceptance and program counter (PC) value to be saved for each type of exception event.

**Table 9.1 Timing of Acceptance and Saved PC Value**

| Exception | | Type of Handling | Timing of Acceptance | Value Saved in the BPC/ on the Stack |
|---|---|---|---|---|
| Undefined instruction exception | | Instruction canceling type | During instruction execution | PC value of the instruction that generated the exception |
| Privileged instruction exception | | Instruction canceling type | During instruction execution | PC value of the instruction that generated the exception |
| Floating-point exceptions | | Instruction canceling type | During instruction execution | PC value of the instruction that generated the exception |
| Reset | | Program abandonment type | Any machine cycle | None |
| Non-maskable interrupt | During execution of the RMPA, SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, and SWHILE instructions | Instruction suspending type | During instruction execution | PC value of the instruction being executed |
| | Other than the above | Instruction completion type | At the next break between instructions | PC value of the next instruction |
| Interrupts | During execution of the RMPA, SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, and SWHILE instructions | Instruction suspending type | During instruction execution | PC value of the instruction being executed |
| | Other than the above | Instruction completion type | At the next break between instructions | PC value of the next instruction |
| Unconditional trap | | Instruction completion type | At the next break between instructions | PC value of the next instruction |



Downloaded from Arrow.com.

### 9.3.2 Vector and Site for Saving the Values in the PC and PSW

The vector for each type of exception and the site for saving the values of the program counter (PC) and processor status word (PSW) are listed in table 9.2.

**Table 9.2 Vector and Site for Saving the Values in the PC and PSW**

| Exception | | Vector | Site for Saving the Values in the PC and PSW |
|---|---|---|---|
| Undefined instruction exception | | Fixed vector table | Stack |
| Privileged instruction exception | | Fixed vector table | Stack |
| Floating-point exceptions | | Fixed vector table | Stack |
| Reset | | Fixed vector table | Nowhere |
| Non-maskable interrupt | | Fixed vector table | Stack |
| Interrupts | Fast interrupt | FINTV | BPC and BPSW |
| | Other than the above | Relocatable vector table (INTB) | Stack |
| Unconditional trap | | Relocatable vector table (INTB) | Stack |



Downloaded from Arrow.com.

## 9.4 Hardware Processing for Accepting and Returning from Exceptions

This section describes the hardware processing for accepting and returning from exceptions other than a reset.

### (1) Hardware pre-processing for accepting an exception

#### (a) Saving the values in the PSW

- For the fast interrupt
  PSW → BPSW
- For other exceptions
  PSW → Stack area

Note: The values in the FPSW are not saved by hardware pre-processing. Therefore, if floating-point instructions are to be used within an exception handling routine, the user must ensure that these values are saved on the stack.

#### (b) Updating of the PM, U, and I bits in the PSW

I: Cleared
I: Cleared
PM: Cleared

#### (c) Saving the values in the PC

- For the fast interrupt
  PC → BPC
- For other exceptions
  PC → Stack area

#### (d) Set the branch-destination address of the exception handling routine in the PC

Processing is shifted to the exception handling routine by acquiring the vector corresponding to the exception and branching accordingly.

### (2) Hardware post-processing for execution of RTE and RTFI instructions

#### (a) Restoring the values in the PSW

- For the fast interrupt
  BPSW → PSW
- For other exceptions
  Stack area→ PSW

#### (b) Restoring the values in the PC

- For the fast interrupt
  BPC → PC
- For other exceptions
  Stack area→ PC



Downloaded from Arrow.com.

## 9.5 Hardware Pre-Processing

The sequences of hardware pre-processing from reception of each exception request to execution of the associated exception handling routine are explained below.

### 9.5.1 Undefined Instruction Exception

1. The value of the processor status word (PSW) is saved on the stack (ISP).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. The value of the program counter (PC) is saved on the stack (ISP).
4. The address of the processing routine is fetched from the vector address, FFFFFFDCh.
5. The PC is set to the fetched address and processing branches to the start of the exception handling routine.

### 9.5.2 Privileged Instruction Exception

1. The value in the processor status word (PSW) is saved on the stack (ISP).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. The value of the program counter (PC) is saved on the stack (ISP).
4. The address of the processing routine is fetched from the vector address, FFFFFFD0h.
5. The PC is set to the fetched address and processing branches to the start of the exception handling routine.

### 9.5.3 Floating-Point Exceptions

1. The value in the processor status word (PSW) is saved on the stack (ISP).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. The value of the program counter (PC) is saved on the stack (ISP).
4. The address of the processing routine is fetched from the vector address, FFFFFFE4h.
5. The PC is set to the fetched address and processing branches to the start of the exception handling routine.

### 9.5.4 Reset

1. The control registers are initialized.
2. The address of the processing routine is fetched from the vector address, FFFFFFFCh.
3. The PC is set to the fetched address.



Downloaded from Arrow.com.

### 9.5.5 Non-Maskable Interrupt

1. The value of the processor status word (PSW) is saved on the stack (ISP).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. If the interrupt was generated during the execution of an RMPA, SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, or SWHILE instruction, the value of the program counter (PC) for that instruction is saved on the stack (ISP). For other instructions, the PC value of the next instruction is saved.
4. The processor interrupt priority level bits (IPL[2:0]) in the PSW are set to 111b.
5. The address of the processing routine is fetched from the vector address, FFFFFFF8h.
6. The PC is set to the fetched address and the processing branches to the start of the exception handling routine.

### 9.5.6 Interrupts

1. The value of the processor status word (PSW) is saved on the stack (ISP) or, for the fast interrupt, in the backup PSW (BPSW).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. If the interrupt was generated during the execution of an RMPA, SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, or SWHILE instruction, the value of the program counter (PC) for that instruction is saved. For other instructions, the PC value of the next instruction is saved. Saving of the PC is in the backup PC (BPC) for fast interrupts.
4. The processor interrupt priority level bits (IPL[2:0]) in the PSW indicate the interrupt priority level of the interrupt.
5. The address of the processing routine for an interrupt source other than the fast interrupt is fetched from the relocatable vector table. For the fast interrupt, the address is fetched from the fast interrupt vector register (FINTV).
6. The PC is set to the fetched address and processing branches to the start of the exception handling routine.

### 9.5.7 Unconditional Trap

1. The value in the processor status word (PSW) is saved on the stack (ISP).
2. The processor mode select bit (PM), the stack pointer select bit (U), and the interrupt enable bit (I) in the PSW are cleared to 0.
3. The value of the program counter (PC) for the next instruction is saved on the stack (ISP).
4. For the INT instruction, the value at the vector corresponding to the INT instruction number is fetched from the relocatable vector table.
   For the BRK instruction, the value at the vector from the start address is fetched from the relocatable vector table.
5. The PC is set to the fetched address and the processing branches to the start of the exception handling routine.



Downloaded from Arrow.com.

## 9.6 Return from Exception Handling Routines

Executing the instructions listed in table 9.3 at the end of the corresponding exception handling routines restores the values of the program counter (PC) and processor status word (PSW) that were saved on the stack or in the control registers (BPC and BPSW) immediately before the exception handling sequence.

**Table 9.3 Return from Exception Handling Routines**

| Exception | | Instruction for Return |
|---|---|---|
| Undefined instruction exception | | RTE |
| Privileged instruction exception | | RTE |
| Floating-point exceptions | | RTE |
| Reset | | Return is impossible |
| Non-maskable interrupt | | Return is impossible |
| Interrupts | Fast interrupt | RTFI |
| | Other than the above | RTE |
| Unconditional trap | | RTE |

## 9.7 Order of Priority for Exceptions

The order of priority for exceptions is given in table 9.4. When multiple exceptions are generated at the same time, the exception with the highest priority is accepted first.

**Table 9.4 Order of Priority for Exceptions**

| Order of Priority | | Exception |
|---|---|---|
| High | 1 | Reset |
| ↑ | 2 | Non-maskable interrupt |
| | 3 | Interrupts |
| | 4 | Undefined instruction exception<br>Privileged instruction exception |
| | 5 | Unconditional trap |
| Low | 6 | Floating-point exceptions |



Downloaded from Arrow.com.

# 10. Interrupt Control Unit (ICU)

## 10.1 Overview

The interrupt control unit (ICU) responds to interrupt signals from peripheral modules and external pins to convey interrupt requests to the CPU and activate the DTC and DMAC.

Table 10.1 lists the specifications of the interrupt control unit, and figure 10.1 shows a block diagram of the interrupt control unit.

**Table 10.1 Specifications of the Interrupt Control Unit**

| Item | | Description |
|---|---|---|
| Interrupts | Peripheral function interrupts | Interrupts from peripheral modules<br>• Number of sources: 116<br>• Interrupt detection: Edge detection/level detection<br>Edge detection or level detection is determined for each source of connected peripheral modules. |
| | External interrupts | Interrupts from pins IRQ15 to IRQ0<br>• Number of sources: 16<br>• Interrupt detection: Low level/falling edge/rising edge/rising and falling edges<br>One of these detection methods can be set for each source. |
| Non-maskable interrupt | NMI pin interrupts | Interrupts from the NMI pin<br>• Number of sources: 1<br>• Interrupt detection: Falling edge/rising edge |
| Interrupt priority | | Specified by registers. |
| Vector address | | Since a vector address is allocated to each interrupt source, interrupt sources need not be identified by the software program. |
| Fast interrupt function | | Faster interrupt processing of the CPU can be set only for a single interrupt source. |
| DTC/DMAC control | | The DTC and DMAC can be activated by interrupt sources.<br>• Number of DTC activating sources: 86 (70 peripheral function interrupts + 16 external interrupts)<br>• Number of DMAC activating sources: 42 (38 peripheral function interrupts + 4 external interrupts) |
| Return from low power consumption modes | | • Sleep mode: Return is initiated by NMI pin interrupts or any other interrupts.<br>• All-module clock stop mode: Return is initiated by NMI pin interrupts, external interrupts, or peripheral module interrupts (WDT and TMR)<br>• Software standby mode: Return is initiated by NMI pin interrupts or external interrupts. |



Downloaded from Arrow.com.

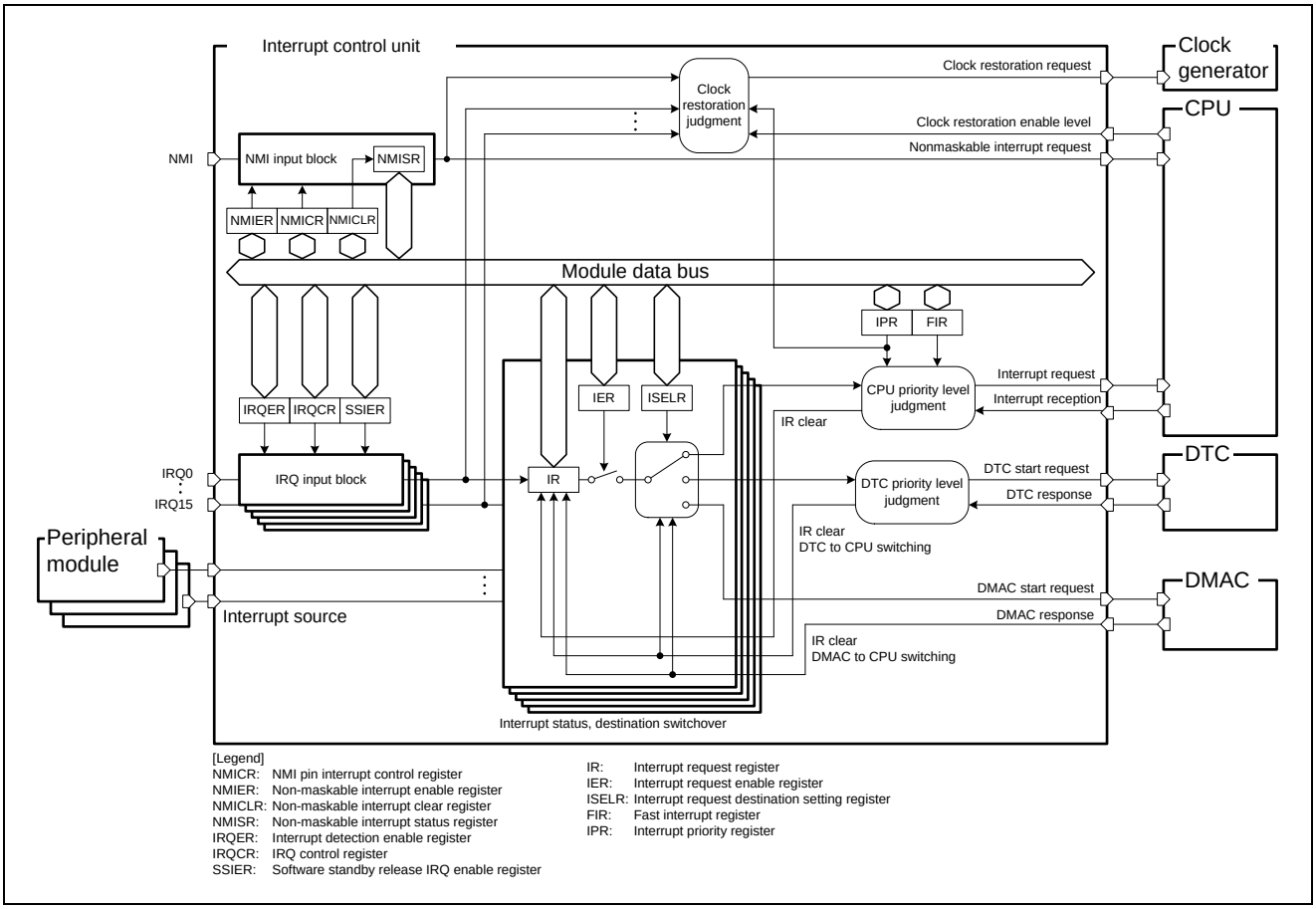



**Figure 10.1 Block Diagram of Interrupt Control Unit**

Table 10.2 shows the input/output pins of the interrupt control unit.

**Table 10.2 Pin Configuration of ICU**

| Pin Name | I/O | Description |
|---|---|---|
| NMI | Input | Non-maskable interrupt request pin |
| IRQ15 to IRQ0 | Input | External interrupt request pins |



Downloaded from Arrow.com.

## 10.2 Register Descriptions

Table 10.3 lists the registers of the interrupt control unit.

**Table 10.3 Registers of the Interrupt Control Unit**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request register 016 | IR016 | 00h | 0008 7010h | 8 |
| Interrupt request register 021 | IR021 | 00h | 0008 7015h | 8 |
| Interrupt request register 023 | IR023 | 00h | 0008 7017h | 8 |
| Interrupt request register 028 | IR028 | 00h | 0008 701Ch | 8 |
| Interrupt request register 029 | IR029 | 00h | 0008 701Dh | 8 |
| Interrupt request register 030 | IR030 | 00h | 0008 701Eh | 8 |
| Interrupt request register 031 | IR031 | 00h | 0008 701Fh | 8 |
| Interrupt request register 064 | IR064 | 00h | 0008 7040h | 8 |
| Interrupt request register 065 | IR065 | 00h | 0008 7041h | 8 |
| Interrupt request register 066 | IR066 | 00h | 0008 7042h | 8 |
| Interrupt request register 067 | IR067 | 00h | 0008 7043h | 8 |
| Interrupt request register 068 | IR068 | 00h | 0008 7044h | 8 |
| Interrupt request register 069 | IR069 | 00h | 0008 7045h | 8 |
| Interrupt request register 070 | IR070 | 00h | 0008 7046h | 8 |
| Interrupt request register 071 | IR071 | 00h | 0008 7047h | 8 |
| Interrupt request register 072 | IR072 | 00h | 0008 7048h | 8 |
| Interrupt request register 073 | IR073 | 00h | 0008 7049h | 8 |
| Interrupt request register 074 | IR074 | 00h | 0008 704Ah | 8 |
| Interrupt request register 075 | IR075 | 00h | 0008 704Bh | 8 |
| Interrupt request register 076 | IR076 | 00h | 0008 704Ch | 8 |
| Interrupt request register 077 | IR077 | 00h | 0008 704Dh | 8 |
| Interrupt request register 078 | IR078 | 00h | 0008 704Eh | 8 |
| Interrupt request register 079 | IR079 | 00h | 0008 704Fh | 8 |
| Interrupt request register 096 | IR096 | 00h | 0008 7060h | 8 |
| Interrupt request register 098 | IR098 | 00h | 0008 7062h | 8 |
| Interrupt request register 099 | IR099 | 00h | 0008 7063h | 8 |
| Interrupt request register 100 | IR100 | 00h | 0008 7064h | 8 |
| Interrupt request register 101 | IR101 | 00h | 0008 7065h | 8 |
| Interrupt request register 104 | IR104 | 00h | 0008 7068h | 8 |
| Interrupt request register 105 | IR105 | 00h | 0008 7069h | 8 |
| Interrupt request register 106 | IR106 | 00h | 0008 706Ah | 8 |
| Interrupt request register 107 | IR107 | 00h | 0008 706Bh | 8 |
| Interrupt request register 108 | IR108 | 00h | 0008 706Ch | 8 |
| Interrupt request register 111 | IR111 | 00h | 0008 706Fh | 8 |
| Interrupt request register 112 | IR112 | 00h | 0008 7070h | 8 |
| Interrupt request register 115 | IR115 | 00h | 0008 7073h | 8 |
| Interrupt request register 116 | IR116 | 00h | 0008 7074h | 8 |
| Interrupt request register 117 | IR117 | 00h | 0008 7075h | 8 |
| Interrupt request register 118 | IR118 | 00h | 0008 7076h | 8 |
| Interrupt request register 120 | IR120 | 00h | 0008 7078h | 8 |
| Interrupt request register 121 | IR121 | 00h | 0008 7079h | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request register 122 | IR122 | 00h | 0008 707Ah | 8 |
| Interrupt request register 123 | IR123 | 00h | 0008 707Bh | 8 |
| Interrupt request register 124 | IR124 | 00h | 0008 707Ch | 8 |
| Interrupt request register 125 | IR125 | 00h | 0008 707Dh | 8 |
| Interrupt request register 126 | IR126 | 00h | 0008 707Eh | 8 |
| Interrupt request register 127 | IR127 | 00h | 0008 707Fh | 8 |
| Interrupt request register 128 | IR128 | 00h | 0008 7080h | 8 |
| Interrupt request register 131 | IR131 | 00h | 0008 7083h | 8 |
| Interrupt request register 132 | IR132 | 00h | 0008 7084h | 8 |
| Interrupt request register 133 | IR133 | 00h | 0008 7085h | 8 |
| Interrupt request register 134 | IR134 | 00h | 0008 7086h | 8 |
| Interrupt request register 136 | IR136 | 00h | 0008 7088h | 8 |
| Interrupt request register 137 | IR137 | 00h | 0008 7089h | 8 |
| Interrupt request register 138 | IR138 | 00h | 0008 708Ah | 8 |
| Interrupt request register 139 | IR139 | 00h | 0008 708Bh | 8 |
| Interrupt request register 140 | IR140 | 00h | 0008 708Ch | 8 |
| Interrupt request register 141 | IR141 | 00h | 0008 708Dh | 8 |
| Interrupt request register 142 | IR142 | 00h | 0008 708Eh | 8 |
| Interrupt request register 145 | IR145 | 00h | 0008 7091h | 8 |
| Interrupt request register 146 | IR146 | 00h | 0008 7092h | 8 |
| Interrupt request register 149 | IR149 | 00h | 0008 7095h | 8 |
| Interrupt request register 150 | IR150 | 00h | 0008 7096h | 8 |
| Interrupt request register 151 | IR151 | 00h | 0008 7097h | 8 |
| Interrupt request register 152 | IR152 | 00h | 0008 7098h | 8 |
| Interrupt request register 154 | IR154 | 00h | 0008 709Ah | 8 |
| Interrupt request register 155 | IR155 | 00h | 0008 709Bh | 8 |
| Interrupt request register 156 | IR156 | 00h | 0008 709Ch | 8 |
| Interrupt request register 157 | IR157 | 00h | 0008 709Dh | 8 |
| Interrupt request register 158 | IR158 | 00h | 0008 709Eh | 8 |
| Interrupt request register 159 | IR159 | 00h | 0008 709Fh | 8 |
| Interrupt request register 160 | IR160 | 00h | 0008 70A0h | 8 |
| Interrupt request register 161 | IR161 | 00h | 0008 70A1h | 8 |
| Interrupt request register 162 | IR162 | 00h | 0008 70A2h | 8 |
| Interrupt request register 165 | IR165 | 00h | 0008 70A5h | 8 |
| Interrupt request register 166 | IR166 | 00h | 0008 70A6h | 8 |
| Interrupt request register 167 | IR167 | 00h | 0008 70A7h | 8 |
| Interrupt request register 168 | IR168 | 00h | 0008 70A8h | 8 |
| Interrupt request register 170 | IR170 | 00h | 0008 70AAh | 8 |
| Interrupt request register 171 | IR171 | 00h | 0008 70ABh | 8 |
| Interrupt request register 174 | IR174 | 00h | 0008 70AEh | 8 |
| Interrupt request register 175 | IR175 | 00h | 0008 70AFh | 8 |
| Interrupt request register 176 | IR176 | 00h | 0008 70B0h | 8 |
| Interrupt request register 177 | IR177 | 00h | 0008 70B1h | 8 |
| Interrupt request register 178 | IR178 | 00h | 0008 70B2h | 8 |
| Interrupt request register 179 | IR179 | 00h | 0008 70B3h | 8 |
| Interrupt request register 180 | IR180 | 00h | 0008 70B4h | 8 |
| Interrupt request register 181 | IR181 | 00h | 0008 70B5h | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request register 182 | IR182 | 00h | 0008 70B6h | 8 |
| Interrupt request register 183 | IR183 | 00h | 0008 70B7h | 8 |
| Interrupt request register 184 | IR184 | 00h | 0008 70B8h | 8 |
| Interrupt request register 185 | IR185 | 00h | 0008 70B9h | 8 |
| Interrupt request register 198 | IR198 | 00h | 0008 70C6h | 8 |
| Interrupt request register 199 | IR199 | 00h | 0008 70C7h | 8 |
| Interrupt request register 200 | IR200 | 00h | 0008 70C8h | 8 |
| Interrupt request register 201 | IR201 | 00h | 0008 70C9h | 8 |
| Interrupt request register 214 | IR214 | 00h | 0008 70D6h | 8 |
| Interrupt request register 215 | IR215 | 00h | 0008 70D7h | 8 |
| Interrupt request register 216 | IR216 | 00h | 0008 70D8h | 8 |
| Interrupt request register 217 | IR217 | 00h | 0008 70D9h | 8 |
| Interrupt request register 218 | IR218 | 00h | 0008 70DAh | 8 |
| Interrupt request register 219 | IR219 | 00h | 0008 70DBh | 8 |
| Interrupt request register 220 | IR220 | 00h | 0008 70DCh | 8 |
| Interrupt request register 221 | IR221 | 00h | 0008 70DDh | 8 |
| Interrupt request register 222 | IR222 | 00h | 0008 70DEh | 8 |
| Interrupt request register 223 | IR223 | 00h | 0008 70DFh | 8 |
| Interrupt request register 224 | IR224 | 00h | 0008 70E0h | 8 |
| Interrupt request register 225 | IR225 | 00h | 0008 70E1h | 8 |
| Interrupt request register 226 | IR226 | 00h | 0008 70E2h | 8 |
| Interrupt request register 227 | IR227 | 00h | 0008 70E3h | 8 |
| Interrupt request register 228 | IR228 | 00h | 0008 70E4h | 8 |
| Interrupt request register 229 | IR229 | 00h | 0008 70E5h | 8 |
| Interrupt request register 230 | IR230 | 00h | 0008 70E6h | 8 |
| Interrupt request register 231 | IR231 | 00h | 0008 70E7h | 8 |
| Interrupt request register 232 | IR232 | 00h | 0008 70E8h | 8 |
| Interrupt request register 233 | IR233 | 00h | 0008 70E9h | 8 |
| Interrupt request register 234 | IR234 | 00h | 0008 70EAh | 8 |
| Interrupt request register 235 | IR235 | 00h | 0008 70EBh | 8 |
| Interrupt request register 236 | IR236 | 00h | 0008 70ECh | 8 |
| Interrupt request register 237 | IR237 | 00h | 0008 70EDh | 8 |
| Interrupt request register 238 | IR238 | 00h | 0008 70EEh | 8 |
| Interrupt request register 239 | IR239 | 00h | 0008 70EFh | 8 |
| Interrupt request register 240 | IR240 | 00h | 0008 70F0h | 8 |
| Interrupt request register 241 | IR241 | 00h | 0008 70F1h | 8 |
| Interrupt request register 246 | IR246 | 00h | 0008 70F6h | 8 |
| Interrupt request register 247 | IR247 | 00h | 0008 70F7h | 8 |
| Interrupt request register 248 | IR248 | 00h | 0008 70F8h | 8 |
| Interrupt request register 249 | IR249 | 00h | 0008 70F9h | 8 |
| Interrupt request register 250 | IR250 | 00h | 0008 70FAh | 8 |
| Interrupt request register 251 | IR251 | 00h | 0008 70FBh | 8 |
| Interrupt request register 252 | IR252 | 00h | 0008 70FCh | 8 |
| Interrupt request register 253 | IR253 | 00h | 0008 70FDh | 8 |
| Interrupt request destination setting register 028 | ISELR028 | 00h | 0008 711Ch | 8 |
| Interrupt request destination setting register 029 | ISELR029 | 00h | 0008 711Dh | 8 |
| Interrupt request destination setting register 030 | ISELR030 | 00h | 0008 711Eh | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request destination setting register 031 | ISELR031 | 00h | 0008 711Fh | 8 |
| Interrupt request destination setting register 064 | ISELR064 | 00h | 0008 7140h | 8 |
| Interrupt request destination setting register 065 | ISELR065 | 00h | 0008 7141h | 8 |
| Interrupt request destination setting register 066 | ISELR066 | 00h | 0008 7142h | 8 |
| Interrupt request destination setting register 067 | ISELR067 | 00h | 0008 7143h | 8 |
| Interrupt request destination setting register 068 | ISELR068 | 00h | 0008 7144h | 8 |
| Interrupt request destination setting register 069 | ISELR069 | 00h | 0008 7145h | 8 |
| Interrupt request destination setting register 070 | ISELR070 | 00h | 0008 7146h | 8 |
| Interrupt request destination setting register 071 | ISELR071 | 00h | 0008 7147h | 8 |
| Interrupt request destination setting register 072 | ISELR072 | 00h | 0008 7148h | 8 |
| Interrupt request destination setting register 073 | ISELR073 | 00h | 0008 7149h | 8 |
| Interrupt request destination setting register 074 | ISELR074 | 00h | 0008 714Ah | 8 |
| Interrupt request destination setting register 075 | ISELR075 | 00h | 0008 714Bh | 8 |
| Interrupt request destination setting register 076 | ISELR076 | 00h | 0008 714Ch | 8 |
| Interrupt request destination setting register 077 | ISELR077 | 00h | 0008 714Dh | 8 |
| Interrupt request destination setting register 078 | ISELR078 | 00h | 0008 714Eh | 8 |
| Interrupt request destination setting register 079 | ISELR079 | 00h | 0008 714Fh | 8 |
| Interrupt request destination setting register 098 | ISELR098 | 00h | 0008 7162h | 8 |
| Interrupt request destination setting register 099 | ISELR099 | 00h | 0008 7163h | 8 |
| Interrupt request destination setting register 100 | ISELR100 | 00h | 0008 7164h | 8 |
| Interrupt request destination setting register 101 | ISELR101 | 00h | 0008 7165h | 8 |
| Interrupt request destination setting register 104 | ISELR104 | 00h | 0008 7168h | 8 |
| Interrupt request destination setting register 105 | ISELR105 | 00h | 0008 7169h | 8 |
| Interrupt request destination setting register 106 | ISELR106 | 00h | 0008 716Ah | 8 |
| Interrupt request destination setting register 107 | ISELR107 | 00h | 0008 716Bh | 8 |
| Interrupt request destination setting register 111 | ISELR111 | 00h | 0008 716Fh | 8 |
| Interrupt request destination setting register 112 | ISELR112 | 00h | 0008 7170h | 8 |
| Interrupt request destination setting register 117 | ISELR117 | 00h | 0008 7175h | 8 |
| Interrupt request destination setting register 118 | ISELR118 | 00h | 0008 7176h | 8 |
| Interrupt request destination setting register 122 | ISELR122 | 00h | 0008 717Ah | 8 |
| Interrupt request destination setting register 123 | ISELR123 | 00h | 0008 717Bh | 8 |
| Interrupt request destination setting register 124 | ISELR124 | 00h | 0008 717Ch | 8 |
| Interrupt request destination setting register 125 | ISELR125 | 00h | 0008 717Dh | 8 |
| Interrupt request destination setting register 127 | ISELR127 | 00h | 0008 717Fh | 8 |
| Interrupt request destination setting register 128 | ISELR128 | 00h | 0008 7180h | 8 |
| Interrupt request destination setting register 133 | ISELR133 | 00h | 0008 7185h | 8 |
| Interrupt request destination setting register 134 | ISELR134 | 00h | 0008 7186h | 8 |
| Interrupt request destination setting register 138 | ISELR138 | 00h | 0008 718Ah | 8 |
| Interrupt request destination setting register 139 | ISELR139 | 00h | 0008 718Bh | 8 |
| Interrupt request destination setting register 140 | ISELR140 | 00h | 0008 718Ch | 8 |
| Interrupt request destination setting register 141 | ISELR141 | 00h | 0008 718Dh | 8 |
| Interrupt request destination setting register 145 | ISELR145 | 00h | 0008 7191h | 8 |
| Interrupt request destination setting register 146 | ISELR146 | 00h | 0008 7192h | 8 |
| Interrupt request destination setting register 151 | ISELR151 | 00h | 0008 7197h | 8 |
| Interrupt request destination setting register 152 | ISELR152 | 00h | 0008 7198h | 8 |
| Interrupt request destination setting register 156 | ISELR156 | 00h | 0008 719Ch | 8 |
| Interrupt request destination setting register 157 | ISELR157 | 00h | 0008 719Dh | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request destination setting register 158 | ISELR158 | 00h | 0008 719Eh | 8 |
| Interrupt request destination setting register 159 | ISELR159 | 00h | 0008 719Fh | 8 |
| Interrupt request destination setting register 161 | ISELR161 | 00h | 0008 71A1h | 8 |
| Interrupt request destination setting register 162 | ISELR162 | 00h | 0008 71A2h | 8 |
| Interrupt request destination setting register 167 | ISELR167 | 00h | 0008 71A7h | 8 |
| Interrupt request destination setting register 168 | ISELR168 | 00h | 0008 71A8h | 8 |
| Interrupt request destination setting register 174 | ISELR174 | 00h | 0008 71AEh | 8 |
| Interrupt request destination setting register 175 | ISELR175 | 00h | 0008 71AFh | 8 |
| Interrupt request destination setting register 177 | ISELR177 | 00h | 0008 71B1h | 8 |
| Interrupt request destination setting register 178 | ISELR178 | 00h | 0008 71B2h | 8 |
| Interrupt request destination setting register 180 | ISELR180 | 00h | 0008 71B4h | 8 |
| Interrupt request destination setting register 181 | ISELR181 | 00h | 0008 71B5h | 8 |
| Interrupt request destination setting register 183 | ISELR183 | 00h | 0008 71B7h | 8 |
| Interrupt request destination setting register 184 | ISELR184 | 00h | 0008 71B8h | 8 |
| Interrupt request destination setting register 198 | ISELR198 | 00h | 0008 71C6h | 8 |
| Interrupt request destination setting register 199 | ISELR199 | 00h | 0008 71C7h | 8 |
| Interrupt request destination setting register 200 | ISELR200 | 00h | 0008 71C8h | 8 |
| Interrupt request destination setting register 201 | ISELR201 | 00h | 0008 71C9h | 8 |
| Interrupt request destination setting register 215 | ISELR215 | 00h | 0008 71D7h | 8 |
| Interrupt request destination setting register 216 | ISELR216 | 00h | 0008 71D8h | 8 |
| Interrupt request destination setting register 219 | ISELR219 | 00h | 0008 71DBh | 8 |
| Interrupt request destination setting register 220 | ISELR220 | 00h | 0008 71DCh | 8 |
| Interrupt request destination setting register 223 | ISELR223 | 00h | 0008 71DFh | 8 |
| Interrupt request destination setting register 224 | ISELR224 | 00h | 0008 71E0h | 8 |
| Interrupt request destination setting register 227 | ISELR227 | 00h | 0008 71E3h | 8 |
| Interrupt request destination setting register 228 | ISELR228 | 00h | 0008 71E4h | 8 |
| Interrupt request destination setting register 231 | ISELR231 | 00h | 0008 71E7h | 8 |
| Interrupt request destination setting register 232 | ISELR232 | 00h | 0008 71E8h | 8 |
| Interrupt request destination setting register 235 | ISELR235 | 00h | 0008 71EBh | 8 |
| Interrupt request destination setting register 236 | ISELR236 | 00h | 0008 71ECh | 8 |
| Interrupt request destination setting register 239 | ISELR239 | 00h | 0008 71EFh | 8 |
| Interrupt request destination setting register 240 | ISELR240 | 00h | 0008 71F0h | 8 |
| Interrupt request destination setting register 247 | ISELR247 | 00h | 0008 71F7h | 8 |
| Interrupt request destination setting register 248 | ISELR248 | 00h | 0008 71F8h | 8 |
| Interrupt request destination setting register 251 | ISELR251 | 00h | 0008 71FBh | 8 |
| Interrupt request destination setting register 252 | ISELR252 | 00h | 0008 71FCh | 8 |
| Interrupt request enable register 02 | IER02 | 00h | 0008 7202h | 8 |
| Interrupt request enable register 03 | IER03 | 00h | 0008 7203h | 8 |
| Interrupt request enable register 08 | IER08 | 00h | 0008 7208h | 8 |
| Interrupt request enable register 09 | IER09 | 00h | 0008 7209h | 8 |
| Interrupt request enable register 0C | IER0C | 00h | 0008 720Ch | 8 |
| Interrupt request enable register 0D | IER0D | 00h | 0008 720Dh | 8 |
| Interrupt request enable register 0E | IER0E | 00h | 0008 720Eh | 8 |
| Interrupt request enable register 0F | IER0F | 00h | 0008 720Fh | 8 |
| Interrupt request enable register 10 | IER10 | 00h | 0008 7210h | 8 |
| Interrupt request enable register 11 | IER11 | 00h | 0008 7211h | 8 |
| Interrupt request enable register 12 | IER12 | 00h | 0008 7212h | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt request enable register 13 | IER13 | 00h | 0008 7213h | 8 |
| Interrupt request enable register 14 | IER14 | 00h | 0008 7214h | 8 |
| Interrupt request enable register 15 | IER15 | 00h | 0008 7215h | 8 |
| Interrupt request enable register 16 | IER16 | 00h | 0008 7216h | 8 |
| Interrupt request enable register 17 | IER17 | 00h | 0008 7217h | 8 |
| Interrupt request enable register 18 | IER18 | 00h | 0008 7218h | 8 |
| Interrupt request enable register 19 | IER19 | 00h | 0008 7219h | 8 |
| Interrupt request enable register 1A | IER1A | 00h | 0008 721Ah | 8 |
| Interrupt request enable register 1B | IER1B | 00h | 0008 721Bh | 8 |
| Interrupt request enable register 1C | IER1C | 00h | 0008 721Ch | 8 |
| Interrupt request enable register 1D | IER1D | 00h | 0008 721Dh | 8 |
| Interrupt request enable register 1E | IER1E | 00h | 0008 721Eh | 8 |
| Interrupt request enable register 1F | IER1F | 00h | 0008 721Fh | 8 |
| Interrupt priority register 00 | IPR00 | 00h | 0008 7300h | 8 |
| Interrupt priority register 01 | IPR01 | 00h | 0008 7301h | 8 |
| Interrupt priority register 02 | IPR02 | 00h | 0008 7302h | 8 |
| Interrupt priority register 04 | IPR04 | 00h | 0008 7304h | 8 |
| Interrupt priority register 05 | IPR05 | 00h | 0008 7305h | 8 |
| Interrupt priority register 06 | IPR06 | 00h | 0008 7306h | 8 |
| Interrupt priority register 07 | IPR07 | 00h | 0008 7307h | 8 |
| Interrupt priority register 20 | IPR20 | 00h | 0008 7320h | 8 |
| Interrupt priority register 21 | IPR21 | 00h | 0008 7321h | 8 |
| Interrupt priority register 22 | IPR22 | 00h | 0008 7322h | 8 |
| Interrupt priority register 23 | IPR23 | 00h | 0008 7323h | 8 |
| Interrupt priority register 24 | IPR24 | 00h | 0008 7324h | 8 |
| Interrupt priority register 25 | IPR25 | 00h | 0008 7325h | 8 |
| Interrupt priority register 26 | IPR26 | 00h | 0008 7326h | 8 |
| Interrupt priority register 27 | IPR27 | 00h | 0008 7327h | 8 |
| Interrupt priority register 28 | IPR28 | 00h | 0008 7328h | 8 |
| Interrupt priority register 29 | IPR29 | 00h | 0008 7329h | 8 |
| Interrupt priority register 2A | IPR2A | 00h | 0008 732Ah | 8 |
| Interrupt priority register 2B | IPR2B | 00h | 0008 732Bh | 8 |
| Interrupt priority register 2C | IPR2C | 00h | 0008 732Ch | 8 |
| Interrupt priority register 2D | IPR2D | 00h | 0008 732Dh | 8 |
| Interrupt priority register 2E | IPR2E | 00h | 0008 732Eh | 8 |
| Interrupt priority register 2F | IPR2F | 00h | 0008 732Fh | 8 |
| Interrupt priority register 40 | IPR40 | 00h | 0008 7340h | 8 |
| Interrupt priority register 44 | IPR44 | 00h | 0008 7344h | 8 |
| Interrupt priority register 45 | IPR45 | 00h | 0008 7345h | 8 |
| Interrupt priority register 46 | IPR46 | 00h | 0008 7346h | 8 |
| Interrupt priority register 47 | IPR47 | 00h | 0008 7347h | 8 |
| Interrupt priority register 4C | IPR4C | 00h | 0008 734Ch | 8 |
| Interrupt priority register 4D | IPR4D | 00h | 0008 734Dh | 8 |
| Interrupt priority register 4E | IPR4E | 00h | 0008 734Eh | 8 |
| Interrupt priority register 4F | IPR4F | 00h | 0008 734Fh | 8 |
| Interrupt priority register 50 | IPR50 | 00h | 0008 7350h | 8 |
| Interrupt priority register 51 | IPR51 | 00h | 0008 7351h | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Interrupt priority register 52 | IPR52 | 00h | 0008 7352h | 8 |
| Interrupt priority register 53 | IPR53 | 00h | 0008 7353h | 8 |
| Interrupt priority register 54 | IPR54 | 00h | 0008 7354h | 8 |
| Interrupt priority register 55 | IPR55 | 00h | 0008 7355h | 8 |
| Interrupt priority register 56 | IPR56 | 00h | 0008 7356h | 8 |
| Interrupt priority register 57 | IPR57 | 00h | 0008 7357h | 8 |
| Interrupt priority register 58 | IPR58 | 00h | 0008 7358h | 8 |
| Interrupt priority register 59 | IPR59 | 00h | 0008 7359h | 8 |
| Interrupt priority register 5A | IPR5A | 00h | 0008 735Ah | 8 |
| Interrupt priority register 5B | IPR5B | 00h | 0008 735Bh | 8 |
| Interrupt priority register 5C | IPR5C | 00h | 0008 735Ch | 8 |
| Interrupt priority register 5D | IPR5D | 00h | 0008 735Dh | 8 |
| Interrupt priority register 5E | IPR5E | 00h | 0008 735Eh | 8 |
| Interrupt priority register 5F | IPR5F | 00h | 0008 735Fh | 8 |
| Interrupt priority register 60 | IPR60 | 00h | 0008 7360h | 8 |
| Interrupt priority register 61 | IPR61 | 00h | 0008 7361h | 8 |
| Interrupt priority register 62 | IPR62 | 00h | 0008 7362h | 8 |
| Interrupt priority register 63 | IPR63 | 00h | 0008 7363h | 8 |
| Interrupt priority register 68 | IPR68 | 00h | 0008 7368h | 8 |
| Interrupt priority register 69 | IPR69 | 00h | 0008 7369h | 8 |
| Interrupt priority register 6A | IPR6A | 00h | 0008 736Ah | 8 |
| Interrupt priority register 6B | IPR6B | 00h | 0008 736Bh | 8 |
| Interrupt priority register 70 | IPR70 | 00h | 0008 7370h | 8 |
| Interrupt priority register 71 | IPR71 | 00h | 0008 7371h | 8 |
| Interrupt priority register 72 | IPR72 | 00h | 0008 7372h | 8 |
| Interrupt priority register 73 | IPR73 | 00h | 0008 7373h | 8 |
| Interrupt priority register 80 | IPR80 | 00h | 0008 7380h | 8 |
| Interrupt priority register 81 | IPR81 | 00h | 0008 7381h | 8 |
| Interrupt priority register 82 | IPR82 | 00h | 0008 7382h | 8 |
| Interrupt priority register 83 | IPR83 | 00h | 0008 7383h | 8 |
| Interrupt priority register 84 | IPR84 | 00h | 0008 7384h | 8 |
| Interrupt priority register 85 | IPR85 | 00h | 0008 7385h | 8 |
| Interrupt priority register 86 | IPR86 | 00h | 0008 7386h | 8 |
| Interrupt priority register 88 | IPR88 | 00h | 0008 7388h | 8 |
| Interrupt priority register 89 | IPR89 | 00h | 0008 7389h | 8 |
| Interrupt priority register 8A | IPR8A | 00h | 0008 738Ah | 8 |
| Interrupt priority register 8B | IPR8B | 00h | 0008 738Bh | 8 |
| Interrupt priority register 8C | IPR8C | 00h | 0008 738Ch | 8 |
| Interrupt priority register 8D | IPR8D | 00h | 0008 738Dh | 8 |
| Interrupt priority register 8E | IPR8E | 00h | 0008 738Eh | 8 |
| Interrupt priority register 8F | IPR8F | 00h | 0008 738Fh | 8 |
| Fast interrupt register | FIR | 0000h | 0008 73F0h | 16 |
| IRQ detection enable register 0 | IRQER0 | 00h | 0008 C300h | 8 |
| IRQ detection enable register 1 | IRQER1 | 00h | 0008 C301h | 8 |
| IRQ detection enable register 2 | IRQER2 | 00h | 0008 C302h | 8 |
| IRQ detection enable register 3 | IRQER3 | 00h | 0008 C303h | 8 |
| IRQ detection enable register 4 | IRQER4 | 00h | 0008 C304h | 8 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| IRQ detection enable register 5 | IRQER5 | 00h | 0008 C305h | 8 |
| IRQ detection enable register 6 | IRQER6 | 00h | 0008 C306h | 8 |
| IRQ detection enable register 7 | IRQER7 | 00h | 0008 C307h | 8 |
| IRQ detection enable register 8 | IRQER8 | 00h | 0008 C308h | 8 |
| IRQ detection enable register 9 | IRQER9 | 00h | 0008 C309h | 8 |
| IRQ detection enable register 10 | IRQER10 | 00h | 0008 C30Ah | 8 |
| IRQ detection enable register 11 | IRQER11 | 00h | 0008 C30Bh | 8 |
| IRQ detection enable register 12 | IRQER12 | 00h | 0008 C30Ch | 8 |
| IRQ detection enable register 13 | IRQER13 | 00h | 0008 C30Dh | 8 |
| IRQ detection enable register 14 | IRQER14 | 00h | 0008 C30Eh | 8 |
| IRQ detection enable register 15 | IRQER15 | 00h | 0008 C30Fh | 8 |
| IRQ control register 0 | IRQCR0 | 00h | 0008 C320h | 8 |
| IRQ control register 1 | IRQCR1 | 00h | 0008 C321h | 8 |
| IRQ control register 2 | IRQCR2 | 00h | 0008 C322h | 8 |
| IRQ control register 3 | IRQCR3 | 00h | 0008 C323h | 8 |
| IRQ control register 4 | IRQCR4 | 00h | 0008 C324h | 8 |
| IRQ control register 5 | IRQCR5 | 00h | 0008 C325h | 8 |
| IRQ control register 6 | IRQCR6 | 00h | 0008 C326h | 8 |
| IRQ control register 7 | IRQCR7 | 00h | 0008 C327h | 8 |
| IRQ control register 8 | IRQCR8 | 00h | 0008 C328h | 8 |
| IRQ control register 9 | IRQCR9 | 00h | 0008 C329h | 8 |
| IRQ control register 10 | IRQCR10 | 00h | 0008 C32Ah | 8 |
| IRQ control register 11 | IRQCR11 | 00h | 0008 C32Bh | 8 |
| IRQ control register 12 | IRQCR12 | 00h | 0008 C32Ch | 8 |
| IRQ control register 13 | IRQCR13 | 00h | 0008 C32Dh | 8 |
| IRQ control register 14 | IRQCR14 | 00h | 0008 C32Eh | 8 |
| IRQ control register 15 | IRQCR15 | 00h | 0008 C32Fh | 8 |
| Software standby release IRQ enable register | SSIER | 0000h | 0008 C340h | 16 |
| Non-maskable interrupt enable register | NMIER | 00h | 0008 C350h | 8 |
| NMI pin interrupt control register | NMICR | 00h | 0008 C351h | 8 |
| Non-maskable interrupt status register | NMISR | 00h | 0008 C352h | 8 |
| Non-maskable interrupt clear register | NMICLR | 00h | 0008 C353h | 8 |



Downloaded from Arrow.com.

## 10.2.1 Interrupt Request Register i (IRi) (i = interrupt vector number)

Addresses: 0008 7010h to 0008 70FDh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | IR |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | IR | Interrupt Status Flag | 0: No interrupt request is generated<br>1: An interrupt request is generated | R/(W)* |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

Note: * In the case of the edge detection, only 0 can be written to clear the flag.
Writing 1 is enabled under the conditions described in section 10.7.3, Notes on DMAC/DTC Transfer using Communication Function (SCI, RIIC).
In the case of the level detection, writing to this flag is disabled.

The IRi register indicates interrupt request status.

IRi is provided for each interrupt source, where "i" shows an interrupt vector number.

For the correspondence between interrupt sources and interrupt vector numbers, see table 10.4, Interrupt Vector Table.

**IR Flag (Interrupt Status Flag)**

This bit is the status flag for the corresponding interrupt request. If the flag is set to 1 while interrupt requests are enabled by the IENj bit in IERi, an interrupt request signal is output to the destination that has been specified by the ISEL[1:0] bits in ISELRi.

When an interrupt request from the source is detected in the source of interrupt generation, the flag is set to 1. For the actual interrupt to be generated, the interrupt enable bit of the corresponding peripheral module must be set to enable interrupt output or detection of the interrupt signal on the given pin must be enabled by the IRQEN bit in IRQERn for interrupts on pin IRQn.

Interrupts are detected as either an edge or level of the interrupt signal. For details on interrupt detection, see section 10.4.2, Interrupt Status Flag.

- Peripheral module interrupts (except IR064 to IR079)

  Edge detection

  [Setting condition]

  - Generation of the interrupt signal from the source

  [Clearing conditions]

  - Writing of a 0 to the flag
    However, when the DTC or the DMAC is specified as an interrupt request destination, writing 0 to the IR flag is prohibited.
  - If the setting of the ISEL[1:0] bits in ISELRi is 00b, interrupt exception handling by the CPU
  - If the setting of the ISEL[1:0] bits in ISELRi is 01b and the setting of the DISEL bit in MRB of the DTC is 0, activation of the DTC



Downloaded from Arrow.com.

- If the setting of the ISEL[1:0] bits in ISELRi is 10b, activation of the DMAC

Level detection

[Setting condition]

- Generation of the interrupt signal from the source

[Clearing conditions]

- Clearing the status flag of the interrupt source
- Setting the interrupt enable bit of the corresponding interrupt source to disable the output of interrupt requests

- External interrupts (IR064 to IR079)

Edge detection (IRQMD[1:0] bits in IRQCRn = 01b/10b/11b)

[Setting condition]

- Generation of the interrupt signal from the source

[Clearing condition]

- Writing of a 0 to the flag
- If the setting of the ISEL[1:0] bits in ISELRi is 00b, interrupt exception handling by the CPU
- If the setting of the ISEL[1:0] bits in ISELRi is 01b and the setting of the DISEL bit in MRB of the DTC is 0, activation of the DTC
- If the setting of the ISEL[1:0] bits in ISELRi is 10b, activation of the DMAC

Level detection (IRQMD[1:0] bits in IRQCRn = 00b)

[Setting condition]

- Generation of the interrupt signal from the source

[Clearing conditions]

- Writing of 0 with the IRQn pin at the high level *1, *2
- Disabling the detection of interrupts by the setting of the IRQEN bit in IRQERn of pin IRQn

Notes: 1. For details on clearing in the case of level-detected external interrupts, see section 10.4.2.2, Interrupt Status Flag in Level Detection.

2. Do not write 0 to the corresponding IR flag while the signal on an IRQn pin is at the low level. While doing so temporarily clears the IR flag to 0 and withdraws the interrupt request, the IR flag is returned to 1 to generate the interrupt request again after two clock cycles of PCLK have elapsed.



Downloaded from Arrow.com.

## 10.2.2 Interrupt Request Destination Setting Register i (ISELRi) (i = interrupt vector number)

Addresses: 0008 711Ch to 0008 71FDh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | ISEL[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | ISEL[1:0] | Interrupt Destination | b1 b0<br>0 0: The request is sent to the CPU.<br>0 1: The request activates the DTC and is passed to the CPU on completion of data transfer. *1<br>1 0: The request activates the DMAC.<br>1 1: The request activates the DMAC and is then passed to the CPU. *1 | R/W*2 |
| b7 to b2 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

Notes: 1. The ISEL[1:0] bits are automatically cleared to 00b on completion of DTC data transfer or DMAC activation. For details, refer to section 10.4.3, Selecting Interrupt Request Destinations.

2. The DMAC cannot be selected as an interrupt request destination for some interrupt sources. In such cases, the lowest 1 bit is valid and the upper 7 bits are reserved. The reserved bits are read as 0. The write value to the reserved bits should always be 0.

The ISELRi register is used to set the destination of an interrupt request.

### ISEL[1:0] Bits (Interrupt Destination)

These bits specify the destination of an interrupt request.

The ISEL[1:0] bits should be modified after the interrupt request is disabled by the setting of the IENj bit in IERi.

The interrupt request destinations available for specification by the ISEL[1:0] bits depend on the interrupt source. For the correspondence between specifiable interrupt request destinations and interrupt sources, see table 10.4, Interrupt Vector Table.

If the setting of the IENj bit in IERi enables the interrupt request and the IR flag in the corresponding IRi is set to 1, the interrupt request is output to the destination that has been specified by the ISEL[1:0] bits. For details, see section 10.4.3, Selecting Interrupt Request Destinations.



Downloaded from Arrow.com.

### 10.2.3 Interrupt Request Enable Register m (IERi) (i = 02h to 1Fh)

Addresses: 0008 7202h to 0008 721Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | IEN7 | IEN6 | IEN5 | IEN4 | IEN3 | IEN2 | IEN1 | IEN0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W* |
|---|---|---|---|---|
| b0 | IEN0 | Interrupt Request Enable 0 | 0: Interrupt request is disabled<br>1: Interrupt request is enabled | R/W |
| b1 | IEN1 | Interrupt Request Enable 1 | | R/W |
| b2 | IEN2 | Interrupt Request Enable 2 | | R/W |
| b3 | IEN3 | Interrupt Request Enable 3 | | R/W |
| b4 | IEN4 | Interrupt Request Enable 4 | | R/W |
| b5 | IEN5 | Interrupt Request Enable 5 | | R/W |
| b6 | IEN6 | Interrupt Request Enable 6 | | R/W |
| b7 | IEN7 | Interrupt Request Enable 7 | | R/W |

Note: * Write 0 to the bit that corresponds to the interrupt source for reservation. Such a bit is read as 0.

The IERm register is used to enable or disable an interrupt request to the CPU and DMAC/DTC activation request.

#### IENj Bits (Interrupt Request Enable) (j = 0 to 7)

There is an interrupt request enable bit for each interrupt source.

For the correspondence between interrupt sources and interrupt request enable bits, see table 10.4, Interrupt Vector Table.

When an IENj bit is 1, the corresponding interrupt request is enabled. When an IENj bit is 0, the corresponding interrupt request is disabled.

The IRi.IR flag is not affected by the IENj bit. Even if the IENj bit is 0, the IR flag changes under the conditions described in section 10.2.1, Interrupt Request Register i (IRi) (i = interrupt vector number). The correspondence between the interrupt sources and the IERm.IENj bits, refer to table 10.4, Interrupt Vector Table. For IERm.IENj bit setting procedure for selection of the interrupt request destination, refer to section 10.4.3, Selecting Interrupt Request Destinations.



Downloaded from Arrow.com.

## 10.2.4 Interrupt Priority Register i (IPRi) (i = 00h to 8Fh)

Addresses: 0008 7300h to 0008 738Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | IPR[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | IPR[2:0] | Interrupt Priority Level Select | b2 b0<br>0 0 0: Level 0 (interrupt prohibited)<br>0 0 1: Level 1<br>0 1 0: Level 2<br>0 1 1: Level 3<br>1 0 0: Level 4<br>1 0 1: Level 5<br>1 1 0: Level 6<br>1 1 1: Level 7 (highest) | R/W |
| b7 to b3 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

The IPRi register is used to set the priority level of an interrupt source.

IPRi is provided for each interrupt source group, where "i" is a serial number (00 to 8F) in hexadecimal notation.

For the correspondence between interrupt sources and groups, see table 10.4, Interrupt Vector Table.

**IPR[2:0] Bits (Interrupt Priority Level Select)**

These bits specify the priority level of interrupt requests.

Priority levels specified by the IPR[2:0] bits are used only to determine the priority of interrupt requests to the CPU, and do not affect activation requests to the DTC and DMAC.

The CPU accepts only interrupt requests higher than the priority level specified by the IPL[2:0] bits in PSW, and handles accepted interrupts.

If two or more interrupt requests are generated at the same time, their priority levels are compared with the value of the IPR[2:0] bits. If interrupt requests of the same priority level are generated at the same time, an interrupt request with a smaller vector number takes precedence.

Write to these bits while the interrupt request is disabled (IERm.IENj bit is 0).



Downloaded from Arrow.com.

## 10.2.5 Fast Interrupt Register (FIR)

Address: 0008 73F0h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | FIEN | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FVCT[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | FVCT[7:0] | Fast Interrupt Vector Number | Specify the vector number of an interrupt source to be a fast interrupt. | R/W |
| b14 to b8 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |
| b15 | FIEN | Fast Interrupt Enable | 0: Fast interrupt is disabled<br>1: Fast interrupt is enabled | R/W |

The FIR register is used to set up the fast interrupt.

The fast interrupt settings are only applicable to an interrupt request for the CPU. That is, they do not affect activation requests for the DTC and DMAC.

Write to these bits while the interrupt request is disabled (IERm.IENj bit is 0).

### FVCT[7:0] Bits (Fast Interrupt Vector Number)

The FVCT[7:0] bits specify the vector number of a source for use as the fast interrupt.

When the CPU is the interrupt request destination and an interrupt request corresponding to the vector number specified with the FVCT[7:0] bits is generated while the FIEN bit is 1, the interrupt request is output to the CPU as the fast interrupt regardless of the setting of the IPRi register. However, to use a fast interrupt for recovery from software standby mode, refer to section 10.6.2, Returning from Software Standby Mode.

When the setting of the IENj bit in IERi has disabled interrupt requests from the interrupt source, such interrupt requests are not output to the CPU.

For settable vector numbers, see table 10.4, Interrupt Vector Table.

Do not write a reserved vector number to these bits.

### FIEN Bit (Fast Interrupt Enable)

This bit enables the fast interrupt.

Setting this bit to 1 makes the interrupt request corresponding to the vector number specified in the FVCT[7:0] bits operate as the fast interrupt.

For details on the fast interrupt, see sections 9, Exceptions and 10.4.5, Fast Interrupt.



Downloaded from Arrow.com.

## 10.2.6 IRQ Detection Enable Register n (IRQERn) (n = 0 to 15)

Addresses: 0008 C300h to 0008 C30Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | IRQEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | IRQEN | IRQ Detection Enable | 0: Detection of the signal on the corresponding IRQn pin as an external interrupt source is disabled<br>1: Detection of the signal on the corresponding IRQn pin as an external interrupt source is enabled<br>(n = 0 to 15) | R/W |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

The IRQERn register is used to enable or disable the detection of the corresponding external interrupt source (interrupt on the corresponding IRQn pin; n = 0 to 15).

**IRQEN Bit (IRQ Enable)**

This bit specifies whether to enable or disable the detection of the corresponding external interrupt source (interrupt on the corresponding IRQn pin; n = 0 to 15).



Downloaded from Arrow.com.

## 10.2.7 IRQ Control Register n (IRQCRn) (n = 0 to 15)

Addresses: 0008 C320h to 0008 C32Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | IRQMD[1:0] | | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |
| b3, b2 | IRQMD[1:0] | IRQ Detection Select | b3 b2<br>0 0: Low level<br>0 1: Falling edge<br>1 0: Rising edge<br>1 1: Rising and falling edges | R/W |
| b7 to b4 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

The IRQCRn register is used to set up the external interrupt IRQn pin (n = 0 to 15).

The contents of this register should be modified while the corresponding interrupt request enable bit is set to disable an interrupt request (IERm.IENj bit is 0). After modification of the contents of this register, the IR flag should be cleared, and then the interrupt request enable bit should be set to enable the interrupt request.

### IRQMD[1:0] Bits (IRQ Detection Sense Select)

These bits select the detection method for external interrupt source IRQn, where n = 0 to 15.

For setting to detect the corresponding external interrupt source, see section 10.4.6, External Interrupts.



Downloaded from Arrow.com.

## 10.2.8 Non-maskable Interrupt Enable Register (NMIER)

Address: 0008 C350h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | NMIEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NMIEN | NMI Enable | 0: NMI pin interrupt is disabled<br>1: NMI pin interrupt is enabled | R/W* |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

Note: * A 1 can be written to this bit only once, and subsequent write accesses are no longer enabled.

The NMIER register is used to enable the non-maskable interrupt.

**NMIEN Bit (NMI Enable)**

This bit enables interrupts by the signal on the NMI pin.

A 1 can be written to this bit only once. Once the interrupt has been enabled, further write access is not possible.

Do not write 0 to this bit. The NMI cannot be disabled once it has been enabled.



Downloaded from Arrow.com.

## 10.2.9 NMI Pin Interrupt Control Register (NMICR)

Address: 0008 C351h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | NMIMD | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |
| b3 | NMIMD | NMI Detection Select | 0: Falling edge<br>1: Rising edge | R/W |
| b7 to b4 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

The NMICR register is used to set up the NMI (interrupt on the NMI pin).

Change the setting before enabling the NMI (setting the NMIEN bit in NMIER to 1).

**NMIMD Bit (NMI Detection Sense Select)**

This bit sets the detection method for the NMI.



Downloaded from Arrow.com.

## 10.2.10 Non-maskable Interrupt Status Register (NMISR)

Address: 0008 C352h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | NMIST |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NMIST | NMI Status Flag | 0: No NMI pin request is generated<br>1: An NMI pin request is generated | R |
| b7 to b1 | — | Reserved | These bits are read as 0 and cannot be modified. | R |

The NMISR register is used to monitor the non-maskable interrupt status.

To clear the NMISR.NMIST flag to 0, set the NMICLR.NMICLR bit to 1. After that, confirm that the NMISR.NMIST flag is 0, and then execute the next instruction.

**NMIST Flag (NMI Status Flag)**

The NMIST flag indicates the NMI pin interrupt request.

This is a read-only flag and is cleared by the NMICR bit in NMICLR.

[Setting condition]

- This flag is set to 1 when an edge specified by the NMIMD bit in NMICR is input to the NMI pin while the NMIEN bit in NMIER is 1 (NMI pin interrupt enable).

[Clearing condition]

- This flag is cleared to 0 by writing 1 to the NMICLR bit in NMICLR.



Downloaded from Arrow.com.

## 10.2.11 Non-maskable Interrupt Clear Register (NMICLR)

Address: 0008 C353h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | NMICLR |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NMICLR | NMI Clear | This bit is always read as 0.<br>Writing 1 to this bit clears the NMIST flag in NMISR.<br>Writing 0 to this bit has no effect. | R/W* |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

Note: * Only 1 can be written to this bit.

The NMICLR register is used to clear the non-maskable interrupt status register (NMISR).

**NMICLR Bit (NMI Clear)**

Writing 1 to this bit clears the NMIST flag in NMISR.

The NMICLR bit does not retain the "1" state. This bit is always read as 0.



Downloaded from Arrow.com.

## 10.2.12 Software Standby Release IRQ Enable Register (SSIER)

Address: 0008 C340h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | SSI15 | SSI14 | SSI13 | SSI12 | SSI11 | SSI10 | SSI9 | SSI8 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | SSI7 | SSI6 | SSI5 | SSI4 | SSI3 | SSI2 | SSI1 | SSI0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SSI0 | Software Standby Release IRQ | 0: The signal on the corresponding IRQn pin as an external interrupt source is not sampled in the software standby state.<br>1: When the signal on the corresponding IRQn pin as an external interrupt source is generated in the software standby state, the interrupt control unit returns from the software standby state after the oscillation settling time. (n = 0 to 15) | R/W |
| b1 | SSI1 | | | R/W |
| b2 | SSI2 | | | R/W |
| b3 | SSI3 | | | R/W |
| b4 | SSI4 | | | R/W |
| b5 | SSI5 | | | R/W |
| b6 | SSI6 | | | R/W |
| b7 | SSI7 | | | R/W |
| b8 | SSI8 | | | R/W |
| b9 | SSI9 | | | R/W |
| b10 | SSI10 | | | R/W |
| b11 | SSI11 | | | R/W |
| b12 | SSI12 | | | R/W |
| b13 | SSI13 | | | R/W |
| b14 | SSI14 | | | R/W |
| b15 | SSI15 | | | R/W |

The SSIER register is used to set whether to use the IRQn pin (n = 0 to 15) as a source that allows the interrupt control unit to return from the software standby state.

**SSIj Bits (j = 0 to 15) (Software Standby Release IRQ)**

These bits specify whether to use the IRQn pin corresponding to the bit number as a source that allows the interrupt control unit to return from the software standby state.

For how to set up return from the software standby state, see section 10.6.2, Returning from Software Standby Mode.



Downloaded from Arrow.com.

## 10.3 Vector Table

The interrupt control unit detects two types of interrupt exceptions: maskable interrupts and the non-maskable interrupt.

When the CPU accepts an interrupt or non-maskable interrupt, it acquires a four-byte vector address from the vector table.

### 10.3.1 Interrupt Vector Table

The interrupt vector table is placed in the 1024-byte range (4 bytes x 256 sources) beginning at the address specified in the interrupt table register (INTB) of the CPU. Write a value to the INTB register before enabling interrupts. Setting this to a multiple of four speeds up the execution of interrupt exception handling.

Table 10.4 shows the interrupt vector table. "Sstb recovery" means recovery from software (S/W) standby mode, and "Sacs recovery" means recovery from all-module clock stop mode.

**Table 10.4 Interrupt Vector Table**

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High | — | Reserved | 0 | 0000h | — | × | × | × | × | × | — | — |
| ↑ | | Reserved | 1 | 0004h | — | × | × | × | × | × | — | — |
| | | Reserved | 2 | 0008h | — | × | × | × | × | × | — | — |
| | | Reserved | 3 | 000Ch | — | × | × | × | × | × | — | — |
| | | Reserved | 4 | 0010h | — | × | × | × | × | × | — | — |
| | | Reserved | 5 | 0014h | — | × | × | × | × | × | — | — |
| | | Reserved | 6 | 0018h | — | × | × | × | × | × | — | — |
| | | Reserved | 7 | 001Ch | — | × | × | × | × | × | — | — |
| | | Reserved | 8 | 0020h | — | × | × | × | × | × | — | — |
| | | Reserved | 9 to 15 | 0024h to 003Ch | — | × | × | × | × | × | — | — |
| | Bus error | BUSERR | 16 | 0040h | Level | √ | × | × | × | × | ER02.IEN0 | IPR00 |
| | — | Reserved | 17 | 0044h | — | × | × | × | × | × | IER02. EN1 | — |
| | | Reserved | 18 | 0048h | — | × | × | × | × | × | IER02. EN2 | — |
| | | Reserved | 19 | 004Ch | — | × | × | × | × | × | IER02. EN3 | — |
| | | Reserved | 20 | 0050h | — | × | × | × | × | × | IER02. EN4 | — |
| | FCU | FIFERR | 21 | 0054h | Level | √ | × | × | × | × | ER02.IEN5 | IPR01 |
| | | Reserved | 22 | 0058h | — | × | × | × | × | × | ER02.IEN6 | — |
| | | FRDYI | 23 | 005Ch | Edge | √ | × | × | × | × | ER02.IEN7 | IPR02 |
| | — | Reserved | 24 | 0060h | — | × | × | × | × | × | ER03.IEN0 | — |
| | | Reserved | 25 | 0064h | — | × | × | × | × | × | ER03.IEN1 | — |
| | | Reserved | 26 | 0068h | — | × | × | × | × | × | ER03.IEN2 | — |
| | | Reserved | 27 | 006Ch | — | × | × | × | × | × | ER03.IEN3 | — |
| | CMT | CMT0 | 28 | 0070h | Edge | √ | √ | √ | × | × | ER03.IEN4 | IPR04 |
| | unit 0 | CMT1 | 29 | 0074h | Edge | √ | √ | √ | × | × | ER03.IEN5 | IPR05 |
| | CMT | CMT2 | 30 | 0078h | Edge | √ | √ | √ | × | × | ER03.IEN6 | IPR06 |
| ↓ | unit 1 | CMT3 | 31 | 007Ch | Edge | √ | √ | √ | × | × | ER03.IEN7 | IPR07 |
| Low | — | Reserved | 32 to 63 | 0080h to 00FCh | — | × | × | × | × | × | — | — |



Downloaded from Arrow.com.

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High | External | IRQ0 | 64 | 0100h | Edge/Level | √ | √ | √ | √ | √ | ER08.IEN0 | IPR20 |
| ↑ | pin | IRQ1 | 65 | 0104h | Edge/Level | √ | √ | √ | √ | √ | ER08.IEN1 | IPR21 |
| | | IRQ2 | 66 | 0108h | Edge/Level | √ | √ | √ | √ | √ | ER08.IEN2 | IPR22 |
| | | IRQ3 | 67 | 010Ch | Edge/Level | √ | √ | √ | √ | √ | ER08.IEN3 | IPR23 |
| | | IRQ4 | 68 | 0110h | Edge/Level | √ | √ | × | √ | √ | ER08.IEN4 | IPR24 |
| | | IRQ5 | 69 | 0114h | Edge/Level | √ | √ | × | √ | √ | ER08.IEN5 | IPR25 |
| | | IRQ6 | 70 | 0118h | Edge/Level | √ | √ | × | √ | √ | ER08.IEN6 | IPR26 |
| | | IRQ7 | 71 | 011Ch | Edge/Level | √ | √ | × | √ | √ | ER08.IEN7 | IPR27 |
| | | IRQ8 | 72 | 0120h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN0 | IPR28 |
| | | IRQ9 | 73 | 0124h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN1 | IPR29 |
| | | IRQ10 | 74 | 0128h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN2 | IPR2A |
| | | IRQ11 | 75 | 012Ch | Edge/Level | √ | √ | × | √ | √ | ER09.IEN3 | IPR2B |
| | | IRQ12 | 76 | 0130h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN4 | IPR2C |
| | | IRQ13 | 77 | 0134h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN5 | IPR2D |
| | | IRQ14 | 78 | 0138h | Edge/Level | √ | √ | × | √ | √ | ER09.IEN6 | IPR2E |
| | | IRQ15 | 79 | 013Ch | Edge/Level | √ | √ | × | √ | √ | ER09.IEN7 | IPR2F |
| | — | Reserved | 80 to 95 | 0140h to 017Ch | — | × | × | × | × | × | — | — |
| | WDT | WOVI | 96 | 0180h | Edge | √ | × | × | × | √ | ER0C.IEN0 | IPR40 |
| | | Reserved | 97 | 0184h | — | × | × | × | × | × | ER0C.IEN1 | — |
| | AD0 | ADI0 | 98 | 0188h | Edge | √ | √ | √ | × | × | ER0C.IEN2 | IPR44 |
| | AD1 | ADI1 | 99 | 018Ch | Edge | √ | √ | √ | × | × | ER0C.IEN3 | IPR45 |
| | AD2 | ADI2 | 100 | 0190h | Edge | √ | √ | √ | × | × | ER0C.IEN4 | IPR46 |
| | AD3 | ADI3 | 101 | 0194h | Edge | √ | √ | √ | × | × | ER0C.IEN5 | IPR47 |
| | — | Reserved | 102 | 0198h | — | × | × | × | × | × | ER0C.IEN6 | — |
| | | Reserved | 103 | 019Ch | — | × | × | × | × | × | ER0C.IEN7 | — |
| | TPU0 | TGI0A | 104 | 01A0h | Edge | √ | √ | √ | × | × | ER0D.IEN0 | IPR4C |
| | | TGI0B | 105 | 01A4h | Edge | √ | √ | × | × | × | ER0D.IEN1 | |
| | | TGI0C | 106 | 01A8h | Edge | √ | √ | × | × | × | ER0D.IEN2 | |
| | | TGI0D | 107 | 01ACh | Edge | √ | √ | × | × | × | ER0D.IEN3 | |
| | | TC 0V | 108 | 01B0h | Edge | √ | × | × | × | × | ER0D.IEN4 | IPR4D |
| | | Reserved | 109 | 01B4h | — | × | × | × | × | × | ER0D.IEN5 | — |
| | | Reserved | 110 | 01B8h | — | × | × | × | × | × | ER0D.IEN6 | — |
| | TPU1 | TGI1A | 111 | 01BCh | Edge | √ | √ | √ | × | × | ER0D.IEN7 | IPR4E |
| | | TGI1B | 112 | 01C0h | Edge | √ | √ | × | × | × | ER0E. EN0 | |
| | | Reserved | 113 | 01C4h | — | × | × | × | × | × | ER0E. EN1 | — |
| | | Reserved | 114 | 01C8h | — | × | × | × | × | × | ER0E. EN2 | — |
| | | TCI1V | 115 | 01CCh | Edge | √ | × | × | × | × | ER0E. EN3 | IPR4F |
| | | TCI1U | 116 | 01D0h | Edge | √ | × | × | × | × | ER0E. EN4 | |
| | TPU2 | TGI2A | 117 | 01D4h | Edge | √ | √ | √ | × | × | ER0E. EN5 | IPR50 |
| | | TGI2B | 118 | 01D8h | Edge | √ | √ | × | × | × | ER0E. EN6 | |
| | | Reserved | 119 | 01DCh | — | × | × | × | × | × | ER0E. EN7 | — |
| ↓ | | TCI2V | 120 | 01E0h | Edge | √ | × | × | × | × | ER0F.IEN0 | IPR51 |
| Low | | TCI2U | 121 | 01E4h | Edge | √ | × | × | × | × | ER0F.IEN1 | |



Downloaded from Arrow.com.

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High ↑ | TPU3 | TGI3A | 122 | 01E8h | Edge | √ | √ | √ | × | × | ER0F.IEN2 | IPR52 |
| | | TGI3B | 123 | 01ECh | Edge | √ | √ | × | × | × | ER0F.IEN3 | |
| | | TGI3C | 124 | 01F0h | Edge | √ | √ | × | × | × | ER0F.IEN4 | |
| | | TGI3D | 125 | 01F4h | Edge | √ | √ | × | × | × | ER0F.IEN5 | |
| | | TCI3V | 126 | 01F8h | Edge | √ | × | × | × | × | ER0F.IEN6 | IPR53 |
| | TPU4 | TGI4A | 127 | 01FCh | Edge | √ | √ | √ | × | × | ER0F.IEN7 | IPR54 |
| | | TGI4B | 128 | 0200h | Edge | √ | √ | × | × | × | ER10.IEN0 | |
| | | Reserved | 129 | 0204h | — | × | × | × | × | × | ER10.IEN1 | — |
| | | Reserved | 130 | 0208h | — | × | × | × | × | × | ER10.IEN2 | — |
| | | TCI4V | 131 | 020Ch | Edge | √ | × | × | × | × | ER10.IEN3 | IPR55 |
| | | TCI4U | 132 | 0210h | Edge | √ | × | × | × | × | ER10.IEN4 | |
| | TPU5 | TGI5A | 133 | 0214h | Edge | √ | √ | √ | × | × | ER10.IEN5 | IPR56 |
| | | TGI5B | 134 | 0218h | Edge | √ | √ | × | × | × | ER10.IEN6 | |
| | | Reserved | 135 | 021Ch | — | × | × | × | × | × | ER10.IEN7 | — |
| | | TCI5V | 136 | 0220h | Edge | √ | × | × | × | × | ER11.IEN0 | IPR57 |
| | | TCI5U | 137 | 0224h | Edge | √ | × | × | × | × | ER11.IEN1 | |
| | TPU6 | TGI6A | 138 | 0228h | Edge | √ | √ | √ | × | × | ER11.IEN2 | IPR58 |
| | | TGI6B | 139 | 022Ch | Edge | √ | √ | × | × | × | ER11.IEN3 | |
| | | TGI6C | 140 | 0230h | Edge | √ | √ | × | × | × | ER11.IEN4 | |
| | | TGI6D | 141 | 0234h | Edge | √ | √ | × | × | × | ER11.IEN5 | |
| | | TCI6V | 142 | 0238h | Edge | √ | × | × | × | × | ER11.IEN6 | IPR59 |
| | | Reserved | 143 | 023Ch | — | × | × | × | × | × | ER11.IEN7 | — |
| | | Reserved | 144 | 0240h | — | × | × | × | × | × | ER12.IEN0 | — |
| | TPU7 | TGI7A | 145 | 0244h | Edge | √ | √ | √ | × | × | ER12.IEN1 | IPR5A |
| | | TGI7B | 146 | 0248h | Edge | √ | √ | × | × | × | ER12.IEN2 | |
| | | Reserved | 147 | 024Ch | — | × | × | × | × | × | ER12.IEN3 | — |
| | | Reserved | 148 | 0250h | — | × | × | × | × | × | ER12.IEN4 | — |
| | | TCI7V | 149 | 0254h | Edge | √ | × | × | × | × | ER12.IEN5 | IPR5B |
| | | TCI7U | 150 | 0258h | Edge | √ | × | × | × | × | ER12.IEN6 | |
| | TPU8 | TGI8A | 151 | 025Ch | Edge | √ | √ | √ | × | × | ER12.IEN7 | IPR5C |
| | | TGI8B | 152 | 0260h | Edge | √ | √ | × | × | × | ER13.IEN0 | |
| | | Reserved | 153 | 0264h | — | × | × | × | × | × | ER13.IEN1 | — |
| | | TCI8V | 154 | 0268h | Edge | √ | × | × | × | × | ER13.IEN2 | IPR5D |
| | | TCI8U | 155 | 026Ch | Edge | √ | × | × | × | × | ER13.IEN3 | |
| | TPU9 | TGI9A | 156 | 0270h | Edge | √ | √ | √ | × | × | ER13.IEN4 | IPR5E |
| | | TGI9B | 157 | 0274h | Edge | √ | √ | × | × | × | ER13.IEN5 | |
| | | TGI9C | 158 | 0278h | Edge | √ | √ | × | × | × | ER13.IEN6 | |
| | | TGI9D | 159 | 027Ch | Edge | √ | √ | × | × | × | ER13.IEN7 | |
| | | TCI9V | 160 | 0280h | Edge | √ | × | × | × | × | ER14.IEN0 | IPR5F |
| | TPU10 | TGI10A | 161 | 0284h | Edge | √ | √ | √ | × | × | IER14. EN1 | IPR60 |
| | | TGI10B | 162 | 0288h | Edge | √ | √ | × | × | × | ER14.IEN2 | |
| | | Reserved | 163 | 028Ch | — | × | × | × | × | × | ER14.IEN3 | — |
| | | Reserved | 164 | 0290h | — | × | × | × | × | × | ER14.IEN4 | — |
| ↓ Low | | TCI10V | 165 | 0294h | Edge | √ | × | × | × | × | ER14.IEN5 | IPR61 |
| | | TCI10U | 166 | 0298h | Edge | √ | × | × | × | × | ER14.IEN6 | |



Downloaded from Arrow.com.

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High ↑ | TPU11 | TGI11A | 167 | 029Ch | Edge | √ | √ | √ | × | × | IER14. EN7 | PR62 |
| | | TGI11B | 168 | 02A0h | Edge | √ | √ | × | × | × | IER15. EN0 | |
| | | Reserved | 169 | 02A4h | — | × | × | × | × | × | IER15. EN1 | — |
| | | TCI11V | 170 | 02A8h | Edge | √ | × | × | × | × | IER15. EN2 | IPR63 |
| | | TCI11U | 171 | 02ACh | Edge | √ | × | × | × | × | IER15. EN3 | |
| | — | Reserved | 172 | 02B0h | — | × | × | × | × | × | IER15. EN4 | — |
| | | Reserved | 173 | 02B4h | — | × | × | × | × | × | IER15. EN5 | — |
| | TMR0 | CMIA0 | 174 | 02B8h | Edge | √ | √ | × | × | √ | IER15. EN6 | IPR68 |
| | | CMIB0 | 175 | 02BCh | Edge | √ | √ | × | × | √ | IER15. EN7 | |
| | | OV 0 | 176 | 02C0h | Edge | √ | × | × | × | √ | IER16. EN0 | |
| | TMR1 | CMIA1 | 177 | 02C4h | Edge | √ | √ | × | × | √ | IER16. EN1 | IPR69 |
| | | CMIB1 | 178 | 02C8h | Edge | √ | √ | × | × | √ | IER16. EN2 | |
| | | OVI1 | 179 | 02CCh | Edge | √ | × | × | × | √ | IER16. EN3 | |
| | TMR2 | CMIA2 | 180 | 02D0h | Edge | √ | √ | × | × | √ | IER16. EN4 | IPR6A |
| | | CMIB2 | 181 | 02D4h | Edge | √ | √ | × | × | √ | IER16. EN5 | |
| | | OVI2 | 182 | 02D8h | Edge | √ | × | × | × | √ | IER16. EN6 | |
| | TMR3 | CMIA3 | 183 | 02DCh | Edge | √ | √ | × | × | √ | IER16. EN7 | IPR6B |
| | | CMIB3 | 184 | 02E0h | Edge | √ | √ | × | × | √ | IER17. EN0 | |
| | | OVI3 | 185 | 02E4h | Edge | √ | × | × | × | √ | IER17. EN1 | |
| | — | Reserved | 186 | 02E8h | — | × | × | × | × | × | IER17. EN2 | — |
| | | Reserved | 187 | 02ECh | — | × | × | × | × | × | IER17. EN3 | — |
| | | Reserved | 188 | 02F0h | — | × | × | × | × | × | IER17. EN4 | — |
| | | Reserved | 189 | 02F4h | — | × | × | × | × | × | IER17. EN5 | — |
| | | Reserved | 190 | 02F8h | — | × | × | × | × | × | IER17. EN6 | — |
| | | Reserved | 191 | 02FCh | — | × | × | × | × | × | IER17. EN7 | — |
| | | Reserved | 192 | 0300h | — | × | × | × | × | × | IER18. EN0 | — |
| | | Reserved | 193 | 0304h | — | × | × | × | × | × | IER18. EN1 | — |
| | | Reserved | 194 | 0308h | — | × | × | × | × | × | IER18. EN2 | — |
| | | Reserved | 195 | 030Ch | — | × | × | × | × | × | IER18. EN3 | — |
| | | Reserved | 196 | 0310h | — | × | × | × | × | × | IER18. EN4 | — |
| | | Reserved | 197 | 0314h | — | × | × | × | × | × | IER18. EN5 | — |
| | DMAC | DMTEND0 | 198 | 0318h | Edge | √ | √ | × | × | × | IER18. EN6 | PR70 |
| | | DMTEND1 | 199 | 031Ch | Edge | √ | √ | × | × | × | IER18. EN7 | PR71 |
| | | DMTEND2 | 200 | 0320h | Edge | √ | √ | × | × | × | IER19. EN0 | PR72 |
| | | DMTEND3 | 201 | 0324h | Edge | √ | √ | × | × | × | IER19. EN1 | PR73 |
| | — | Reserved | 202 | 0328h | — | × | × | × | × | × | IER19. EN2 | — |
| | | Reserved | 203 | 032Ch | — | × | × | × | × | × | IER19. EN3 | — |
| | | Reserved | 204 | 0330h | — | × | × | × | × | × | IER19. EN4 | — |
| | | Reserved | 205 | 0334h | — | × | × | × | × | × | IER19. EN5 | — |
| ↓ Low | | Reserved | 206 | 0338h | — | × | × | × | × | × | IER19. EN6 | — |



Downloaded from Arrow.com.

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High ↑ | — | Reserved | 207 | 033Ch | — | × | × | × | × | × | IER19. EN7 | — |
| | | Reserved | 208 | 0340h | — | × | × | × | × | × | IER1A.IEN0 | — |
| | | Reserved | 209 | 0344h | — | × | × | × | × | × | IER1A.IEN1 | — |
| | | Reserved | 210 | 0348h | — | × | × | × | × | × | IER1A.IEN2 | — |
| | | Reserved | 211 | 034Ch | — | × | × | × | × | × | IER1A.IEN3 | — |
| | | Reserved | 212 | 0350h | — | × | × | × | × | × | IER1A.IEN4 | — |
| | | Reserved | 213 | 0354h | — | × | × | × | × | × | IER1A.IEN5 | — |
| | SCI0 | ERI0 | 214 | 0358h | Level | √ | × | × | × | × | IER1A.IEN6 | IPR80 |
| | | RXI0 | 215 | 035Ch | Edge | √ | √ | √ | × | × | IER1A.IEN7 | |
| | | TXI0 | 216 | 0360h | Edge | √ | √ | √ | × | × | IER1B.IEN0 | |
| | | TEI0 | 217 | 0364h | Level | √ | × | × | × | × | IER1B.IEN1 | |
| | SCI1 | ERI1 | 218 | 0368h | Level | √ | × | × | × | × | IER1B.IEN2 | IPR81 |
| | | RXI1 | 219 | 036Ch | Edge | √ | √ | √ | × | × | IER1B.IEN3 | |
| | | TXI1 | 220 | 0370h | Edge | √ | √ | √ | × | × | IER1B.IEN4 | |
| | | TEI1 | 221 | 0374h | Level | √ | × | × | × | × | IER1B.IEN5 | |
| | SCI2 | ERI2 | 222 | 0378h | Level | √ | × | × | × | × | IER1B.IEN6 | IPR82 |
| | | RXI2 | 223 | 037Ch | Edge | √ | √ | √ | × | × | IER1B.IEN7 | |
| | | TXI2 | 224 | 0380h | Edge | √ | √ | √ | × | × | IER1C.IEN0 | |
| | | TEI2 | 225 | 0384h | Level | √ | × | × | × | × | IER1C.IEN1 | |
| | SCI3 | ERI3 | 226 | 0388h | Level | √ | × | × | × | × | IER1C.IEN2 | IPR83 |
| | | RXI3 | 227 | 038Ch | Edge | √ | √ | √ | × | × | IER1C.IEN3 | |
| | | TXI3 | 228 | 0390h | Edge | √ | √ | √ | × | × | IER1C.IEN4 | |
| | | TEI3 | 229 | 0394h | Level | √ | × | × | × | × | IER1C.IEN5 | |
| | SCI4 | ERI4 | 230 | 0398h | Level | √ | × | × | × | × | IER1C.IEN6 | IPR84 |
| | | RXI4 | 231 | 039Ch | Edge | √ | √ | √ | × | × | IER1C.IEN7 | |
| | | TXI4 | 232 | 03A0h | Edge | √ | √ | √ | × | × | IER1D.IEN0 | |
| | | TEI4 | 233 | 03A4h | Level | √ | × | × | × | × | IER1D.IEN1 | |
| | SCI5 | ERI5 | 234 | 03A8h | Level | √ | × | × | × | × | IER1D.IEN2 | IPR85 |
| | | RXI5 | 235 | 03ACh | Edge | √ | √ | √ | × | × | IER1D.IEN3 | |
| | | TXI5 | 236 | 03B0h | Edge | √ | √ | √ | × | × | IER1D.IEN4 | |
| | | TEI5 | 237 | 03B4h | Level | √ | × | × | × | × | IER1D.IEN5 | |
| | SCI6 | ERI6 | 238 | 03B8h | Level | √ | × | × | × | × | IER1D.IEN6 | IPR86 |
| | | RXI6 | 239 | 03BCh | Edge | √ | √ | √ | × | × | IER1D.IEN7 | |
| | | TXI6 | 240 | 03C0h | Edge | √ | √ | √ | × | × | IER1E.IEN0 | |
| | | TEI6 | 241 | 03C4h | Level | √ | × | × | × | × | IER1E.IEN1 | |
| | — | Reserved | 242 | 03C8h | — | × | × | × | × | × | IER1E.IEN2 | — |
| | | Reserved | 243 | 03CCh | — | × | × | × | × | × | IER1E.IEN3 | — |
| ↓ | | Reserved | 244 | 03D0h | — | × | × | × | × | × | IER1E.IEN4 | — |
| Low | | Reserved | 245 | 03D4h | — | × | × | × | × | × | IER1E.IEN5 | — |



Downloaded from Arrow.com.

| Priority | Interrupt Request Source | Name | Vector Number | Vector Address Offset | Form of Detection | Selectable Interrupt Request Destination | | | | | IER | IPR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CPU | DTC | DMAC | Sstb Recovery | Sacs Recovery | | |
| High ↑ | RIIC0 | ICEEI0 | 246 | 03D8h | Level | √ | × | × | × | × | IER1E.IEN6 | PR88 |
| | | ICRXI0 | 247 | 03DCh | Edge | √ | √ | √ | × | × | IER1E.IEN7 | PR89 |
| | | ICTXI0 | 248 | 03E0h | Edge | √ | √ | √ | × | × | IER1F.IEN0 | PR8A |
| | | ICTEI0 | 249 | 03E4h | Level | √ | × | × | × | × | IER1F.IEN1 | PR8B |
| | RIIC1 | ICEEI1 | 250 | 03E8h | Level | √ | × | × | × | × | IER1F.IEN2 | PR8C |
| | | ICRXI1 | 251 | 03ECh | Edge | √ | √ | √ | × | × | IER1F.IEN3 | PR8D |
| | | ICTXI1 | 252 | 03F0h | Edge | √ | √ | √ | × | × | IER1F.IEN4 | PR8E |
| | | ICTEI1 | 253 | 03F4h | Level | √ | × | × | × | × | IER1F.IEN5 | PR8F |
| | — | Reserved | 254 | 03F8h | — | × | × | × | × | × | IER1F.IEN6 | — |
| ↓ Low | | Reserved | 255 | 03FCh | — | × | × | × | × | × | IER1F.IEN7 | — |

[Legend] √: Selectable ×: Not selectable

### 10.3.2 Fast Interrupt Vector Address

The vector address for the interrupt that corresponds to the vector number setting of the fast interrupt is placed in the fast interrupt vector register (FINTV) of the CPU.

### 10.3.3 Non-maskable Interrupt Vector Address

The vector address of the non-maskable interrupt is FFFF FFF8h.



Downloaded from Arrow.com.

## 10.4 Operation

The interrupt control unit determines priority levels of interrupts and non-maskable interrupts and outputs interrupt request signals to the CPU, DTC and/or DMAC.

When the condition for the interrupt source is generated, the corresponding interrupt status flag (IR flag in IRi) is set and interrupt request signal is output to the request destination. For the interrupt request signals to be output to the interrupt request destination, the setting of the IENj bit in IERi must enable the interrupt. If multiple IR flags in IRi are set to 1 at the same time, the interrupt request signals from the highest priority sources for the respective interrupt request destinations are output to the CPU and/or DTC.*

Note: * When multiple interrupt requests that have been set up to activate the DMAC are generated simultaneously, the priority of the sources is determined by the DMAC.

### 10.4.1 Enabling and Disabling Interrupts

The following settings are required to enable an interrupt requests.

- In the case of interrupt requests from peripheral modules, enabling of interrupt output for the corresponding source by the setting of the interrupt enable bit (or bits) of the peripheral module
- In the case of external interrupts, enabling of interrupt output in response to signals on the corresponding IRQn pin by the setting of the IRQEN bit in IRQERn
- Enabling of the interrupt by the corresponding IENj bit in IERi

When an interrupt signal is generated while output for that interrupt source is enabled at the source for interrupt generation, the IR flag in the corresponding IRi register will be set.

If the corresponding IENj bit in IERi allows output of the interrupt request, the interrupt request corresponding to the IR flag in IRi is conveyed to the selected destination. If the IENj bit in IERi has been set to disable the interrupt request, the interrupt request corresponding to the IR flag in IRi retains the indication that the interrupt signal was generated.

That is, the setting of the IENj bit in IERi does not affect the operation of the IR flag in IRi.



Downloaded from Arrow.com.

### 10.4.2 Interrupt Status Flag

The interrupt status flag (IR flag) in IRi detects the corresponding interrupt signal and retains an indication that the request was generated.

There are two ways to detect interrupt sources: level detection and edge detection.

For interrupts from peripheral modules, detection is by either edge or level according to the source.

For the IRQn pins (n = 0 to 15) as interrupt sources, edge or level detection can be selected by the setting of the IRQMD[1:0] bits in IRQCRn.

For the interrupt request and the corresponding detection method of each interrupt request source, see table 10.4, Interrupt Vector Table.

#### 10.4.2.1 Interrupt Status Flag in Edge Detection

Figure 10.2 shows the operation of the IR flag in IRi in the case of edge detection of peripheral module interrupts and external interrupts.

When an interrupt signal is generated, the IR flag in IRi is set to 1 and the interrupt request is conveyed to the selected destination immediately after the point of transition of the interrupt signal. If the destination accepts the interrupt request, the IR flag in IRi is automatically cleared to 0. Therefore, the software being executed does not have to clear the IR flag.

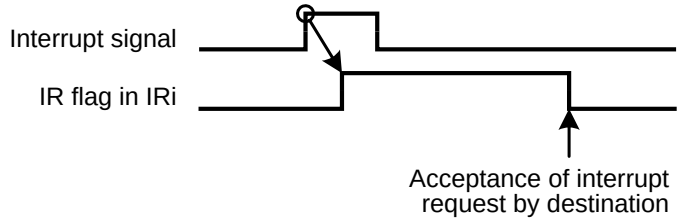



**Figure 10.2 Operation of the IR Flag in IRi in the Case of Edge Detection**

While the IR flag in IRi is set to 1 because an interrupt signal has been generated, re-generation of the interrupt signal is ignored. If the interrupt signal is generated after the IR flag in IRi has been cleared, the IR flag in IRi is re-set. Figure 10.3 shows the timing for re-setting of the IR flag in IRi.

When the communication function (SCI/RIIC) is combined with DTC/DMAC function, an interrupt request is ignored, and a transfer request may be lost. For details, refer to section 10.7.3, Notes on Transferring DMAC/DTC Using Communication Function (SCI, RIIC).

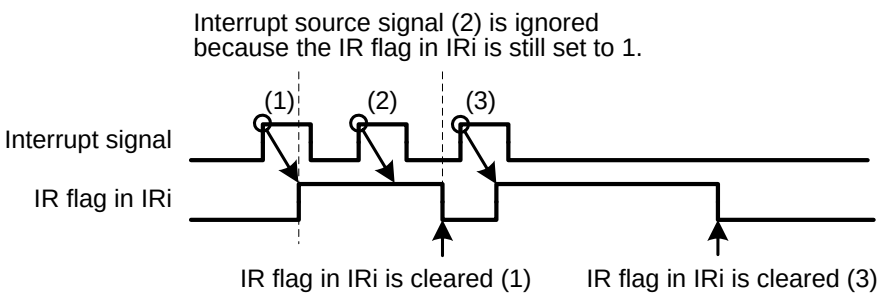



**Figure 10.3 Timing for Re-setting the IR Flag in IRi**



Downloaded from Arrow.com.

Once the IR flag in IRi has been set to 1, even if the interrupt is disabled at its source, that is, if output of the interrupt is disabled by the interrupt enable bit of the corresponding peripheral module or detection of an external interrupt on pin IRQn is disabled by the IRQEN bit in IRQERn, the IR flag in IRi is not affected and retains its value. Figure 10.4 shows operation when the interrupt is disabled at its source.

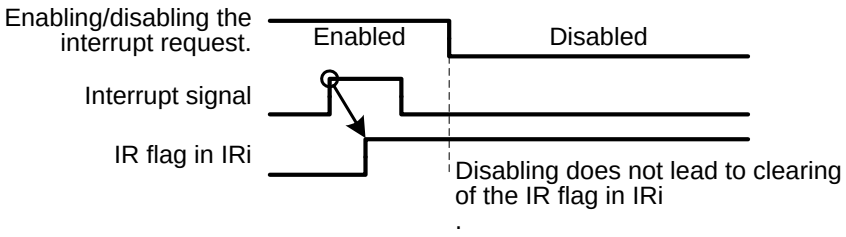



**Figure 10.4 Relation between the IR Flag in IRi and Disabling of the Interrupt Sources**

### 10.4.2.2 Interrupt Status Flag in Level Detection

Operation is individually described for the cases of level detection of peripheral module interrupts and external interrupts.

Figure 10.5 shows how the IR flag in IRi operates in the case of level detection of an interrupt from a peripheral function.

When the IR flag in IRi for an interrupt from a peripheral module has been set to 1, this setting is maintained while generation of the interrupt signal by the source continues. To clear the IR flag in IRi, software should clear the status flag at the source of the interrupt or change the interrupt enable bit at the source of the interrupt to disable interrupt generation. Also, after clearing the source of the interrupt, confirm that the IR flag in IRi has actually been cleared before executing further instructions.

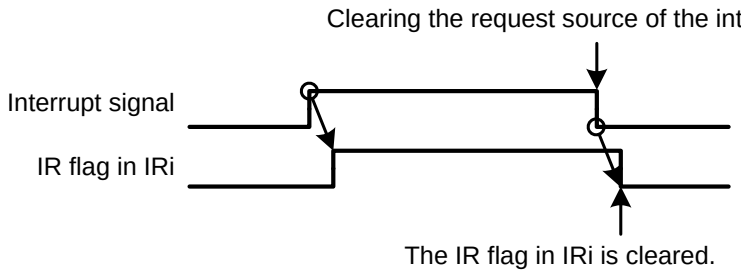



**Figure 10.5 Operation of the IR Flag in IRi in the Case of Level Detection of a Peripheral Module Interrupt**



Downloaded from Arrow.com.

Figure 10.6 shows how the IR flag in IRi operates in the case of level detection of an external interrupt.

If the setting of the IRQMD[1:0] bits in IRQCRn is for detection of the external interrupt as the low level of the signal, the interrupt source should maintain the low level on the IRQn pin until handling of the given interrupt. After that, the interrupt exception handling routine should return the input on the IRQn pin to the high level, which leads to clearing of the IR flag in IRi to 0 after four cycles of the PCLK clock. At least four cycles of PCLK can be secured by, for example, reading the Bj bit in Pm.PORT for the corresponding I/O pin and checking twice for the high level on the IRQn pin.

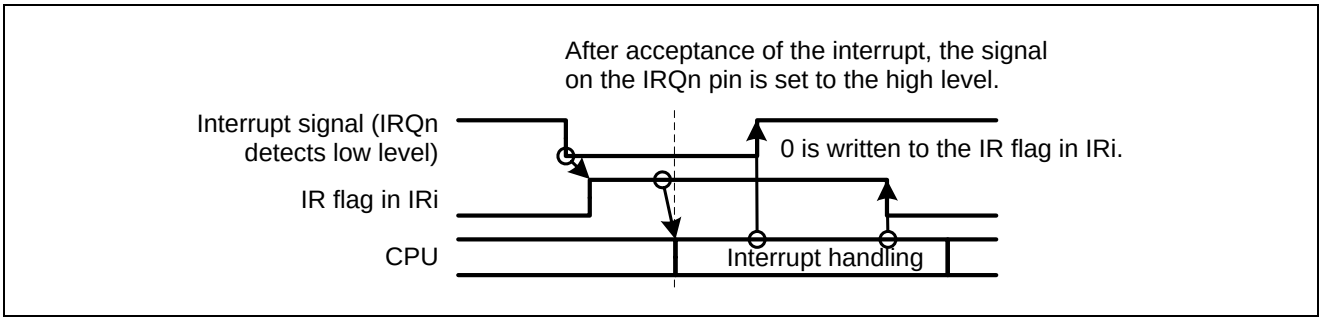



**Figure 10.6 Operation of the IR Flag in IRi in the Case of Level Detection of an External Interrupt**

Once the IR flag in IRi has been set to 1, if the interrupt is disabled at its source, that is, if output of the interrupt is disabled by the interrupt enable bit of the corresponding peripheral module or detection of an external interrupt on pin IRQn is disabled by the IRQEN bit in IRQERn, the IR flag in IRi is cleared. Figure 10.7 shows operation when the interrupt is disabled at its source.

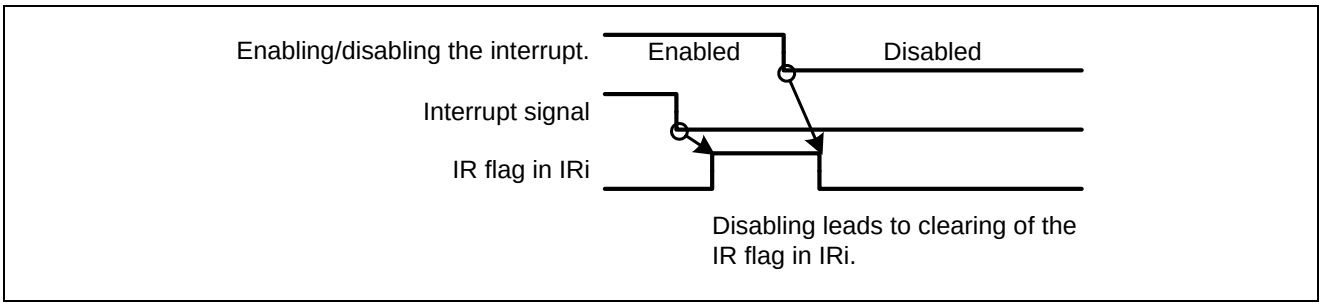



**Figure 10.7 Relation between the IR Flag in IRi and Disabling of the Interrupt Source**



Downloaded from Arrow.com.

### 10.4.3 Selecting Interrupt Request Destinations

Sources that can act as interrupts or activate the DTC and DMAC have ISEL[1:0] bits in the corresponding ISELRi for setting the destination of interrupt requests. The selectable destinations for each of the interrupt sources are fixed and they are listed in table 10.4, Interrupt Vector Table. Do not select a destination which is not indicated in the table.

The following four types of request-destination settings for the ISEL[1:0] bits in ISELRi are available.

1. Interrupt request for the CPU
2. Activate the DTC and convey the interrupt request to the CPU on completion of data transfer.
3. Request to activate the DMAC
4. Activate the DMAC and then convey the interrupt request to the CPU (without waiting for the completion of DMA transfer).

If the source is to activate the DTC or DMAC, use edge detection of the interrupt.

For most interrupts from peripheral modules, only edge detection is available.

If edge detection is to be used for an external interrupt, make the setting for edge detection in the IRQMD[1:0] bits in IRQCRn.

If the setting of the ISEL[1:0] bits in ISELRi is 00b, the interrupt control unit conveys interrupt requests to the CPU.

If the setting of the ISEL[1:0] bits in ISELRi is 01b, the interrupt control unit conveys an activation request to the DTC. Further operation is as follows, according to the setting of the DISEL bit in MRB of the DTC.

- When the DISEL bit in MRB of the DTC is set to 0, the DTC performs the specified number of data transfers. The value of the ISEL[1:0] bits in ISELRi is kept at 01b until the transfer counter reaches 0. On completion of data transfer (i.e. when the transfer counter reaches 0), the ISEL[1:0] bits are automatically updated to 00b. At this point, the interrupt control unit conveys the interrupt request to the CPU.
- When the DISEL bit in MRB of the DTC is set to 1, the ISEL[1:0] bits in ISELRi are automatically updated to 00b on completion of an individual data transfer, regardless of the transfer counter. At this point, the interrupt control unit conveys the interrupt request to the CPU.

In cases where the ISEL[1:0] bits in ISELRi are to be re-set to 01b after having been updated to 00b, modify the value of the bits within the corresponding interrupt exception handler. Furthermore, if re-activation of the DTC is required, ensure that generation of the interrupt signal is possible after setting the ISEL[1:0] bits in ISELRi to 01b.

If the setting of the ISEL[1:0] bits in ISELRi is 10b, the interrupt control unit conveys an activation request to the DMAC.

After DMAC activation, the value of the ISEL[1:0] bits in ISELRi is kept at 10b.

When the selected form of DMA transfer is consecutive-operand transfer or the same interrupt source activates multiple channels, regardless of the type of DMA transfer, enable the generation of further interrupts with the timing described below.

- When the form of DMA transfer is consecutive-operand transfer, make settings to enable the generation of a next interrupt by the source on completion of all DMA transfer on activated channels.[*1]
- When the same interrupt source activates multiple channels, regardless of the type of DMA transfer, make settings to enable the generation of a next interrupt by the source on completion of transfer on all channels.[*2]



Downloaded from Arrow.com.

Notes: 1. Single-operand transfer is transfer for a single operand per activation request. The IR flag in IRi is cleared by activation of transfer for the single operand. In nonstop transfer, one round of DMA transfer proceeds per activation request. The IR flag in IRi is cleared by activation of the DMA transfer. In consecutive-operand transfer, on the other hand, transfer for multiple operands per activation request is possible. The IR flag in IRi is cleared every operand transfer. If the source generates a further interrupt signal before transfer for all operands is complete, once the IR flag in IRi has been set to 1, it is cleared after transfer for individual operands.

2. In cases where the same interrupt source activates multiple channels of the DMAC, all specified channels are activated per activation request regardless of the form of DMA transfer. The IR flag in IRi is cleared every transfer on individual channels. If the source generates a further interrupt signal before transfer for all operands on all channels is complete, once the IR flag in IRi has been set to 1, it is cleared after transfer on individual channels.

If the setting of the ISEL[1:0] bits in ISELRi is 11b, the interrupt control unit activates the DAMC and then automatically updates the ISEL[1:0] bits in ISELRi to 00b without waiting for the completion of data transfer. At this point, the IR flag in IRi is not cleared and the interrupt request is conveyed to the CPU.

In cases where the ISEL[1:0] bits in ISELRi are to be re-set to 01b after having been updated to 00b, modify the value of the bits with the following timing.

- When the selected form of DMA transfer is single-operand transfer or nonstop transfer, modify the value of the bits within the corresponding interrupt exception handler.
- When the selected form of DMA transfer is consecutive-operand transfer, modify the value of the bits on completion of all DMA transfer on activated channels.[*1]
- When the same interrupt source is activating multiple channels, regardless of the type of DMA transfer, modify the value of the bits on completion of transfer on all channels.[*2]

Furthermore, if re-activation of the DMAC is required, ensure that generation of the interrupt signal is possible after setting the ISEL[1:0] bits in ISELRi to 11b.

Notes: 1. Single-operand transfer is transfer for a single operand per activation request. The ISEL[1:0] bits in ISELRi are automatically updated to 00b by activation of transfer for the single operand. In nonstop transfer, one round of DMA transfer proceeds per activation request. The ISEL[1:0] bits in ISELRi are automatically updated to 00b by activation of the DMA transfer. In consecutive-operand transfer, on the other hand, transfer for multiple operands per activation request is possible. The ISEL[1:0] bits in ISELRi are automatically updated to 00 every operand transfer. If the ISEL[1:0] bits in ISELRi are re-set to 11b before transfer for all operands is complete, the ISEL[1:0] bits in ISELRi are updated to 00b after transfer for individual operands.

2. In cases where the same interrupt source activates multiple channels of the DMAC, all specified channels are activated per activation request regardless of the form of DMA transfer. The ISEL[1:0] bits in ISELRi are automatically updated to 00b after transfer on individual channels. If the ISEL[1:0] bits in ISELRi are re-set to 11b before transfer for all operands on all channels is complete, the ISEL[1:0] bits in ISELRi are updated to 00b after transfer on individual channels.



Downloaded from Arrow.com.

### 10.4.4 Determining Priority

The interrupt control unit determines interrupt priority for each interrupt destination.

If multiple interrupt requests are generated for the same destination, the interrupt from the highest priority source is accepted. The method used to determine the priority depends on the interrupt request destination.

#### (1) Determining Priority when Interrupt Request Destination is CPU

For a group for which the ISEL[1:0] bits in ISELRi are set to 00b, the interrupt source with the larger value of the interrupt priority level select bits IPR[2:0] in IPRi takes precedence. If multiple interrupt sources that have the same priority level (same values of the IPR[2:0] bits in IPRi) are generated, the interrupt source with the smallest vector number takes precedence.

When an interrupt source specified as the fast interrupt (described in a later section) is generated, the interrupt request is conveyed to the CPU as an interrupt with the highest priority level (7), regardless of the value of the corresponding IPR[2:0] bits in IPRi and vector number.

#### (2) Determining Priority when the DTC is the Destination of the Interrupt Request

The IPR[2:0] bits in IPRi have no effect on a group for which 01b is set in the ISEL[1:0] bits in ISELRi. An interrupt source with a smaller vector number takes precedence.

#### (3) Determining Priority when the DMAC is the Destination of the Interrupt Request

For a group for which 10b or 11b is set in the ISEL[1:0] bits in ISELRi, the IPR[2:0] bits in IPRi have no effect. The interrupt priority is determined by the setting of the DMAC. See section 12, DMA Controller (DMAC).

### 10.4.5 Fast Interrupt

The fast interrupt is a facility for faster interrupt processing by the CPU. This function has no effect on activation requests for the DTC and DMAC.

By setting the FIEN bit in FIR to 1 and then setting the vector number of the interrupt source to be defined as the fast interrupt in the FVCT[7:0] bits of FIR, interrupts from the source thus defined are output to the CPU for handling as the fast interrupt.

The interrupt source selected for the fast interrupt has the highest priority regardless of the setting of the IPR[2:0] bits in IPRi. When the CPU is engaged in processing for a non-maskable interrupt or a level-7 interrupt, it accepts the fast interrupt after it has completed the current interrupt processing.

For details on the fast interrupt, see section 9, Exceptions.



Downloaded from Arrow.com.

### 10.4.6 External Interrupts

External interrupts are interrupts that have the signals on the IRQn pins (n = 0 to 15) as sources. Figure 10.8 is a block diagram of the circuit for an external interrupt.

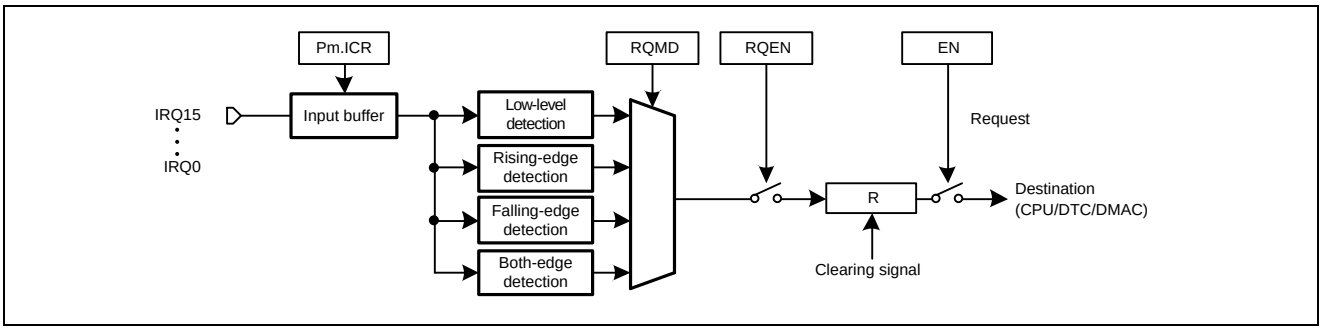



**Figure 10.8 Block Diagram of External Interrupts**

For external interrupts, level detection (low level) or edge detection (falling edge, rising edge, or both edges) is selected by the IRQMD[1:0] bits in IRQCRn.

To set the IRQMD[1:0] bits in IRQCRn, follow the procedure below.

1. Use the IRQEN bit in IRQERn and the IENj bit in IERi to disable detection and requesting of the external interrupt.
2. Set the IRQMD[1:0] bits in IRQCRn.
3. Clear the IR flag in IRi.
4. Use the IRQEN bit in IRQERn and IENj bit in IERi to enable detection and requesting of the external interrupt.

For details on operation in the cases of edge- and level-detected external interrupts, see section 10.4.2, Interrupt Status Flag.

When an external interrupt is in use, the input buffer of the IRQn pin should be enabled by the corresponding Bj bit in Pm.ICR.

When modifying the value of the Bj bit in Pm.ICR, follow the procedure below.

1. Use the IRQEN bit in IRQERn and the IENj bit in IERi to disable detection and requesting of the external interrupt.
2. Modify the setting of the Bj bit in Pm.ICR.
3. Clear the IR flag in IRi after four cycles of the PCLK clock.
4. Use the IRQEN bit in IRQERn and the IENj bit in IERi to enable detection and requesting of the external interrupt.

For the correspondence between the IRQn pins and the Bj bits in Pm.ICR, see section 14, I/O Ports.



Downloaded from Arrow.com.

## 10.5 Non-maskable Interrupt Operation

The interrupt on the non-maskable interrupt (NMI) pin serves as an NMI. Specifically, a rising or falling edge of the signal on the NMI pin issues an NMI request for the CPU. The DTC and DMAC are not selectable as destinations for the NMI. The NMI takes precedence over all other interrupts, including the fast interrupt.

Upon detection of an NMI, the NMI status flag (NMIST bit) in NMISR is set to 1, and an NMI request is issued for the CPU. An NMI request is accepted regardless of the settings of the I bit (interrupt enable bit) and IPL[2:0] bits (processor interrupt priority level) in the PSW of the CPU.

To clear the NMIST flag in NMISR, write 1 to the NMICLR bit in NMICLR. Then, before executing the next instruction, confirm that the NMIST flag in NMISR has been cleared.

To prevent malfunctions in systems that do not require interrupts via the NMI pin, the NMI is disabled by default. If a system is to use the NMI, the procedure below must be included at the beginning of processing by all programs.

**Procedure for Using the NMI**

1. Set the stack pointer (SP).
2. Make the detection setting for the NMI in the NMIMD bit.
3. Clear the NMIST flag in NMISR by writing 1 to the NMICLR bit in NMICLR, and then confirm that the flag is actually cleared.
4. Write 1 to the NMIEN bit in NMIER to enable the NMI.

After the NMIEN bit in NMIER is set to 1, subsequent write access to the bit is ignored. The NMI cannot be disabled.

This feature is essential if the NMI is in use, since it prevents unintentional disabling of the NMI due to a program crashing.

For the flow of non-maskable interrupt processing, see section 9, Exceptions.



Downloaded from Arrow.com.

## 10.6 Returning from Low Power Consumption Modes

The interrupt control unit is capable of returning operation from low power consumption modes in response to interrupts.

Table 10.5 shows the correspondence between low power consumption modes and the interrupt sources that are capable of initiating return from the individual modes.

**Table 10.5 List of Interrupt Sources**

| Low Power Consumption Mode | Capable Interrupt Sources | Interrupt Control Unit Clock |
|---|---|---|
| Sleep | All interrupts including NMI pin interrupts | Runs |
| All-module clock stop | Peripheral function interrupts (WDT, TMR*), external interrupts, NMI pin interrupts | Runs |
| Software standby | External interrupts, NMI pin interrupts | Stopped |

Note: * For details, see section 8, Low Power Consumption.

### 10.6.1 Returning from Sleep Mode and All-Module Clock Stop Mode

The interrupt control unit can return operation from sleep mode or all-module clock stop mode in response to any interrupts.

The following conditions apply.

1. The interrupt destination is the CPU.
2. The given interrupt request is enabled by the IENj bit in IERi.
3. The priority level of the interrupt is higher than that set by the IPL[2:0] bits in the PSW of the CPU.



Downloaded from Arrow.com.

### 10.6.2 Returning from Software Standby Mode

The interrupt control unit can return operation from software standby mode in response to the NMI or an external interrupt on the IRQn pins (n = 0 to 15).

When the NMI or an IRQn (n = 0 to 15) external interrupt is generated, the clock starts oscillating and the clock signal is supplied throughout the LSI, and interrupt processing starts.

For the NMI to act as the trigger for return from software standby mode, the NMIEN bit in NMIER must be set to 1 (enabling the NMI).

For an IRQn (n = 0 to 15) external interrupt to act as the trigger for return from software standby mode, the following conditions apply.

1. The SSIj (j = 0 to 15) bit in SSIER enables the source as a trigger for return.
2. The IRQEN bit in IRQERn enables the detection of external interrupts on the given IRQn pin.
3. The IENj bit in IERi enables the given IRQn pin interrupt request.
4. The ISEL[1:0] bits in ISELRi specify the CPU as the destination of the interrupt request.
5. The IPR[2:0] bits in IPRi specify a priority level higher than that set by the IPL[2:0] bits in the PSW of the CPU.*

Note: * An interrupt source which has been specified as the fast interrupt is the highest-priority interrupt (other than the NMI) regardless of the setting of the IPR[2:0] bits in IPRi. However, if the interrupt source specified as the fast interrupt is to be used as a trigger for return from software standby mode, the setting of the IPR[2:0] bits in IPRi for that interrupt source must satisfy the above relevant condition above.

To cancel software standby mode by an interrupt IRQ0 to IRQ15 for which edge detection is set, 0 should be written to the status flag (the IR bit in IRi of the ICU) of the corresponding interrupt at the beginning of the exception handling routine for the interrupt.

In addition, when an interrupt IRQ0 to IRQ15 is not set as a trigger for return from software standby mode, the input buffer of the corresponding pin becomes invalid in software standby mode and thus the input signal to the LSI is fixed high. In such a case, the interrupt status flag (ICU.IRi.IR) might be set to 1 depending on the state of the pin. Therefore, be sure to execute the WAIT instruction after an interrupt IRQ0 to IRQ15 that is not set as a trigger for cancelation source is masked by the IERi.IENj bit, or its interrupt priority level is made lower by the IPRi.IPR bit. Moreover, clear the interrupt status flag after returning from software standby mode.

For details on the low power consumption states, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

## 10.7 Usage Notes

### 10.7.1 Notes when writing to the Register of the Interrupt Control Unit

The CPU executes the following instruction without waiting for the completion of previous writing to the register of the interrupt control unit. Thus, the following instruction might be executed before the previously written value is stored in the register. To prevent this, be sure to check the completion of writing to the register. In case when multiple registers are successively written to, be sure to check the completion of last writing. For details on checking the completion of writing, see section 5, I/O Registers.

For instance, when masking the interrupt that is not set as a trigger for return from software standby mode, set 0 to the IERi.IEj bit and execute the WAIT instruction before shifting to the software standby state. In this case, if the operation is shifted to the software standby state without waiting for the completion of previous writing, the masked interrupt might be used as a trigger for return. Be sure to check the completion of writing to the IERi.IEj bit before the WAIT instruction is executed.

### 10.7.2 Notes on the WAIT Instruction when the NMI Pin Interrupt is Used

The WAIT instruction should not be executed while the NMISR.NMIST flag is 1. Be sure to check that the NMISR.NMIST flag is 0 before executing the WAIT instruction.

### 10.7.3 Notes on Transferring DMAC/DTC Using Communication Function (SCI, RIIC)

When the DMAC/DTC is activated using an interrupt from the communication function, the DMAC/DTC cannot accept an activation request from the communication function and may not perform DMAC/DTC transfer. In this phenomenon, when the next transfer request is issued by the time that the interrupt status flag (IR flag) is automatically cleared after data transfer (reading reception data or writing transmit data) using the DMAC/DTC, the transfer request is lost.

For example, when the DTC is activated using the SCI reception interrupt with CPU interrupt (DISEL = 1) for every transfer, the IR flag is set to 1 in a first receiving operation and the DTC is activated. After the DTC transfer, the IR flag is retained at 1 until the CPU interrupt is received. In the meantime, if a second receiving operation is completed, a setting of the IR flag that is the transfer request is ignored. Therefore, the DTC cannot be activated, and the data received in second receiving operation cannot be transferred.

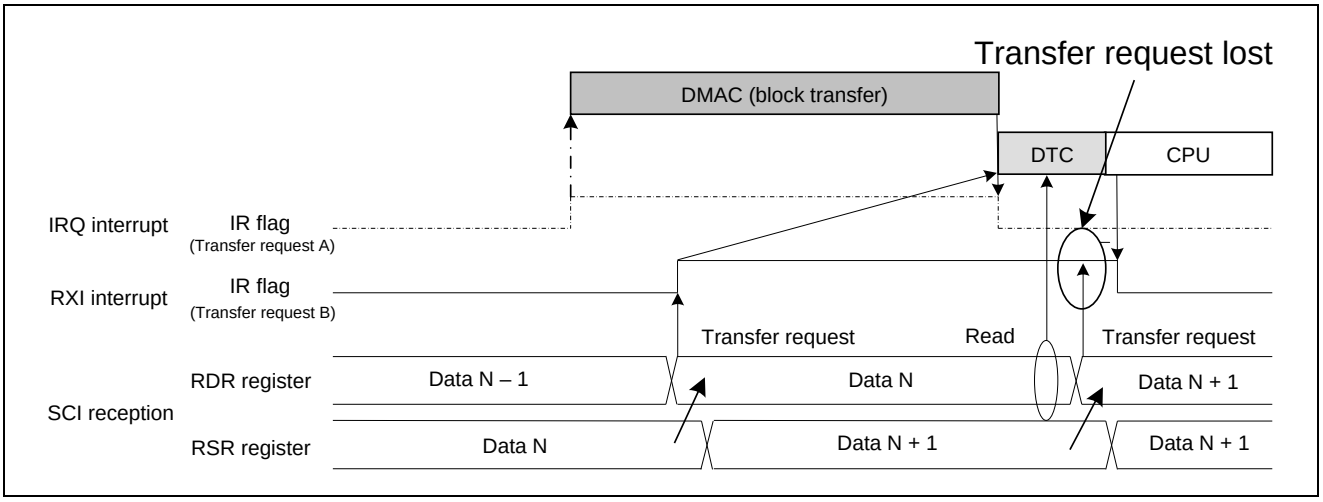



**Figure 10.9 Example of SCI Reception + DTC Transfer (CPU Interrupt (DISEL = 1) for Every Transfer), and a DMAC Block Transfer Using the IRQ Interrupt**



Downloaded from Arrow.com.

**Table 10.6 Setting Conditions of the DMAC/DTC and Occurrence of the Phenomenon**

| Destination of Interrupt Request from Communication Function | Chain Transfer Used or Not Used | No. | Communication Interrupts to CPU Issued or Not Issued | Possibility of Problem Occurrence |
|---|---|---|---|---|
| DMAC | — (Chain transfer not provided) | 1 | No CPU interrupt (ISEL[1:0] = 10b) | Impossible |
| | | 2 | CPU interrupt (ISEL[1:0] = 11b) | Possible |
| DTC | Chain transfer not used | 3 | No CPU interrupt (DISEL = 0) | Impossible |
| | | 4 | CPU interrupt (DISEL = 1) | Possible |
| | Chain transfer used | 5 | No CPU interrupt (DISEL = 0) | Impossible |
| | | 6 | CPU interrupt (DISEL = 1) | Possible |

Note 1. Communication interrupts include transmit data empty and receive data full interrupts from SCI and RIIC.

Note 2. In the final transfer, if the DTC is re-set too late for the transfer request of the next packet to be transmitted/received, the same problem may occur as with the case in DESEL = 1.

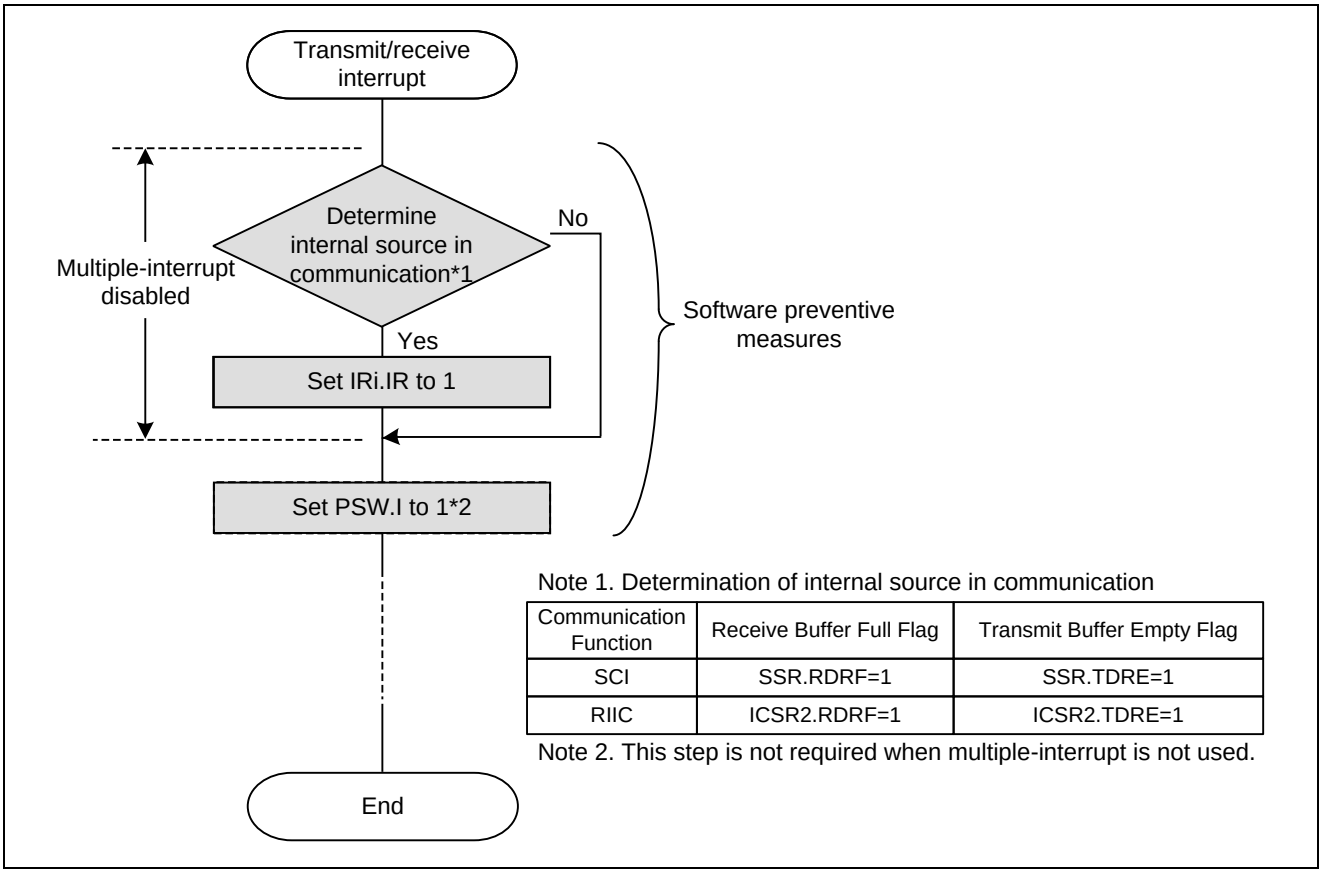
- When the DMAC is used with ISEL[1:0] =11b, use the DTC with DISEL = 1 and implement the following preventive measures.
- When the DTC is used with DISEL = 1, it should be used such that the transfer request is not lost, or implement the software preventive measures of the DTC to prevent the transfer request from being lost.

### (1) Software Preventive Measures

Figure 10.10 shows the flowchart for software preventive measures to be taken for the DTC.



Downloaded from Arrow.com.




Note 1. Determination of internal source in communication

| Communication Function | Receive Buffer Full Flag | Transmit Buffer Empty Flag |
|---|---|---|
| SCI | SSR.RDRF=1 | SSR.TDRE=1 |
| RIIC | ICSR2.RDRF=1 | ICSR2.TDRE=1 |

Note 2. This step is not required when multiple-interrupt is not used.

**Figure 10.10 Flowchart for Measures with DISEL = 1**



Downloaded from Arrow.com.

# 11. Buses

## 11.1 Overview

Table 11.1 lists the bus specifications and figure 11.1 shows the bus configuration.

**Table 11.1 Bus Specifications**

| Bus Type | | Description |
|---|---|---|
| CPU bus | Instruction bus | • Connected to the CPU (for instructions)<br>• Connected to on-chip memory (on-chip RAM, on-chip ROM)<br>• Operates in synchronization with the system clock (ICLK) |
| | Operand bus | • Connected to the CPU (for operands)<br>• Connected to on-chip memory (on-chip RAM, on-chip ROM)<br>• Operates in synchronization with the system clock (ICLK) |
| Internal main bus | Internal main bus 1 | • Connected to the CPU<br>• Operates in synchronization with the system clock (ICLK) |
| | Internal main bus 2 | • Connected to the DMAC and DTC<br>• Connected to on-chip memory (on-chip RAM, on-chip ROM)<br>• Operates in synchronization with the system clock (ICLK) |
| Internal peripheral bus | Internal peripheral bus 1 | • Connected to peripheral modules<br>• Operates in synchronization with the system clock (ICLK) |
| | Internal peripheral bus 2 | • Connected to peripheral modules, on-chip ROM (for programming and erasure), and data-flash memory<br>• Operates in synchronization with the peripheral-module clock (PCLK) |
| External bus | | • Connected to the external devices<br>• Operates in synchronization with the external-bus clock (BCLK) |

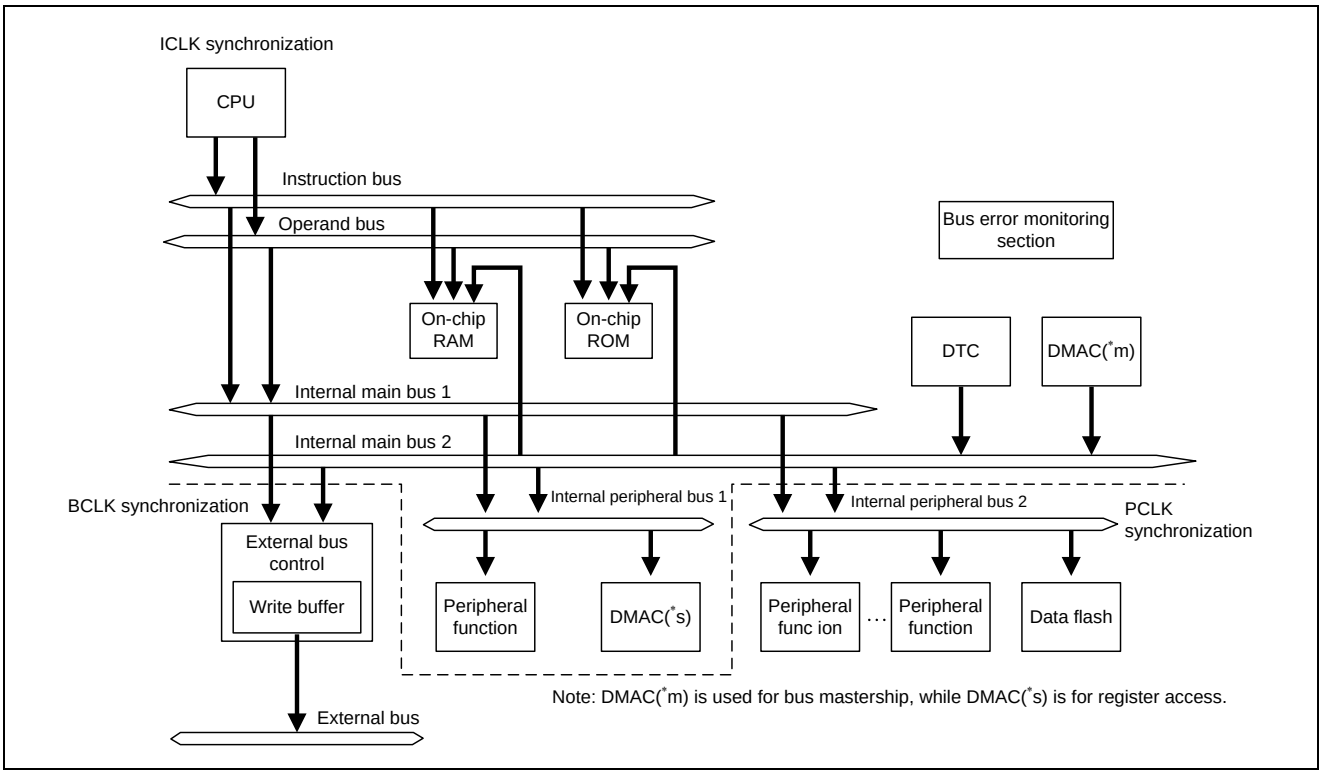



**Figure 11.1 Bus Configuration**



Downloaded from Arrow.com.

## 11.2 Description of Buses

### 11.2.1 CPU Buses

The CPU buses consist of the instruction and operand buses, which are connected to internal main bus 1. As the names suggest, the instruction bus is used to fetch instructions for the CPU, while the operand bus is used for operand access.

Connection of the instruction and operand buses to on-chip RAM and on-chip ROM provides the CPU with direct access to these areas, i.e. access is not via internal main bus 1. However, only reading is possible in direct access to on-chip ROM by the CPU; programming and erasure are handled via an internal peripheral bus.

Units for the arbitration of bus requests for instruction fetching and operand access are on-chip RAM, on-chip ROM, and internal main bus 1. The order of priority is operand access then instruction fetching.

If requests for access via the instruction bus, operand bus, and internal main bus 1 are for different slave modules, the various forms of access can proceed at the same time. For example, parallel access to on-chip ROM and on-chip RAM or to on-chip ROM and external address space is possible.

### 11.2.2 Internal Main Buses

The internal main buses consist of a bus for use by the CPU (internal main bus 1) and a bus for use by the other bus-master modules, i.e. the DMAC and DTC (internal main bus 2). Requests for bus mastership from the DMAC and DTC are arbitrated by internal main bus 2. The order of priority is DMAC and then DTC, as shown in table 11.2.

Units for the arbitration of bus requests from the two internal main buses are slave devices on the external and peripheral buses, and on-chip memory. If the CPU and another bus master are requesting access to different slave modules, the respective bus-access operations can proceed simultaneously.

Internal main bus 2 (for bus masters other than the CPU) has priority over internal main bus 1 (for the CPU). However, when the CPU executes the XCHG instruction, requests for bus access from masters other than the CPU are not accepted until data transfer for the XCHG instruction is completed. Furthermore, requests for bus access from masters other than the DTC are not accepted during reading and writing-back of transfer control information for the DTC.

**Table 11.2 Order of Priority for Bus Masters**

| Priority | Bus Master |
|---|---|
| High ↑ | DMA destination |
| | DMA source |
| | DTC |
| Low | CPU |



Downloaded from Arrow.com.

### 11.2.3 Internal Peripheral Buses

Connection of peripheral modules to the internal peripheral buses is as described in table 11.3.

**Table 11.3 Connection of Peripheral Modules to the Internal Peripheral Buses**

| Type of Bus | Peripheral Modules |
|---|---|
| Internal peripheral bus 1 | • DMAC<br>• Interrupt controller |
| Internal peripheral bus 2 | • Peripheral modules other than those connected to internal peripheral bus 1 |

### 11.2.4 External Bus

Table 11.4 lists the specifications of the external bus.

**Table 11.4 Specifications of the External Bus**

| Item | Description |
|---|---|
| External address space | • An external address space is divided into eight areas (CS0 to CS7) for management.<br>• Chip select signals can be output for each area.<br>• An 8-bit bus space or a 16-bit bus space is selectable for each area.<br>• An endian mode can be specified for each area. |
| Wait control function | • Recovery cycles can be inserted.<br>Read recovery: Up to 15 cycles<br>Write recovery: Up to 15 cycles<br>• Cycle wait function: Wait for up to 31 cycles (page access: up to 7 cycles)<br>• Wait control can be used to set up the following.<br>Timing of assertion and negation for chip-select signals (CS0# to CS7#)<br>The timing of assertion of the read signal (RD#) and write signals (WR0#, WR1#, and WR#)<br>The timing with which data output starts and ends<br>• Write access mode: Single write strobe mode/byte strobe mode |
| Write buffer function | • When write data from the bus master has been written to the write buffer, write access by the bus master is completed. |
| Frequency | • The external bus operates in synchronization with the external bus clock (BCLK). |



Downloaded from Arrow.com.

Table 11.5 shows the input/output pins of the external bus.

**Table 11.5 Pin Configuration of the External Bus**

| Pin Name | I/O | Description |
|---|---|---|
| A23 to A0[*1] | Output | Address output pins |
| BC0#[*1 *2] | Output | A strobe signal; (the BC0# signal being at the low level) during access to an external address space in single write strobe mode indicates that the lower-order byte (D7 to D0) is valid.<br>When an 8-bit bus space is specified, this output pin is always held low regardless of write access mode. |
| BC1#[*2] | Output | A strobe signal; (the BC1# signal being at the low level) during access to an external address space in single write strobe mode indicates that the higher-order byte (D15 to D8) is valid.<br>This pin is not used when the 8-bit bus space is specified. |
| D15 to D0 | I/O | Data input/output pins<br>D15 to D0 pins are enabled when the 16-bit bus space is specified, but D7 to D0 pins are enabled when the 8-bit bus space is specified. |
| CS0# | Output | A strobe signal indicating that area 0 (CS0) is selected |
| CS1# | Output | A strobe signal indicating that area 1 (CS1) is selected |
| CS2# | Output | A strobe signal indicating that area 2 (CS2) is selected |
| CS3# | Output | A strobe signal indicating that area 3 (CS3) is selected |
| CS4# | Output | A strobe signal indicating that area 4 (CS4) is selected |
| CS5# | Output | A strobe signal indicating that area 5 (CS5) is selected |
| CS6# | Output | A strobe signal indicating that area 6 (CS6) is selected |
| CS7# | Output | A strobe signal indicating that area 7 (CS7) is selected |
| RD# | Output | A strobe signal indicating that reading from an external address space is in progress |
| WR0# | Output | A strobe signal; (the WR0# signal being at the low level) during writing to an external address space in byte strobe mode indicates that the higher-order byte (D7 to D0) is valid.<br>This strobe signal also indicates writing to an external address space is in progress in single write strobe mode.<br>When an 8-bit bus space is specified, this output pin is held low during a write access regardless of write access mode. |
| WR1# | Output | A strobe signal; (the WR1# signal being at the low level) during writing to an external address space in byte strobe mode indicates that the higher-order byte (D15 to D8) is valid.<br>The BC1# signal is output in single write strobe mode.<br>This pin is not used when the 8-bit bus space is specified. |
| WR# | Output | A strobe signal to indicate writing to an external address space in single write strobe mode |
| WAIT# | Input | A wait request signal when accessing the external address space (Low: Wait request) |

Note 1. The A0 and BC0# pin functions share the same pin, and either becomes effective according to the area, with the function being A0 in byte-writing mode and BC0# in single write strobe mode. In single write strobe mode, setting 8-bit external bus width is prohibited. For information on other multiplexed pin functions, see section 14, I/O Ports.

Note 2. The BC0# and BC1# signals are readable/writable signals.



Downloaded from Arrow.com.

### 11.2.5 Parallel Operation

Parallel operation is possible when different bus-master modules are requesting access to different slave modules. For example, if the CPU is fetching an instruction from on-chip ROM and an operand from on-chip RAM, the DMAC is able to handle transfer between a peripheral bus and the external bus at the same time.

An example of parallel operations is given in figure 11.2. In this example, the CPU is able to employ the instruction and operand buses for simultaneous access to on-chip ROM and on-chip RAM, respectively. Furthermore, the DMAC simultaneously employs internal main bus 2 for access to a peripheral bus or the external bus during access to on-chip RAM and ROM by the CPU.

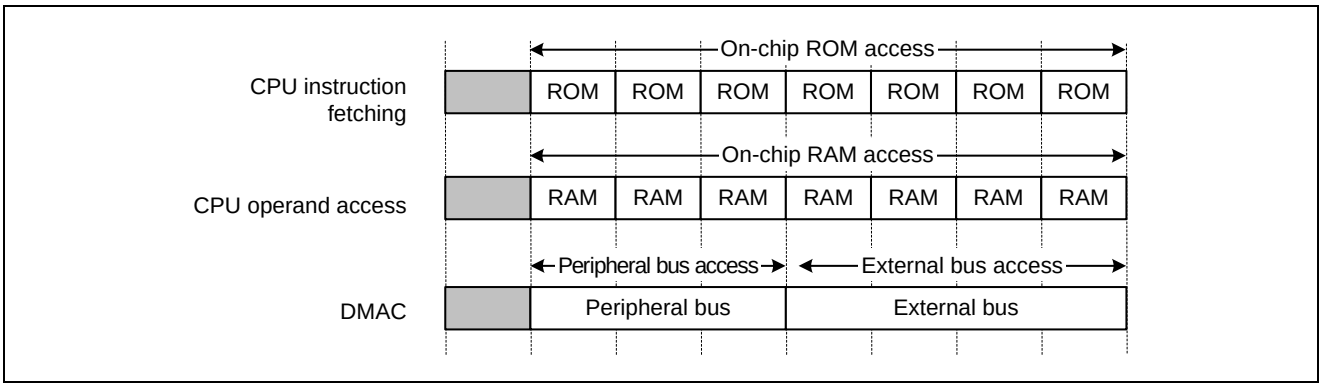



**Figure 11.2 Example of Parallel Operations**



Downloaded from Arrow.com.

## 11.3 Register Descriptions

Table 11.6 lists the registers of the external bus controller.

**Table 11.6 Registers of the External Bus Controller**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| CS0 control register | CS0CNT | 0021h | 0008 3802h | 16 |
| CS0 recovery cycle register | CS0REC | 0000h | 0008 380Ah | 16 |
| CS1 control register | CS1CNT | 0000h | 0008 3812h | 16 |
| CS1 recovery cycle register | CS1REC | 0000h | 0008 381Ah | 16 |
| CS2 control register | CS2CNT | 0000h | 0008 3822h | 16 |
| CS2 recovery cycle register | CS2REC | 0000h | 0008 382Ah | 16 |
| CS3 control register | CS3CNT | 0000h | 0008 3832h | 16 |
| CS3 recovery cycle register | CS3REC | 0000h | 0008 383Ah | 16 |
| CS4 control register | CS4CNT | 0000h | 0008 3842h | 16 |
| CS4 recovery cycle register | CS4REC | 0000h | 0008 384Ah | 16 |
| CS5 control register | CS5CNT | 0000h | 0008 3852h | 16 |
| CS5 recovery cycle register | CS5REC | 0000h | 0008 385Ah | 16 |
| CS6 control register | CS6CNT | 0000h | 0008 3862h | 16 |
| CS6 recovery cycle register | CS6REC | 0000h | 0008 386Ah | 16 |
| CS7 control register | CS7CNT | 0000h | 0008 3872h | 16 |
| CS7 recovery cycle register | CS7REC | 0000h | 0008 387Ah | 16 |
| CS0 mode register | CS0MOD | 0000h | 0008 3002h | 16 |
| CS0 wait control register 1 | CS0WCNT1 | 0707 0707h | 0008 3004h | 32 |
| CS0 wait control register 2 | CS0WCNT2 | 0000 0007h | 0008 3008h | 32 |
| CS1 mode register | CS1MOD | 0000h | 0008 3012h | 16 |
| CS1 wait control register 1 | CS1WCNT1 | 0707 0707h | 0008 3014h | 32 |
| CS1 wait control register 2 | CS1WCNT2 | 0000 0007h | 0008 3018h | 32 |
| CS2 mode register | CS2MOD | 0000h | 0008 3022h | 16 |
| CS2 wait control register 1 | CS2WCNT1 | 0707 0707h | 0008 3024h | 32 |
| CS2 wait control register 2 | CS2WCNT2 | 0000 0007h | 0008 3028h | 32 |
| CS3 mode register | CS3MOD | 0000h | 0008 3032h | 16 |
| CS3 wait control register 1 | CS3WCNT1 | 0707 0707h | 0008 3034h | 32 |
| CS3 wait control register 2 | CS3WCNT2 | 0000 0007h | 0008 3038h | 32 |
| CS4 mode register | CS4MOD | 0000h | 0008 3042h | 16 |
| CS4 wait control register 1 | CS4WCNT1 | 0707 0707h | 0008 3044h | 32 |
| CS4 wait control register 2 | CS4WCNT2 | 0000 0007h | 0008 3048h | 32 |
| CS5 mode register | CS5MOD | 0000h | 0008 3052h | 16 |
| CS5 wait control register 1 | CS5WCNT1 | 0707 0707h | 0008 3054h | 32 |
| CS5 wait control register 2 | CS5WCNT2 | 0000 0007h | 0008 3058h | 32 |



Downloaded from Arrow.com.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| CS6 mode register | CS6MOD | 0000h | 0008 3062h | 16 |
| CS6 wait control register 1 | CS6WCNT1 | 0707 0707h | 0008 3064h | 32 |
| CS6 wait control register 2 | CS6WCNT2 | 0000 0007h | 0008 3068h | 32 |
| CS7 mode register | CS7MOD | 0000h | 0008 3072h | 16 |
| CS7 wait control register 1 | CS7WCNT1 | 0707 0707h | 0008 3074h | 32 |
| CS7 wait control register 2 | CS7WCNT2 | 0000 0007h | 0008 3078h | 32 |
| Bus error source clear register | BERCLR | 00h | 0008 1300h | 8 |
| Bus error monitoring enable register | BEREN | 00h | 0008 1304h | 8 |
| Bus error interrupt enable register | BERIE | 00h | 0008 1306h | 8 |



Downloaded from Arrow.com.

## 11.3.1 CSi Control Register (CSiCNT) (i = 0 to 7)

Address: 0008 3802h (CS0CNT)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | EMODE | – | – | BSIZE[1:0] | | – | – | – | EXENB |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

Addresses: 0008 3812h to 0008 3872h (CS1CNT to CS7CNT)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | EMODE | – | – | BSIZE[1:0] | | – | – | – | EXENB |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | EXENB | Operation Enable | 0: Operation is disabled<br>1: Operation is enabled | R/W*1 |
| b3 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b5, b4 | BSIZE[1:0] | External Bus Width Select | b5 b4<br>0 0: A 16-bit bus space is selected<br>0 1: Setting prohibited<br>1 0: An 8-bit bus space is selected<br>1 1: Setting prohibited | R/W*2 |
| b7, b6 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b8 | EMODE | Endian Mode | 0: Endian of area i (i = 0 to 7) is the same as the endian of operating mode.<br>1: Endian of area i (i = 0 to 7) is not the endian of operating mode. | R/W |
| b15 to b9 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

Notes: 1. After a reset, the value of the EXENB bit in CS0CNT is 1 and the value of the EXENB bit in CSiCNT (i = 1 to 7) is 0.

2. The value of the BSIZE[1:0] bits in CS0CNT after a reset is 10.

CSiCNT is used to set enabling/disabling of the operation, data bus width, and endian for each area in the external address space.

**EXENB Bit (Operation Enable)**

This bit enables or disables operation of the respective CS areas.

After this LSI is reset, operation is only enabled (EXENB = 1) for area 0; operation in other areas is disabled (EXENB = 0).

An attempt at access to an area for which operation has been disabled does not lead to access via the external bus. However, if the illegal address access detection enable bit in the bus error monitoring enable register has been set to enable detection (BEREN.IGAEN = 1), such an attempt will lead to an illegal-access error.

**BSIZE[1:0] Bits (External Bus Width Select)**

These bits specify the data bus width of each area. The data bus width of area 0 after a reset is 8 bits.



Downloaded from Arrow.com.

**EMODE Bit (Endian Mode)**

This bit specifies the endian of each area.

When the endian setting for each area is different from that for the chip, no instruction code can be allocated in the area. If the instruction code is allocated to the external address space, it must be allocated to areas where the endian setting is the same as that for the chip.



Downloaded from Arrow.com.

## 11.3.2 CSi Recovery Cycle Register (CSiREC) (i = 0 to 7)

Addresses: 0008 380Ah to 0008 387Ah

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | WRCV[3:0] | | | | – | – | – | – | RRCV[3:0] | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | RRCV[3:0] | Read Recovery | b3 b2 b1 b0 | R/W |
| | | | 0 0 0 0: No recovery cycle is inserted. | |
| | | | 0 0 0 1: One recovery cycle is inserted. | |
| | | | 0 0 1 0: Two recovery cycles are inserted. | |
| | | | 0 0 1 1: Three recovery cycles are inserted. | |
| | | | 0 1 0 0: Four recovery cycles are inserted. | |
| | | | 0 1 0 1: Five recovery cycles are inserted. | |
| | | | 0 1 1 0: Six recovery cycles are inserted. | |
| | | | 0 1 1 1: Seven recovery cycles are inserted. | |
| | | | 1 0 0 0: Eight recovery cycles are inserted. | |
| | | | 1 0 0 1: Nine recovery cycles are inserted. | |
| | | | 1 0 1 0: Ten recovery cycles are inserted. | |
| | | | 1 0 1 1: Eleven recovery cycles are inserted. | |
| | | | 1 1 0 0: Twelve recovery cycles are inserted. | |
| | | | 1 1 0 1: Thirteen recovery cycles are inserted. | |
| | | | 1 1 1 0: Fourteen recovery cycles are inserted. | |
| | | | 1 1 1 1: Fifteen recovery cycles are inserted. | |
| b7 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b11 to b8 | WRCV[3:0] | Write Recovery | b11 b10 b9 b8 | R/W |
| | | | 0 0 0 0: No recovery cycle is inserted. | |
| | | | 0 0 0 1: One recovery cycle is inserted. | |
| | | | 0 0 1 0: Two recovery cycles are inserted. | |
| | | | 0 0 1 1: Three recovery cycles are inserted. | |
| | | | 0 1 0 0: Four recovery cycles are inserted. | |
| | | | 0 1 0 1: Five recovery cycles are inserted. | |
| | | | 0 1 1 0: Six recovery cycles are inserted. | |
| | | | 0 1 1 1: Seven recovery cycles are inserted. | |
| | | | 1 0 0 0: Eight recovery cycles are inserted. | |
| | | | 1 0 0 1: Nine recovery cycles are inserted. | |
| | | | 1 0 1 0: Ten recovery cycles are inserted. | |
| | | | 1 0 1 1: Eleven recovery cycles are inserted. | |
| | | | 1 1 0 0: Twelve recovery cycles are inserted. | |
| | | | 1 1 0 1: Thirteen recovery cycles are inserted. | |
| | | | 1 1 1 0: Fourteen recovery cycles are inserted. | |
| | | | 1 1 1 1: Fifteen recovery cycles are inserted. | |
| b15 to b12 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

CSiREC is used to set the number of recovery cycles to be inserted after a write access and read access of each area in the external address space.



Downloaded from Arrow.com.

### RRCV[3:0] Bits (Read Recovery)

These bits specify the number of recovery cycles to be inserted after a read access to the external bus.

When a value except 0000b is written to these bits, one to 15 recovery cycles are inserted in the following cases.

- When a write access is made to the external bus after a read access to the external bus
  (Recovery cycles are also inserted when consecutive accesses are made in the same area.)
- When a read access is made to another area after a read access to the external bus
  (No recovery cycle is inserted when consecutive accesses are made in the same area.)

### WRCV[3:0] Bits (Write Recovery)

These bits specify the number of recovery cycles to be inserted after a write access to the external bus.

When a value except 0000b is written to these bits, one to 15 recovery cycles are inserted in the following cases.

- When a read access is made to the external bus after a write access to the external bus
  (Recovery cycles are also inserted when consecutive accesses are made in the same area.)

No recovery cycle is inserted during a write access after a write access.

**Table 11.7 Insertion of Recovery Cycles**

| Access Type | External Address Space | Insertion of Recovery Cycles |
|---|---|---|
| Read access after write access | Same area | Recovery cycles specified by the WRCV[3:0] bits are inserted. |
| | Different area | Recovery cycles specified by the WRCV[3:0] bits are inserted. |
| Write access after write access | Same area | No recovery cycle is inserted. |
| | Different area | No recovery cycle is inserted. |
| Write access after read access | Same area | Recovery cycles specified by the RRCV[3:0] bits are inserted. |
| | Different area | Recovery cycles specified by the RRCV[3:0] bits are inserted. |
| Read access after read access | Same area | No recovery cycle is inserted. |
| | Different area | Recovery cycles specified by the RRCV[3:0] bits are inserted. |



Downloaded from Arrow.com.

### 11.3.3 CSi Mode Register (CSiMOD) (i = 0 to 7)

Addresses: 0008 3002h to 0008 3072h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PRMOD | – | – | – | – | – | PWENB | PRENB | – | – | – | – | EWENB | – | – | WRMOD |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | WRMOD | Write Access Mode Select | 0: Byte strobe mode<br>1: Single write strobe mode | R/W |
| b2, b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b3 | EWENB | External Wait Enable | 0: External wait is disabled<br>1: External wait is enabled | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b8 | PRENB | Page Read Access Enable | 0: Page read access is disabled<br>1: Page read access is enabled | R/W |
| b9 | PWENB | Page Write Access Enable | 0: Page write access is disabled<br>1: Page write access is enabled | R/W |
| b14 to b10 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 | PRMOD | Page Read Access Mode Select | 0: Normal access compatible mode<br>1: External data read continuous assertion mode | R/W |

CSiMOD is used to set access modes of each area in the external address space.

#### WRMOD Bit (Write Access Mode Select)

This bit selects a write access operating mode.

Writing 0 to this bit selects byte strobe mode where data write operation is controlled by the WR0# and WR1# signals corresponding to the respective byte positions.

Writing 1 to this bit selects single write strobe mode where data write operation is controlled by the BCi# (i = 1, 0) signals and WR# signal corresponding to respective byte positions.

Table 11.8 shows whether the control signals are enabled or disabled in each write access mode.

**Table 11.8 Control Signals for Write Access Mode**

| | Data Write Signal | | | Byte Control Signal | |
|---|---|---|---|---|---|
| **Write Access Mode** | **WR0#** | **WR1#** | **WR#** | **BC0#** | **BC1#** |
| Byte strobe mode | Enabled | Enabled | Disabled | Disabled | Disabled |
| Single write strobe mode | Disabled | Disabled | Enabled | Enabled | Enabled |



Downloaded from Arrow.com.

#### EWENB Bit (External Wait Enable)

This bit enables or disables external wait.

Writing 1 to this bit selects external wait and allows control of the number of waits in each cycle with the WAIT# signal. In this state, wait cycles are inserted while the WAIT# signal is at the low level.

Writing 0 to this bit disables the WAIT# signal.

#### PRENB Bit (Page Read Access Enable)

This bit enables or disables page read accesses.

#### PWENB Bit (Page Write Access Enable)

This bit enables or disables page write accesses.

#### PRMOD Bit (Page Read Access Mode Select)

This bit selects a page read access operating mode.

Writing 0 to this bit selects normal access compatible mode where the RD# signal is negated and RD assert wait is inserted each time a piece of data is read. However, when there is no RD assert wait, the RD# signal is negated only when the last data is transferred on the external bus.

Writing 1 to this bit selects external data read continuous assertion mode where RD assert wait is inserted and the RD# signal is continuously asserted during this time period.



Downloaded from Arrow.com.

## 11.3.4 CSi Wait Control Register 1 (CSiWCNT1) (i = 0 to 7)

Addresses: 0008 3004h to 0008 3074h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CSRWAIT[4:0] | | | | | – | – | – | CSWWAIT[4:0] | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | CSPRWAIT[2:0] | | | – | – | – | – | – | CSPWWAIT[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | CSPWWAIT[2:0] | Page Write Cycle Wait Select*1 | b2 b0<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b7 to b3 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b10 to b8 | CSPRWAIT[2:0] | Page Read Cycle Wait Select*2 | b10 b8<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b15 to b11 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b20 to b16 | CSWWAIT[4:0] | Normal Write Cycle Wait Select | b20 b16<br>0 0 0 0 0: No wait is inserted.<br>0 0 0 0 1: Wait with a length of 1 clock cycle is inserted.<br>0 0 0 1 0: Wait with a length of 2 clock cycles is inserted.<br>0 0 0 1 1: Wait with a length of 3 clock cycles is inserted.<br>0 0 1 0 0: Wait with a length of 4 clock cycles is inserted.<br>0 0 1 0 1: Wait with a length of 5 clock cycles is inserted.<br>0 0 1 1 0: Wait with a length of 6 clock cycles is inserted.<br>0 0 1 1 1: Wait with a length of 7 clock cycles is inserted.<br>0 1 0 0 0: Wait with a length of 8 clock cycles is inserted.<br>0 1 0 0 1: Wait with a length of 9 clock cycles is inserted.<br>0 1 0 1 0: Wait with a length of 10 clock cycles is inserted.<br>0 1 0 1 1: Wait with a length of 11 clock cycles is inserted.<br>0 1 1 0 0: Wait with a length of 12 clock cycles is inserted.<br>0 1 1 0 1: Wait with a length of 13 clock cycles is inserted.<br>0 1 1 1 0: Wait with a length of 14 clock cycles is inserted.<br>0 1 1 1 1: Wait with a length of 15 clock cycles is inserted.<br>1 0 0 0 0: Wait with a length of 16 clock cycles is inserted.<br>1 0 0 0 1: Wait with a length of 17 clock cycles is inserted.<br>1 0 0 1 0: Wait with a length of 18 clock cycles is inserted.<br>1 0 0 1 1: Wait with a length of 19 clock cycles is inserted.<br>1 0 1 0 0: Wait with a length of 20 clock cycles is inserted.<br>1 0 1 0 1: Wait with a length of 21 clock cycles is inserted.<br>1 0 1 1 0: Wait with a length of 22 clock cycles is inserted.<br>1 0 1 1 1: Wait with a length of 23 clock cycles is inserted.<br>1 1 0 0 0: Wait with a length of 24 clock cycles is inserted.<br>1 1 0 0 1: Wait with a length of 25 clock cycles is inserted.<br>1 1 0 1 0: Wait with a length of 26 clock cycles is inserted.<br>1 1 0 1 1: Wait with a length of 27 clock cycles is inserted.<br>1 1 1 0 0: Wait with a length of 28 clock cycles is inserted.<br>1 1 1 0 1: Wait with a length of 29 clock cycles is inserted.<br>1 1 1 1 0: Wait with a length of 30 clock cycles is inserted.<br>1 1 1 1 1: Wait with a length of 31 clock cycles is inserted. | R/W |
| b23 to b21 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b28 to b24 | CSRWAIT[4:0] | Normal Read Cycle Wait Select | b28 b24<br>0 0 0 0: No wait is inserted.<br>0 0 0 0: Wait with a length of 1 clock cycle is inserted.<br>0 0 0 1: Wait with a length of 2 clock cycles is inserted.<br>0 0 0 1: Wait with a length of 3 clock cycles is inserted.<br>0 0 1 0: Wait with a length of 4 clock cycles is inserted.<br>0 0 1 0: Wait with a length of 5 clock cycles is inserted.<br>0 0 1 1: Wait with a length of 6 clock cycles is inserted.<br>0 0 1 1: Wait with a length of 7 clock cycles is inserted.<br>0 1 0 0: Wait with a length of 8 clock cycles is inserted.<br>0 1 0 0: Wait with a length of 9 clock cycles is inserted.<br>0 1 0 1: Wait with a length of 10 clock cycles is inserted.<br>0 1 0 1: Wait with a length of 11 clock cycles is inserted.<br>0 1 1 0: Wait with a length of 12 clock cycles is inserted.<br>0 1 1 0: Wait with a length of 13 clock cycles is inserted.<br>0 1 1 1: Wait with a length of 14 clock cycles is inserted.<br>0 1 1 1: Wait with a length of 15 clock cycles is inserted.<br>1 0 0 0: Wait with a length of 16 clock cycles is inserted.<br>1 0 0 0: Wait with a length of 17 clock cycles is inserted.<br>1 0 0 1: Wait with a length of 18 clock cycles is inserted.<br>1 0 0 1: Wait with a length of 19 clock cycles is inserted.<br>1 0 1 0: Wait with a length of 20 clock cycles is inserted.<br>1 0 1 0: Wait with a length of 21 clock cycles is inserted.<br>1 0 1 1: Wait with a length of 22 clock cycles is inserted.<br>1 0 1 1: Wait with a length of 23 clock cycles is inserted.<br>1 1 0 0: Wait with a length of 24 clock cycles is inserted.<br>1 1 0 0: Wait with a length of 25 clock cycles is inserted.<br>1 1 0 1: Wait with a length of 26 clock cycles is inserted.<br>1 1 0 1: Wait with a length of 27 clock cycles is inserted.<br>1 1 1 0: Wait with a length of 28 clock cycles is inserted.<br>1 1 1 0: Wait with a length of 29 clock cycles is inserted.<br>1 1 1 1: Wait with a length of 30 clock cycles is inserted.<br>1 1 1 1: Wait with a length of 31 clock cycles is inserted. | R/W |
| b31 to b29 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

Notes: 1. The CSPWWAIT[2:0] value is valid only when the PWENB bit in CSiMOD is set to 1.
2. The CSPRWAIT[2:0] value is valid only when the PRENB bit in CSiMOD is set to 1.

CSiWCNT1 is used to select the number of wait cycles of each area in the external address space.

### CSPWWAIT[2:0] Bits (Page Write Cycle Wait Select)

These bits specify the number of wait cycles to be inserted into the second and subsequent accesses during a page write cycle.

This setting is enabled when the PWENB bit in CSiMOD is set to 1.

Note: Set these bits so that 1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] or CSON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] is satisfied.



Downloaded from Arrow.com.

### CSPRWAIT[2:0] Bits (Page Read Cycle Wait Select)

These bits specify the number of wait cycles to be inserted into the second and subsequent accesses during a page read cycle.

This setting is enabled when the PRENB bit in CSiMOD is set to 1.

Note: Set these bits so that CSON[2:0] ≤ RDON[2:0] ≤ CSPRWAIT[2:0] is satisfied.

### CSWWAIT[4:0] Bits (Normal Write Cycle Wait Select)

These bits specify the number of wait cycles to be inserted into the first access during a normal write cycle or page write cycle.

Note: Set these bits so that 1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] or CSON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] is satisfied.

### CSRWAIT[4:0] Bits (Normal Read Cycle Wait Select)

These bits specify the number of wait cycles to be inserted into the first access during a normal read cycle or page read cycle.

Note: Set these bits so that CSON[2:0] ≤ RDON[2:0] ≤ CSRWAIT[4:0] is satisfied.

Note: Set each of these bits within a range of the restrictions described in section 11.5.5.1, Limitations at the Time of Normal and Page Access.



Downloaded from Arrow.com.

## 11.3.5 CSi Wait Control Register 2 (CSiWCNT2) (i = 0 to 7)

Addresses: 0008 3008h to 0008 3078h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | CSON[2:0] | | | – | WDON[2:0] | | | – | WRON[2:0] | | | – | RDON[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | WDOFF[2:0] | | | – | CSWOFF[2:0] | | | – | CSROFF[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | CSROFF[2:0] | Read-Access CS Extension Cycle Select | b2 b0<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b3 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b6 to b4 | CSWOFF[2:0] | Write-Access CS Extension Cycle Select | b6 b4<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b7 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b10 to b8 | WDOFF[2:0] | Write Data Output Extension Cycle Select | b10 b8<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b15 to b11 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b18 to b16 | RDON[2:0] | RD Assert Wait Select | b18 b16<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b19 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b22 to b20 | WRON[2:0] | WR Assert Wait Select | b22 b20<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b23 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b26 to b24 | WDON[2:0] | Write Data Output Wait Select | b26 b24<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b27 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b30 to b28 | CSON[2:0] | CS Assert Wait Select | b30 b28<br>0 0: No wait is inserted.<br>0 0: Wait with a length of 1 clock cycle is inserted.<br>0 1: Wait with a length of 2 clock cycles is inserted.<br>0 1: Wait with a length of 3 clock cycles is inserted.<br>1 0: Wait with a length of 4 clock cycles is inserted.<br>1 0: Wait with a length of 5 clock cycles is inserted.<br>1 1: Wait with a length of 6 clock cycles is inserted.<br>1 1: Wait with a length of 7 clock cycles is inserted. | R/W |
| b31 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |

CSiWCNT2 is used to select the number of wait cycles of each area in the external address space.



Downloaded from Arrow.com.

#### CSROFF[2:0] Bits (Read-Access CS Extension Cycle Select)

These bits specify the number of wait cycles to be inserted in a time period from the end of a wait cycle (RD# signal negated) until the CSi# signal (i = 0 to 7) is negated in read access mode.

#### CSWOFF[2:0] Bits (Write-Access CS Extension Cycle Select)

These bits specify the number of wait cycles to be inserted in a time period from the end of a wait cycle (WR0#, WR1#, or WR# signal negated) until the CSi# signal (i = 0 to 7) is negated in write access mode.

Note: Set these bits so that 1 ≤ WDOFF[2:0] ≤ CSWOFF[2:0] is satisfied.

#### WDOFF[2:0] Bits (Write Data Output Extension Cycle Select)

These bits specify the number of wait cycles to be inserted in a time period from the end of a wait cycle (WR0#, WR1#, or WR# signal negated) until the write data output is completed in write access mode.

Note: Set these bits so that 1 ≤ WDOFF[2:0] ≤ CSWOFF[2:0] is satisfied.

#### RDON[2:0] Bits (RD Assert Wait Select)

These bits specify the number of wait cycles to be inserted before the RD# signal is asserted.

Note: Set these bits so that the following conditions are satisfied:
CSON[2:0] ≤ RDON[2:0] ≤ CSRWAIT[4:0] in normal read access
CSON[2:0] ≤ RDON[2:0] ≤ CSPRWAIT[2:0] in page read access

#### WRON[2:0] Bits (WR Assert Wait Select)

These bits specify the number of wait cycles to be inserted before the WR0#, WR1#, or WR# signal is asserted.

Note: Set these bits so that the following conditions are satisfied:
1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] or CSON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] in normal write access
1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] or CSON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] in page write access

#### WDON[2:0] Bits (Write Data Output Wait Select)

These bits specify the number of wait cycles to be inserted before the write data is output.

Note: Set these bits so that the following conditions are satisfied:
1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] in normal write access
1 ≤ WDON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] in page write access



Downloaded from Arrow.com.

### CSON[2:0] Bits (CS Assert Wait Select)

These bits specify the number of wait cycles to be inserted before the CSi# signal (i = 0 to 7) is asserted.

Note: Set these bits so that the following conditions are satisfied:
CSON[2:0] ≤ RDON[2:0] ≤ CSRWAIT[4:0] in normal read access
CSON[2:0] ≤ RDON[2:0] ≤ CSPRWAIT[2:0] in page read access
CSON[2:0] ≤ WRON[2:0] ≤ CSWWAIT[4:0] in normal write access
CSON[2:0] ≤ WRON[2:0] ≤ CSPWWAIT[2:0] in page write access

Note: Set each of these bits within a range of the restrictions described in section 11.5.5.1, Limitations at the Time of Normal and Page Access.



Downloaded from Arrow.com.

## 11.3.6 Bus Error Source Clear Register (BERCLR)

Address: 0008 1300h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | STSCLR |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| 0 | STSCLR | Bus Error Source Clear | 0: No effect<br>1: Clears the bus-error source signals | R/(W)* |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

Note: * Only the writing of 1 is effective; i.e. writing 0 has no effect.

**STSCLR Bit (Bus Error Source Clear)**

Writing 1 to this bit clears the internal bus-error source signals. Clear the bus-error source signals being retained by the bus from within the handling routine for bus-error interrupts.

Writing 0 to this bit has no effect. The value 0 is always read from this bit.

## 11.3.7 Bus Error Monitoring Enable Register (BEREN)

Address: 0008 1304h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | TOEN | IGAEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| 0 | IGAEN | Illegal Address Access Detection Enable | 0: Illegal address access detection is disabled.<br>1: Illegal address access detection is enabled. | R/W |
| 1 | TOEN | Time-out Detection Enable*1*2 | 0: Time-out detection is disabled.<br>1: Time-out detection is enabled. | R/W |
| b7 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

Notes: 1. When detection is disabled (the TOEN bit is cleared to 0), bus access can cause the bus to freeze.
2. Do not set the TOEN bit to the value that disables detection while time-out errors are being detected.

**IGAEN Bit (Illegal Address Access Detection Enable)**

This bit enables or disables detection of access to illegal addresses.

**TOEN Bit (Time-out Detection Enable)**

This bit enables or disables detection of time-out for bus operations.



Downloaded from Arrow.com.

### 11.3.8 Bus Error Interrupt Enable Register (BERIE)

Address: 0008 1306h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | CPEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | CPEN | CPU Bus Error Notification Control | 0: Bus-error interrupts are not conveyed to the CPU.<br>1: Bus-error interrupts are conveyed to the CPU. | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

**CPEN Bit (CPU Bus Error Notification Control)**

This bit controls whether the interrupt controller is (1) or is not (0) notified when bus errors are detected.



Downloaded from Arrow.com.

## 11.4 Endian and Data Alignment

Eight, 16, and 32 bits are the units of access by the CPU and other internal bus-master modules. The external bus controller has a data-alignment facility, and this can control whether access is to bits D15 to D8 or to bits D7 to D0, the bus specification (8-bit or 16-bit bus space), the unit of access, and the endian for areas of the external address space.

### 11.4.1 16-Bit Bus Space

When a 16-bit width is selected for a bus space by the BSIZE[1:0] bits in CSiCNT, the address bus (A23 to A1) output signals for access to 16-bit units are enabled, but the A0 signal is always 0.

When byte strobe mode (the WRMOD bit = 0 in CSiMOD) is selected, the WRi# (i = 0, 1) pins are enabled, but the BCi# (i = 0, 1) pins are not used.

When single write strobe mode (the WRMOD bit = 1 in CSiMOD) is selected, the WR# pin is enabled and always outputs the low level during write access, regardless of the unit of access. The valid byte position is indicated by the BCi# (i = 0, 1) pins. The WRi# (i = 0, 1) pins are not used.

Page access can occur in access to data in 32-bit units. The situations in which page access occurs are indicated by the letter "(p)" in figures 11.3 and 11.4.

The valid positions of data external to the chip and of control signals differ according to whether the endian is big or little.

| Data Size | Access Address | Number of Access | Bus Cycle | Unit of Data | Address | | WR1#/BC1# D15–D8 | WR0#/BC0# D7–D0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | RD# | |
| | | | | | | | Data Bus | |
| 8 bits | 4n | One | First | 8 bits | 4n | | | 7–0 |
| | 4n+1 | One | First | 8 bits | 4n | | 7–0 | |
| | 4n+2 | One | First | 8 bits | 4n+2 | | | 7–0 |
| | 4n+3 | One | First | 8 bits | 4n+2 | | 7–0 | |
| 16 bits | 4n | One | First | 16 bits | 4n | | 15–8 | 7–0 |
| | 4n+1 | Two | First | 8 bits | 4n | | 7–0 | |
| | | | Second | 8 bits | 4n+2 | | | 15–8 |
| | 4n+2 | One | First | 16 bits | 4n+2 | | 15–8 | 7–0 |
| | 4n+3 | Two | First | 8 bits | 4n+2 | | 7–0 | |
| | | | Second | 8 bits | 4n+4 | | | 15–8 |
| 32 bits | 4n | Two | First | 16 bits | 4n | | 15–8 | 7–0 |
| | | | Second | 16 bits | 4n+2 | (p) | 31–24 | 23–16 |
| | 4n+1 | Three | First | 8 bits | 4n | | 7–0 | |
| | | | Second | 16 bits | 4n+2 | | 23–16 | 15–8 |
| | | | Third | 8 bits | 4n+4 | | | 31–24 |
| | 4n+2 | Two | First | 16 bits | 4n+2 | | 15–8 | 7–0 |
| | | | Second | 16 bits | 4n+4 | | 31–24 | 23–16 |
| | 4n+3 | Three | First | 8 bits | 4n+2 | | 7–0 | |
| | | | Second | 16 bits | 4n+4 | | 23–16 | 15–8 |
| | | | Third | 8 bits | 4n+6 | | | 31–24 |

[Legend]
(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSiMOD)

**Figure 11.3 Data Alignment (Little Endian) in 16-Bit Bus Space**



Downloaded from Arrow.com.

| Data Size | Access Address | Number of Access | Bus Cycle | Unit of Data | Address | | WR1#/BC1# / WR0#/BC0# / RD# / Data Bus (D15 D8 D7 D0) |
|---|---|---|---|---|---|---|---|
| 8 bits | 4n | One | First | 8 bits | 4n | | 7 0 |
| | 4n+1 | One | First | 8 bits | 4n | | 7 0 |
| | 4n+2 | One | First | 8 bits | 4n+2 | | 7 0 |
| | 4n+3 | One | First | 8 bits | 4n+2 | | 7 0 |
| 16 bits | 4n | One | First | 16 bits | 4n | | 15 8 7 0 |
| | 4n+1 | Two | First | 8 bits | 4n | | 15 8 |
| | | | Second | 8 bits | 4n+2 | | 7 0 |
| | 4n+2 | One | First | 16 bits | 4n+2 | | 15 8 7 0 |
| | 4n+3 | Two | First | 8 bits | 4n+2 | | 15 8 |
| | | | Second | 8 bits | 4n+4 | | 7 0 |
| 32 bits | 4n | Two | First | 16 bits | 4n | | 31 24 23 16 |
| | | | Second | 16 bits | 4n+2 | (p) | 15 8 7 0 |
| | 4n+1 | Three | First | 8 bits | 4n | | 31 24 |
| | | | Second | 16 bits | 4n+2 | | 23 16 15 8 |
| | | | Third | 8 bits | 4n+4 | | 7 0 |
| | 4n+2 | Two | First | 16 bits | 4n+2 | | 31 24 23 16 |
| | | | Second | 16 bits | 4n+4 | | 15 8 7 0 |
| | 4n+3 | Three | First | 8 bits | 4n+2 | | 31 24 |
| | | | Second | 16 bits | 4n+4 | | 23 16 15 8 |
| | | | Third | 8 bits | 4n+6 | | 7 0 |

[Legend]
(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSiMOD)

**Figure 11.4 Data Alignment (Big Endian) in 16-Bit Bus Space**

## 11.4.2 8-Bit Bus Space

When an 8-bit bus space is selected by the BSIZE[1:0] bits in CSiCNT, the address bus (A23 to A0) output signals for access to byte units are enabled.

When byte strobe mode is selected, the WR0# pin is valid, and when single write strobe mode is selected, the WR# pin is valid. In write access, the low level is always output on the WR0# and WR# pins. The BC0#, WR1#, and BC1# pins are not used.

Page access can occur in access to data in 16- or 32-bit units. The situations in which page access occurs are indicated by the letter "(p)" in figures 11.5 and 11.6.

The valid positions of data external to the chip and of the control signals are not affected by the endian.



Downloaded from Arrow.com.

| Data Size | Access Address | Number of Access | Bus Cycle | Unit of Data | Address | | WR1#/BC1# WR0#/BC0# RD# Data Bus D15 D8 D7 D0 |
|---|---|---|---|---|---|---|---|
| 8 bits | 4n | One | First | 8 bits | 4n | | 7 0 |
| | 4n+1 | One | First | 8 bits | 4n+1 | | 7 0 |
| | 4n+2 | One | First | 8 bits | 4n+2 | | 7 0 |
| | 4n+3 | One | First | 8 bits | 4n+3 | | 7 0 |
| 16 bits | 4n | Two | First | 8 bits | 4n | | 7 0 |
| | | | Second | 8 bits | 4n+1 | (p) | 15 8 |
| | 4n+1 | Two | First | 8 bits | 4n+1 | | 7 0 |
| | | | Second | 8 bits | 4n+2 | (p) | 15 8 |
| | 4n+2 | Two | First | 8 bits | 4n+2 | | 7 0 |
| | | | Second | 8 bits | 4n+3 | (p) | 15 8 |
| | 4n+3 | Two | First | 8 bits | 4n+3 | | 7 0 |
| | | | Second | 8 bits | 4n+4 | | 15 8 |
| 32 bits | 4n | Four | First | 8 bits | 4n | | 7 0 |
| | | | Second | 8 bits | 4n+1 | (p) | 15 8 |
| | | | Third | 8 bits | 4n+2 | (p) | 23 16 |
| | | | Fourth | 8 bits | 4n+3 | (p) | 31 24 |
| | 4n+1 | Four | First | 8 bits | 4n+1 | | 7 0 |
| | | | Second | 8 bits | 4n+2 | (p) | 15 8 |
| | | | Third | 8 bits | 4n+3 | (p) | 23 16 |
| | | | Fourth | 8 bits | 4n+4 | | 31 24 |
| | 4n+2 | Four | First | 8 bits | 4n+2 | | 7 0 |
| | | | Second | 8 bits | 4n+3 | (p) | 15 8 |
| | | | Third | 8 bits | 4n+4 | | 23 16 |
| | | | Fourth | 8 bits | 4n+5 | (p) | 31 24 |
| | 4n+3 | Four | First | 8 bits | 4n+3 | | 7 0 |
| | | | Second | 8 bits | 4n+4 | | 15 8 |
| | | | Third | 8 bits | 4n+5 | (p) | 23 16 |
| | | | Fourth | 8 bits | 4n+6 | (p) | 31 24 |

[Legend]
(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSiMOD)

**Figure 11.5 Data Alignment (Little Endian) in 8-Bit Bus Space**



Downloaded from Arrow.com.

| Data Size | Access Address | Number of Access | Bus Cycle | Unit of Data | Address | | WR1#/BC1# / WR0#/BC0# / RD# Data Bus D15–D8 | D7–D0 |
|---|---|---|---|---|---|---|---|---|
| 8 bits | 4n | One | First | 8 bits | 4n | | | 7 0 |
| | 4n+1 | One | First | 8 bits | 4n+1 | | | 7 0 |
| | 4n+2 | One | First | 8 bits | 4n+2 | | | 7 0 |
| | 4n+3 | One | First | 8 bits | 4n+3 | | | 7 0 |
| 16 bits | 4n | Two | First | 8 bits | 4n | | | 15 8 |
| | | | Second | 8 bits | 4n+1 | (p) | | 7 0 |
| | 4n+1 | Two | First | 8 bits | 4n+1 | | | 15 8 |
| | | | Second | 8 bits | 4n+2 | (p) | | 7 0 |
| | 4n+2 | Two | First | 8 bits | 4n+2 | | | 15 8 |
| | | | Second | 8 bits | 4n+3 | (p) | | 7 0 |
| | 4n+3 | Two | First | 8 bits | 4n+3 | | | 15 8 |
| | | | Second | 8 bits | 4n+4 | | | 7 0 |
| 32 bits | 4n | Four | First | 8 bits | 4n | | | 31 24 |
| | | | Second | 8 bits | 4n+1 | (p) | | 23 16 |
| | | | Third | 8 bits | 4n+2 | (p) | | 15 8 |
| | | | Fourth | 8 bits | 4n+3 | (p) | | 7 0 |
| | 4n+1 | Four | First | 8 bits | 4n+1 | | | 31 24 |
| | | | Second | 8 bits | 4n+2 | (p) | | 23 16 |
| | | | Third | 8 bits | 4n+3 | (p) | | 15 8 |
| | | | Fourth | 8 bits | 4n+4 | | | 7 0 |
| | 4n+2 | Four | First | 8 bits | 4n+2 | | | 31 24 |
| | | | Second | 8 bits | 4n+3 | (p) | | 23 16 |
| | | | Third | 8 bits | 4n+4 | | | 15 8 |
| | | | Fourth | 8 bits | 4n+5 | (p) | | 7 0 |
| | 4n+3 | Four | First | 8 bits | 4n+3 | | | 31 24 |
| | | | Second | 8 bits | 4n+4 | | | 23 16 |
| | | | Third | 8 bits | 4n+5 | (p) | | 15 8 |
| | | | Fourth | 8 bits | 4n+6 | (p) | | 7 0 |

[Legend]
(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSiMOD)

**Figure 11.6 Data Alignment (Big Endian) in 8-Bit Bus Space**



Downloaded from Arrow.com.

## 11.5 Operation

### 11.5.1 Timing of External Bus Access

The various periods in the timing charts are described below.

#### (1) Tw1 to Twn (clock cycles of waiting for a normal read cycle or normal write cycle)

The period Tw1 to Twn is made up of the number of clock cycles between the start of access via the external bus and the cycle where waiting is completed (described below). The number of cycles is selectable within the range from zero to 31. Within this period, the timing of CSi#, RD#, WRi#, and WR# assertion (placing the signals at the low level) is determined by the respective wait settings. Specifically, the periods of waiting are controlled by the CS assertion wait (CSON), RD assertion wait (RDON), WR assertion wait (WRON), and write-data output wait (WDON) bits of CSi wait control register 2 (CSiWCNT2). The number of clock cycles for each of these periods of waiting is selectable as a value from zero to seven counted from the start of bus access. Selectable numbers of cycles are also within the overall number of clock cycles of waiting for reading or writing.

#### (2) Tend (clock cycle where the strobe signal is valid)

Tend is the next clock cycle after completion of the period of waiting for a normal cycle of reading or writing or for a cycle of page reading or page writing. If the number of clock cycles in the period of waiting for a normal cycle of reading or writing or for a cycle of page reading or page writing is zero, the clock cycle where bus access starts is the clock cycle where the strobe signal is valid. The RD#, WRi#, and WR# signals are negated in the next clock cycle after the cycle where the strobe signal is valid. In the case of read access, the clock cycle where the strobe signal is valid becomes the clock cycle where the data to be read are sampled.

If an external wait is enabled, the wait signal is sampled at the time of the cycle where the strobe signal is valid. The bus cycle is extended if the wait signal is at the low level. The bus cycle is completed in the next clock cycle if the wait signal is at the high level. Tend indicates the cycle where sampling of the wait signal starts.

After the first cycle where the strobe signal is valid during page access, second and subsequent page access operations (point 6. below) start in the next cycle except in cases of write access where a setting (other than zero) for write-data output extension clock cycles (point 5. below) has been made. If the setting for the RD or WR assertion wait is a value other than zero, the RD#, WRi#, and WR# signals are negated in the next clock cycle. If the setting is zero, assertion continues. Furthermore, the CSi# signal continues to be asserted rather than being negated.



Downloaded from Arrow.com.

### (3) Tn1 to Tnm (clock cycles of CS extension)

In the case of normal access, Tn1 to Tnm represent the clock cycles of the period following the cycle where the strobe signal is valid (Tend) up to negation of the CSi# signal. For read or write access, the timing of negation is controlled by the CS extension cycles setting for reading (CSROFF) or the CS extension cycles setting for writing (CSROFF) in CSi wait control register 2 (CSiWCNT2).

The number of cycles is counted from the cycle following the cycle where the strobe signal is valid.

In the case of page access, Tn1 to Tnm represent the clock cycles of the period for the cycle following the last cycle where the strobe signal is valid up to negation of the CSi# signal.

For write access, the setting for write data output extension cycles (WDOFF) controls extension of the period where the address and output data are valid.

### (4) Th (address hold time)

In the clock cycle that follows the end of the cycles of CS extension, the value for the address is retained from the previous access. However, when two or more rounds of external bus access are required in response to a request for transfer from a bus master, only the value from the final bus access of the divided-up operation is retained. In bus access other than the final bus access, the address value is updated to that for the next access operation at the time of negation of the CS# signal. In the case of normal access, one clock cycle is inserted as the period of negation of the CSi# signal. In the case of page access, a period for negation of the CS# signal is not inserted (see figures 11.10 and 11.11).

### (5) Tdw1 to Tdwn (write-data output extension clock cycles)

For write access, if the setting for write-data output extension wait is a value other than zero, clock cycles of write-data output extension are inserted from the cycle that follows the cycle where the strobe signal is valid (Tend).

In the case of normal access, this is inserted within the period of clock cycles of CS extension (point 3. above).

In the case of page access, this is inserted within the period of the cycle where the strobe signal is valid and subsequent page access or within the period of clock cycles of CS extension (point 3. above). Valid address and data output are extended over this period, and the WRi# and WR# signals are negated.

### (6) Tpw1 to Tpwn (page-read cycle or page-write cycle wait)

For the second and subsequent cycles of the bus clock during page access, the values for a page-read cycle wait or page-write cycle wait are used instead of the settings for a normal read or write cycle wait. The setting for the wait until WR assertion becomes effective is in a similar way as for the first round of access. How the setting for RD assertion controls operation differs with the setting for page-read access mode (CSiMOD.PRMOD) as described below.

CSiMOD.PRMOD = 0: A wait until RD assertion is inserted in a similar way as for the first round of access, and the RD# signal is negated.

CSiMOD.PRMOD = 1: Although a wait until RD assertion is inserted in a similar way as for normal-access compatibility mode, the RD# signal continues to be asserted over this period.



Downloaded from Arrow.com.

(7) Tr1 to Trn (clock cycles of recovery)

Clock cycles of recovery can be inserted from the point where a bus cycle is completed. The number of clock cycles is controlled by the setting of the read recovery (RRCV) or write recovery (WRCV) bits in the CSi recovery cycle setting register (CSiREC). Both numbers of clock cycles of recovery are counted from the next cycle after the address hold time and have values in the range from zero to 15 (clock cycles). For details on the clock cycles of recovery, see section 11.5.3, Insertion of Recovery Cycles.

### 11.5.1.1 Normal Access

When the PRENB and PWENB bits in CSiMOD are set to 0 to disable page-reading and page-writing access, respectively, all bus access will take the form of normal read and write operations.

Even when the PRENB and PWENB bits in CSiMOD are set to 1 to enable page-reading and page-writing access, respectively, bus access other than page access will take the form of normal read and write operations.

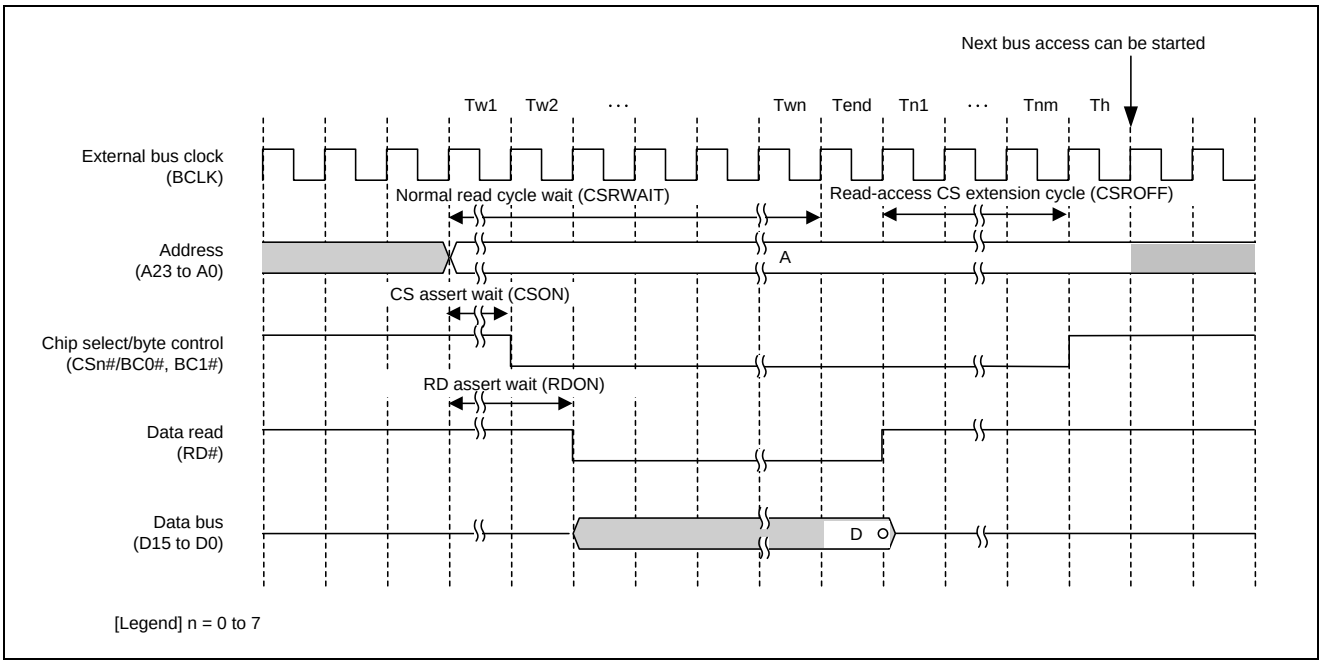



**Figure 11.7 Bus Timing (Normal-Read Operation)**



Downloaded from Arrow.com.

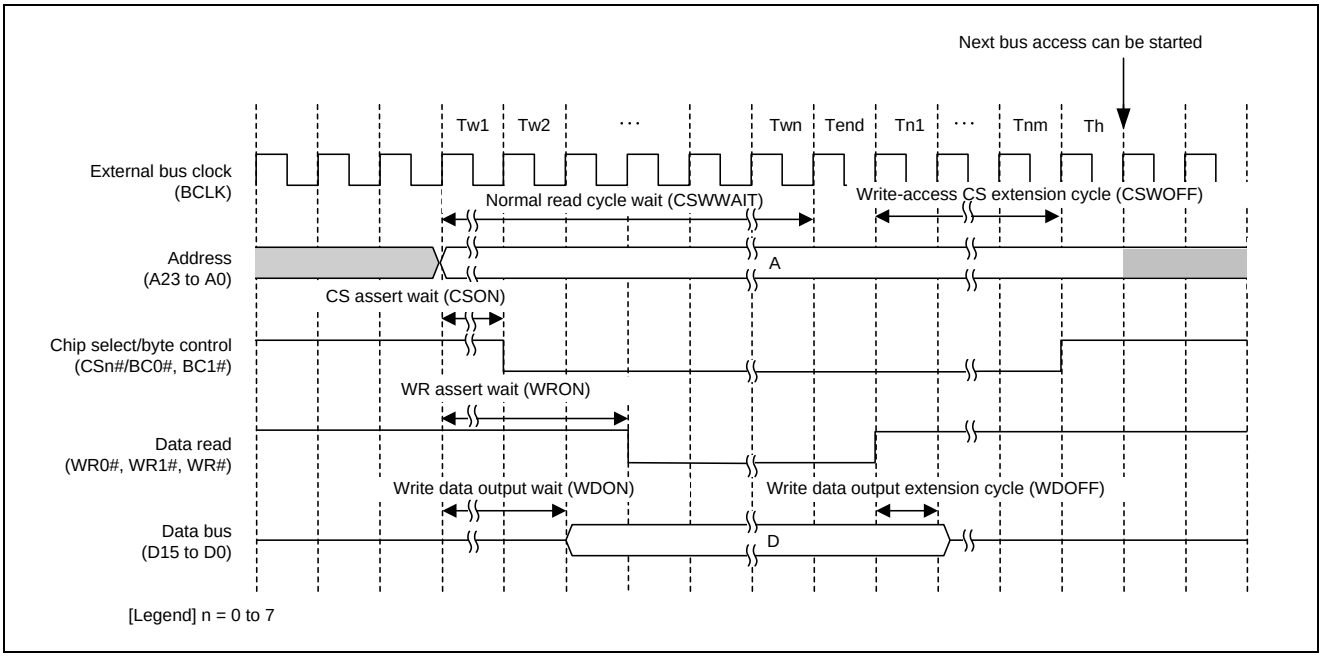



**Figure 11.8 Bus Timing (Normal-Write Operation)**

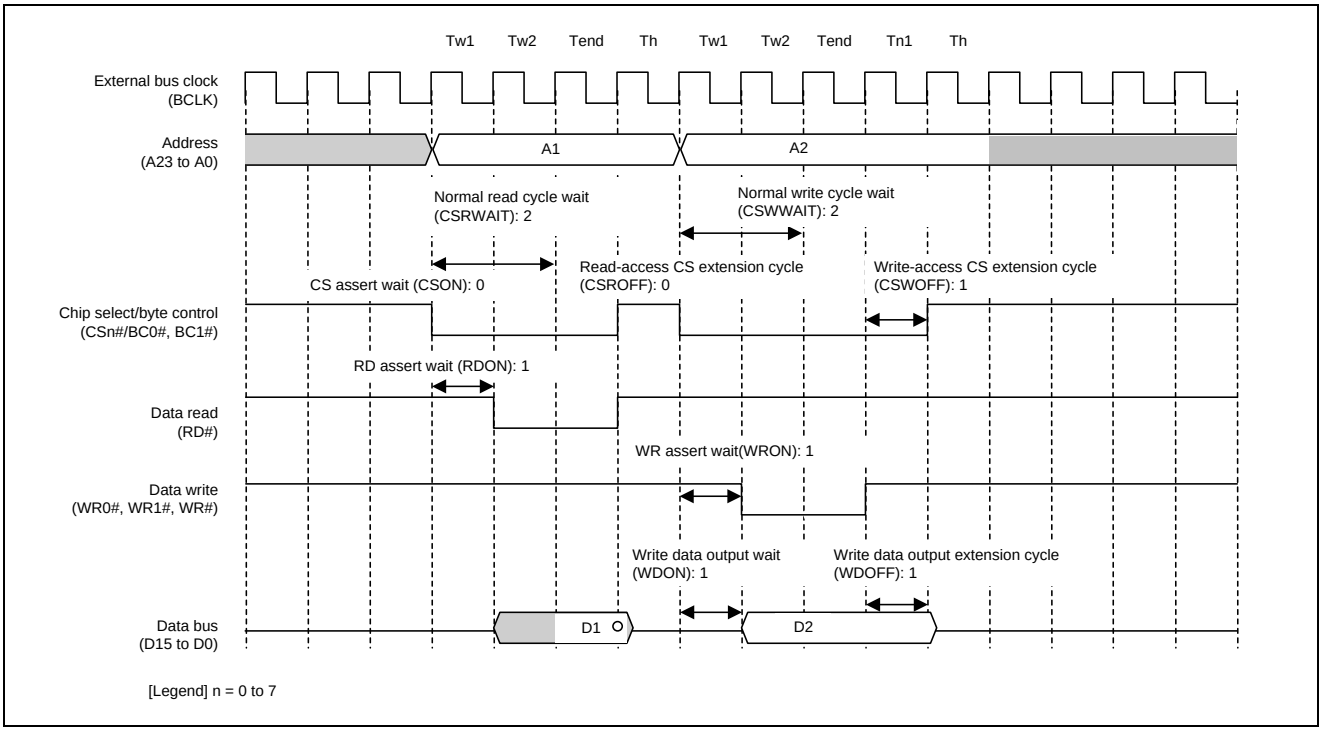



**Figure 11.9 Example of Normal Access Operation (Read/Write)**



Downloaded from Arrow.com.

When two or more rounds of external bus access are required in response to a single request for transfer from a bus master, normal access operations (points 1. to 4. above) are repeated. Figures 11.10 and 11.11 show examples of operations when two rounds of bus access are generated in response to a single request for transfer. The values of the wait control registers are example settings. In practice, the register settings will correspond to the specifications of connected devices.

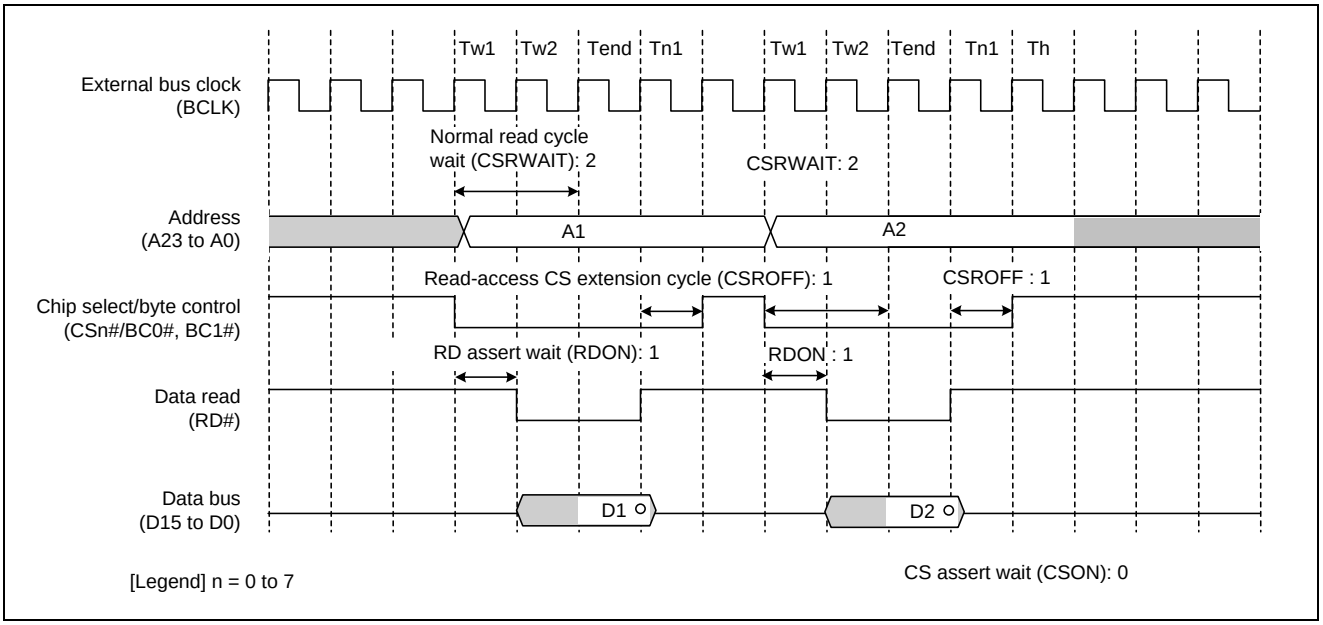



**Figure 11.10 Example of Normal-Read Operation (when Two Rounds of Bus Access are Generated in Response to a Single Request for Transfer)**

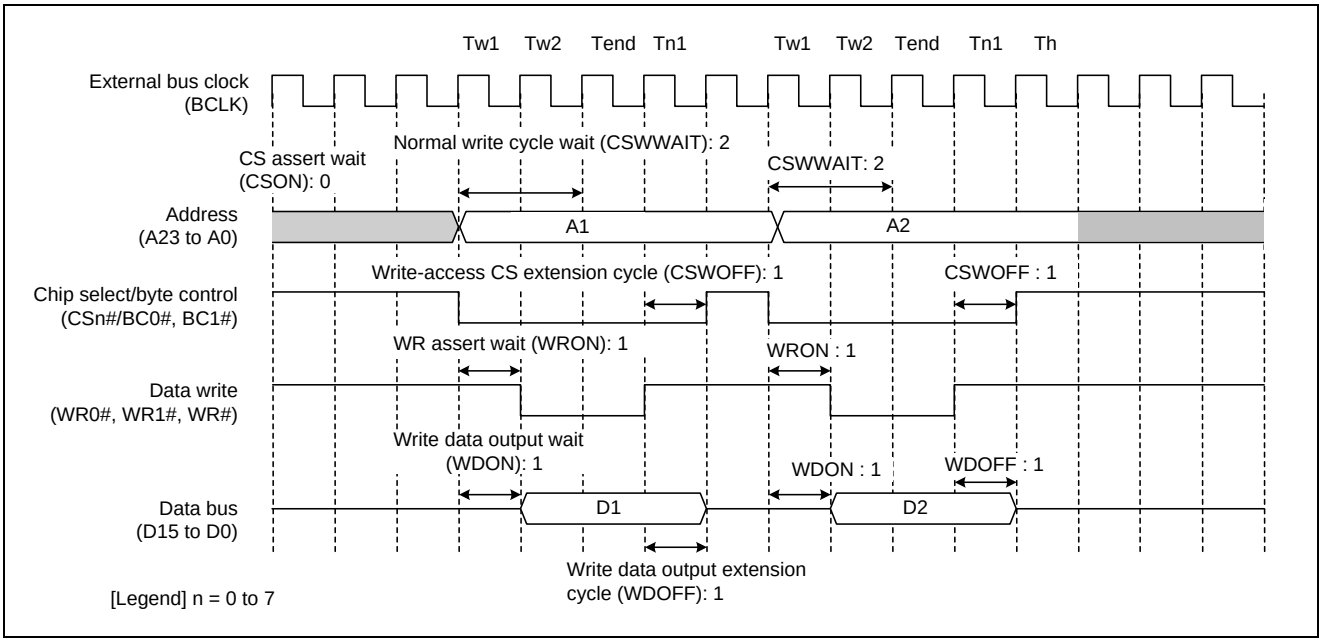



**Figure 11.11 Example of Normal-Write Operation (when Two Rounds of Bus Access are Generated in Response to a Single Request for Transfer)**



Downloaded from Arrow.com.

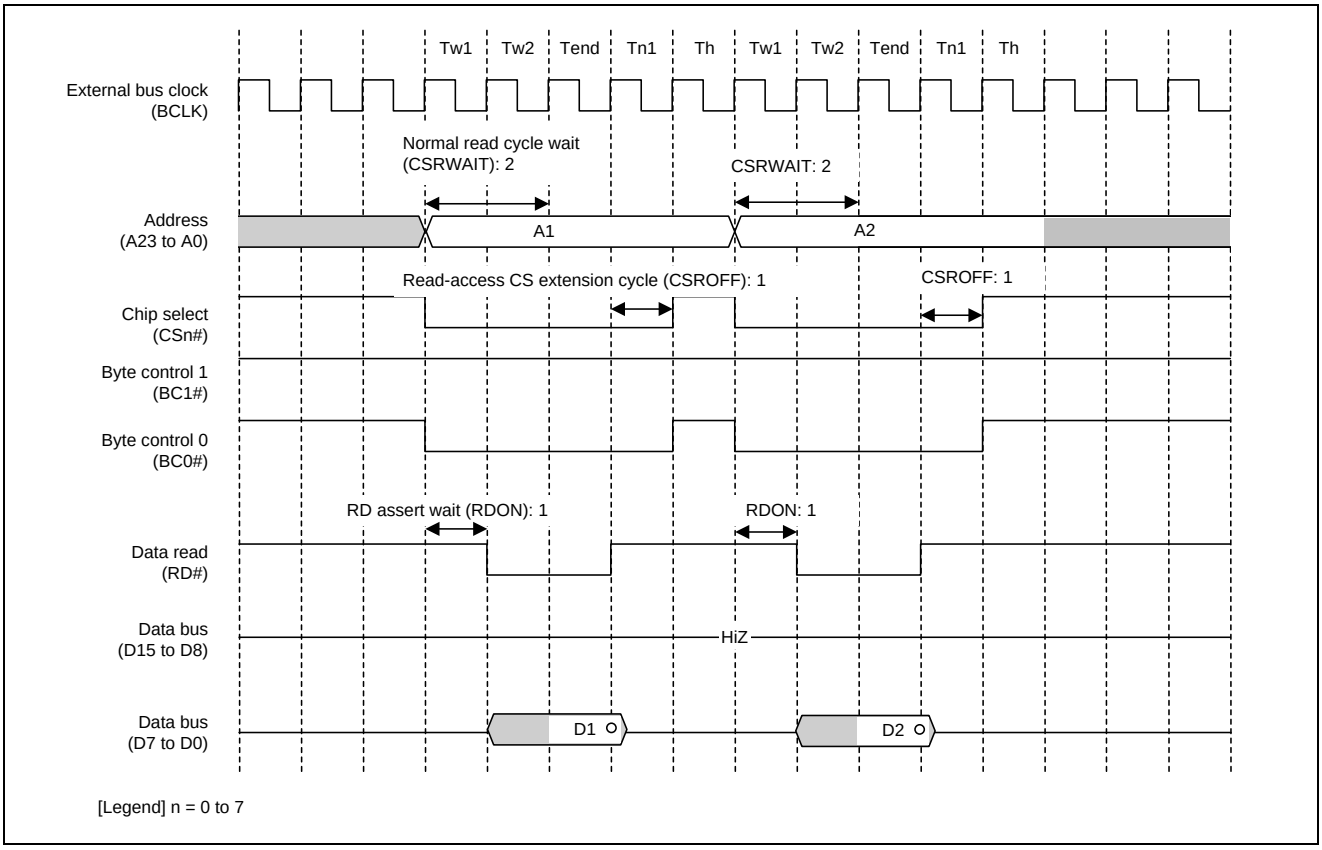



**Figure 11.12 Example of Normal-Read Operation (when 16-Bit Bus Space is Accessed in 8 Bits)**

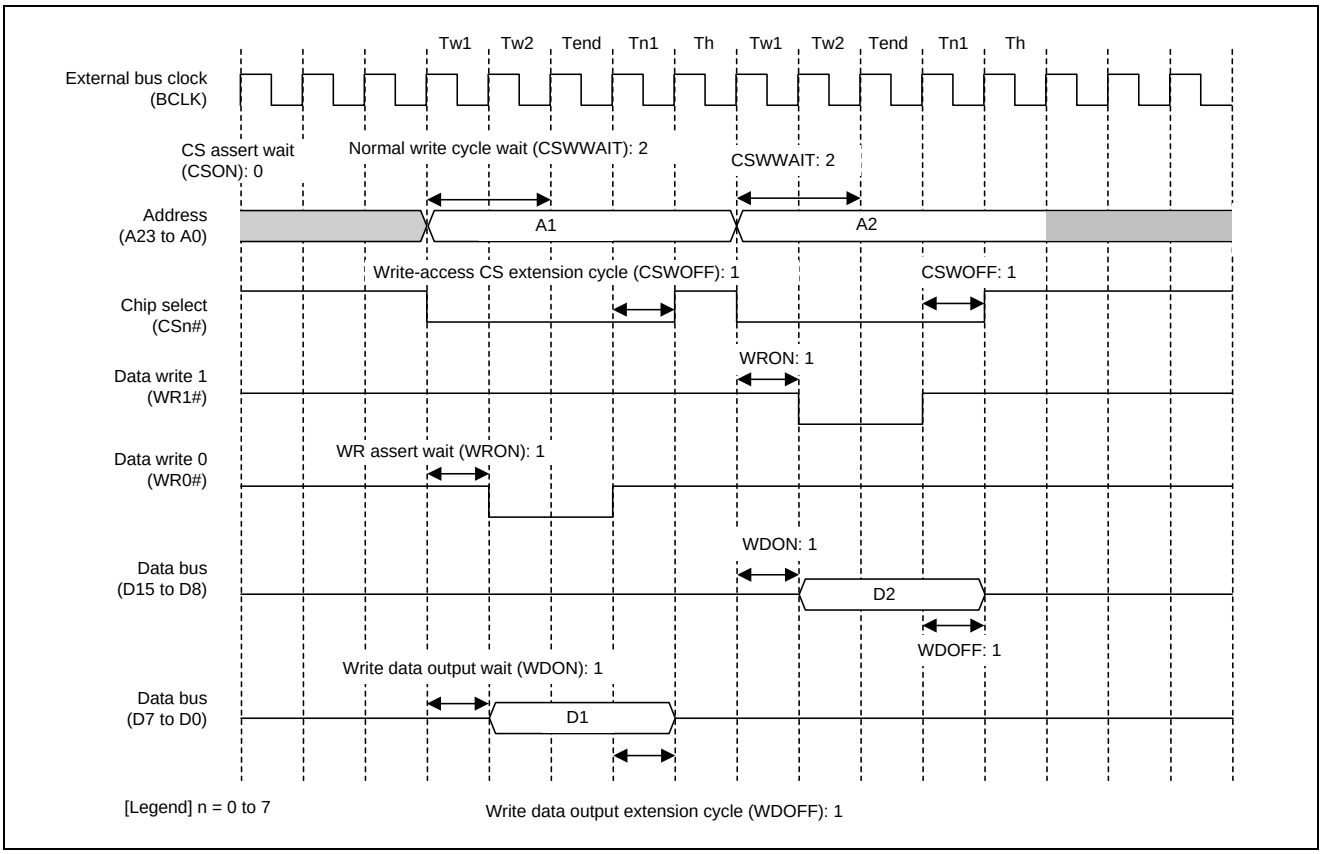



**Figure 11.13 Example of Normal-Write Operation (when 16-Bit Bus Space is Accessed in 8 Bits, in Byte Strobe Mode)**



Downloaded from Arrow.com.

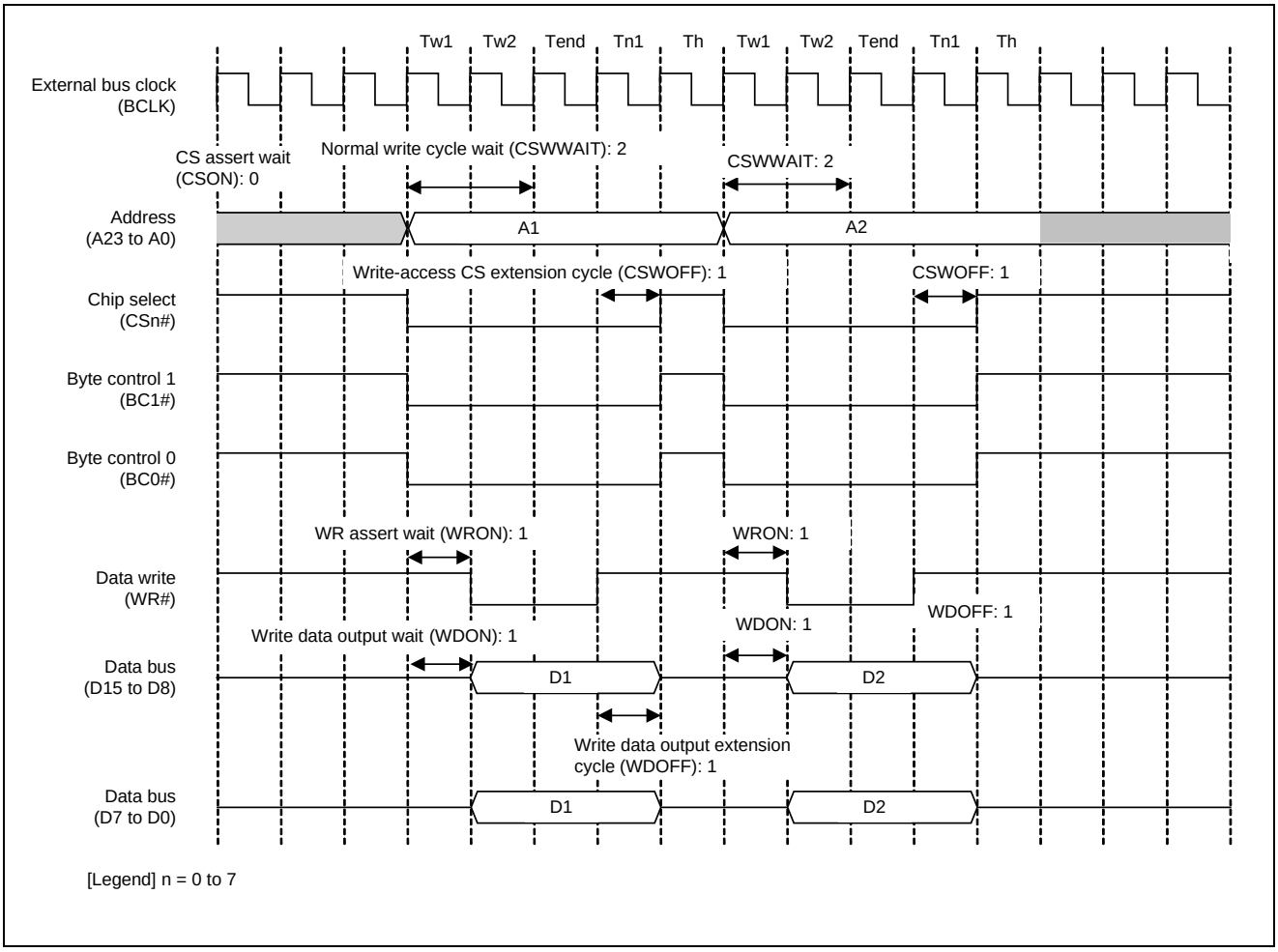



**Figure 11.14 Example of Normal-Write Operation (when 16-Bit Bus Space is Accessed in 16 Bits, in Single Write Strobe Mode)**



Downloaded from Arrow.com.

### 11.5.1.2 Page Access

When the PRENB and PWENB bits in CSiMOD are set to 1 to enable page-reading and page-writing access, respectively, the bus access for page access operations becomes page reading and writing. Page access is made when two or more rounds of external bus access are required for a single transfer request from the bus master. See figures 11.15 to 11.18 for the conditions under which page access occurs.

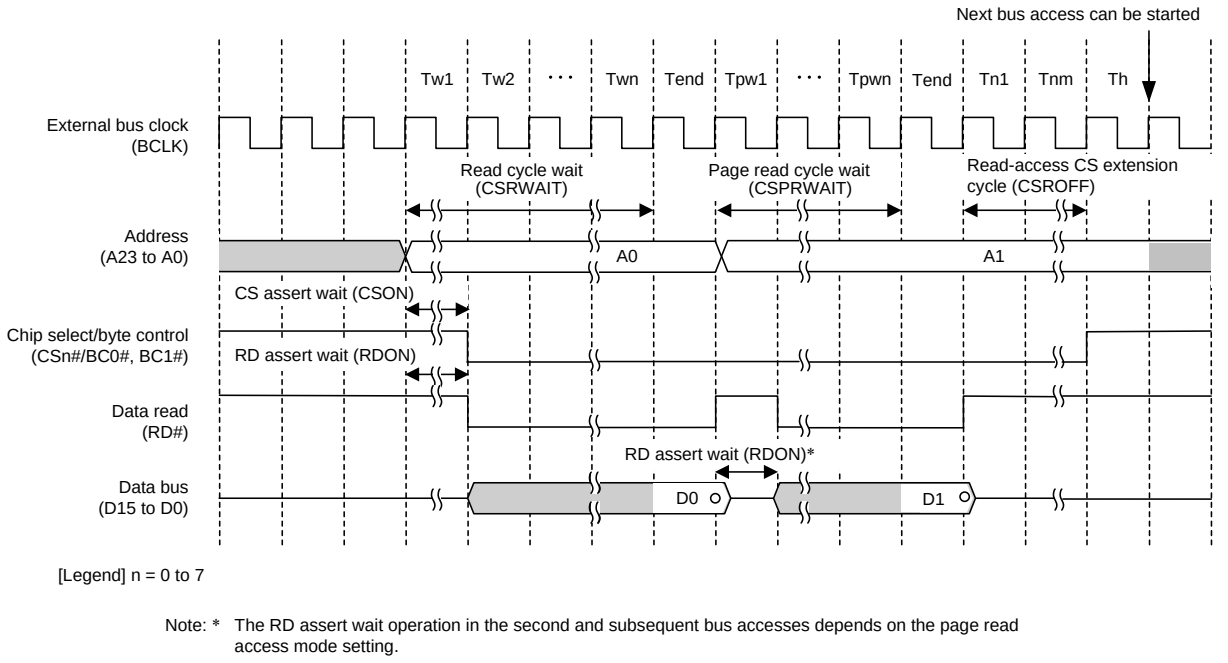



**Figure 11.15 Page-Read Access Timing**



Downloaded from Arrow.com.

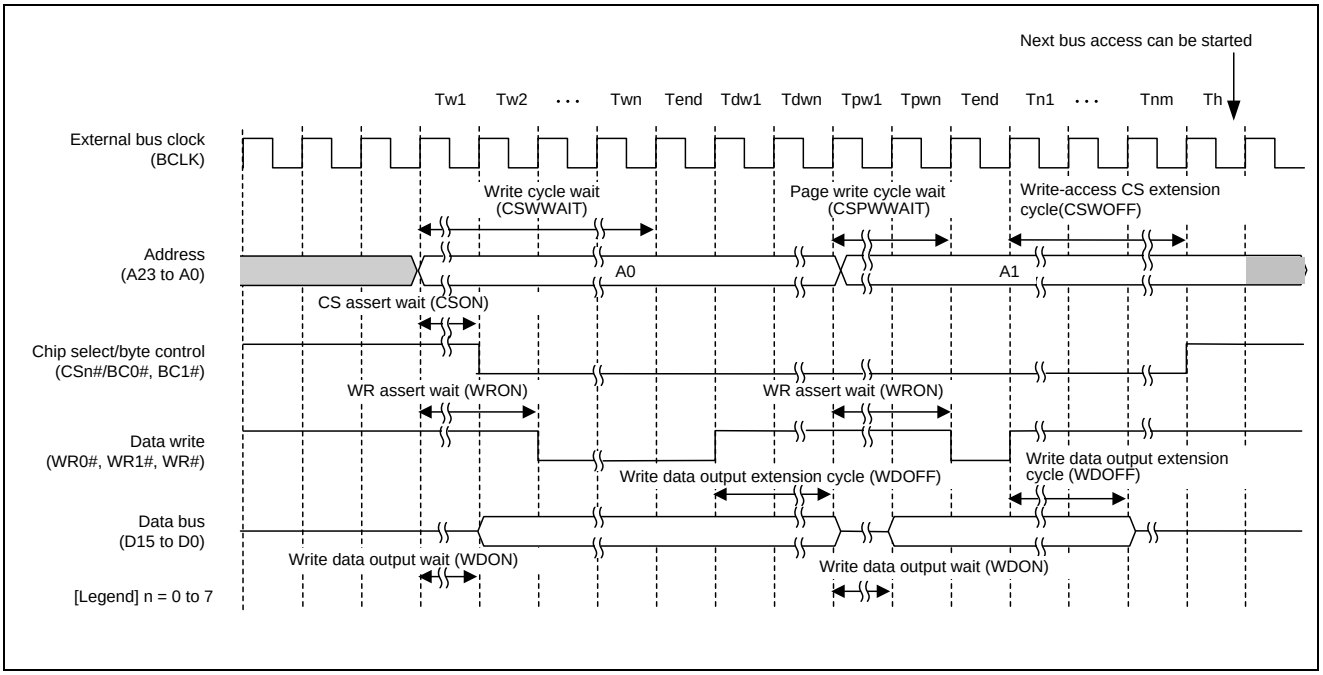



**Figure 11.16 Page-Write Access Timing**

Figures 11.17 and 11.18 depict examples of operations for access to a 16-bit bus space in 32 bits. The values of the wait control registers are example settings. In practice, the register settings will correspond to the specifications of connected devices.

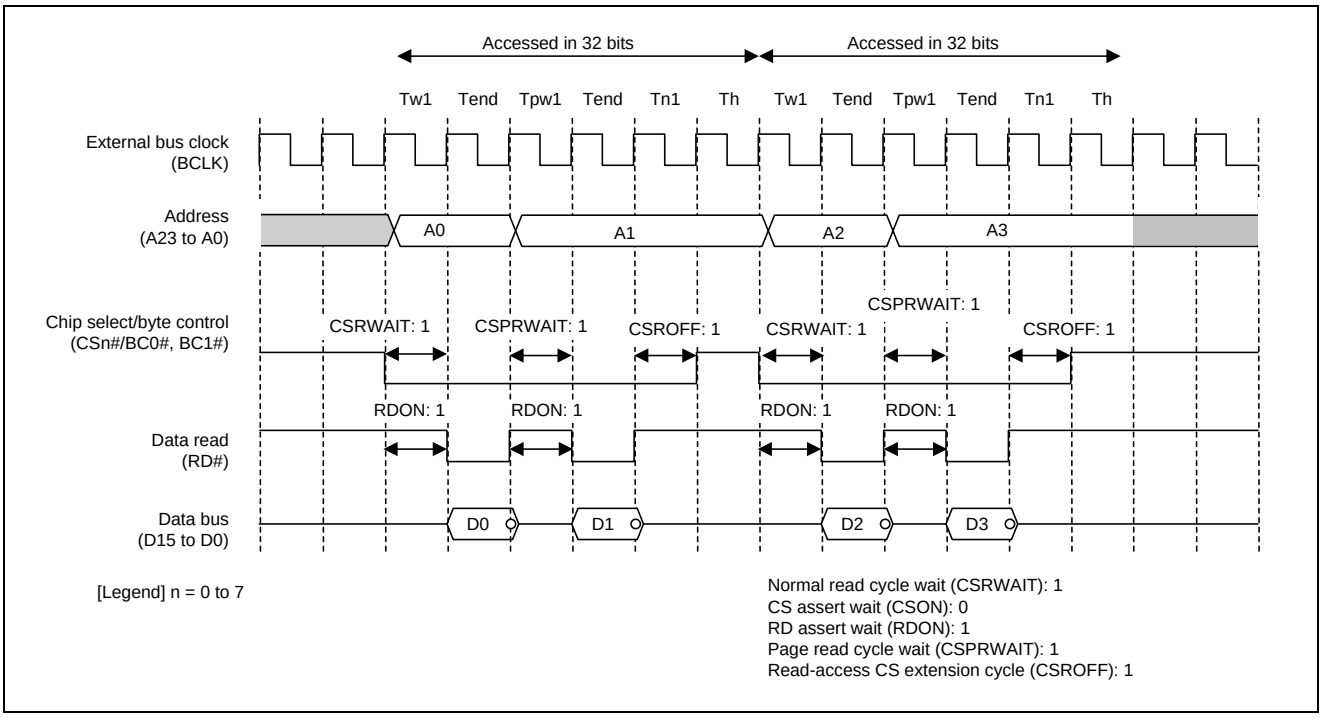



**Figure 11.17 Example of Page-Read Access Operation (when 16-Bit Bus Space is Accessed in 32 Bits)**



Downloaded from Arrow.com.

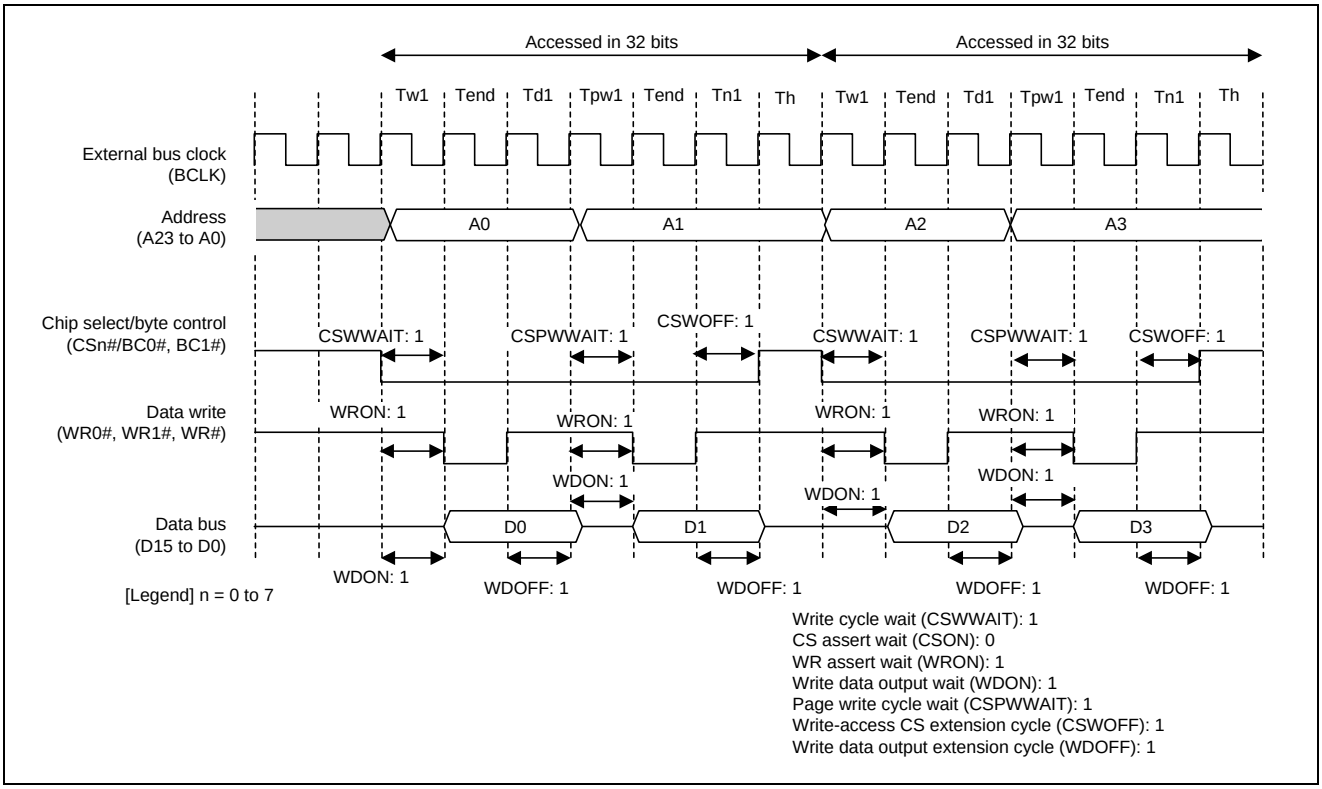



**Figure 11.18 Example of Page-Write Access Operation (when 16-Bit Bus Space is Accessed in 32 Bits)**



Downloaded from Arrow.com.

### 11.5.2 External Wait Function

Wait cycles can be extended by the WAIT# signal over the length of normal access cycle wait (specified by the CSRWAIT[4:0] and CSWWAIT[4:0] bits in CSiWCNT1) and page access cycle wait (specified by the CSPRWAIT[2:0] and CSPWWAIT[2:0] bits in CSiWCNT1).

When external wait is enabled (the EWENB bit = 1 in CSiMOD), wait cycles are inserted while the WAIT# signal is held low. When external wait is disabled (the EWENB bit = 0 in CSiMOD), the WAIT# signal has no effect.

All wait cycles specified in CSiWCNT1 are inserted independently of the WAIT# signal.

#### 11.5.2.1 Normal Access

Sampling of the WAIT# signal begins upon completion of the wait cycle (Tend) specified in CSiWCNT1. The bus cycle is extended while the WAIT# signal is held low. The wait cycle ends (Tend) at the next cycle after the WAIT# signal becomes high.

#### 11.5.2.2 Page Access

The first data read or data write operation is the same as the normal read or write operation. Sampling of the WAIT# signal begins upon completion of the wait cycle (Tend) specified in the CSiWCNT1 register. The bus cycle is extended while the WAIT# signal is held low. The wait cycle ends (Tend) at the next cycle after the WAIT# signal becomes high.

With respect to the second and subsequent read accesses, sampling of the WAIT# signal begins upon completion of the wait cycle of the page access. The wait cycle of the page access is extended while the WAIT# signal is held low, and ends (Tend) at the next cycle after the WAIT# signal becomes high.

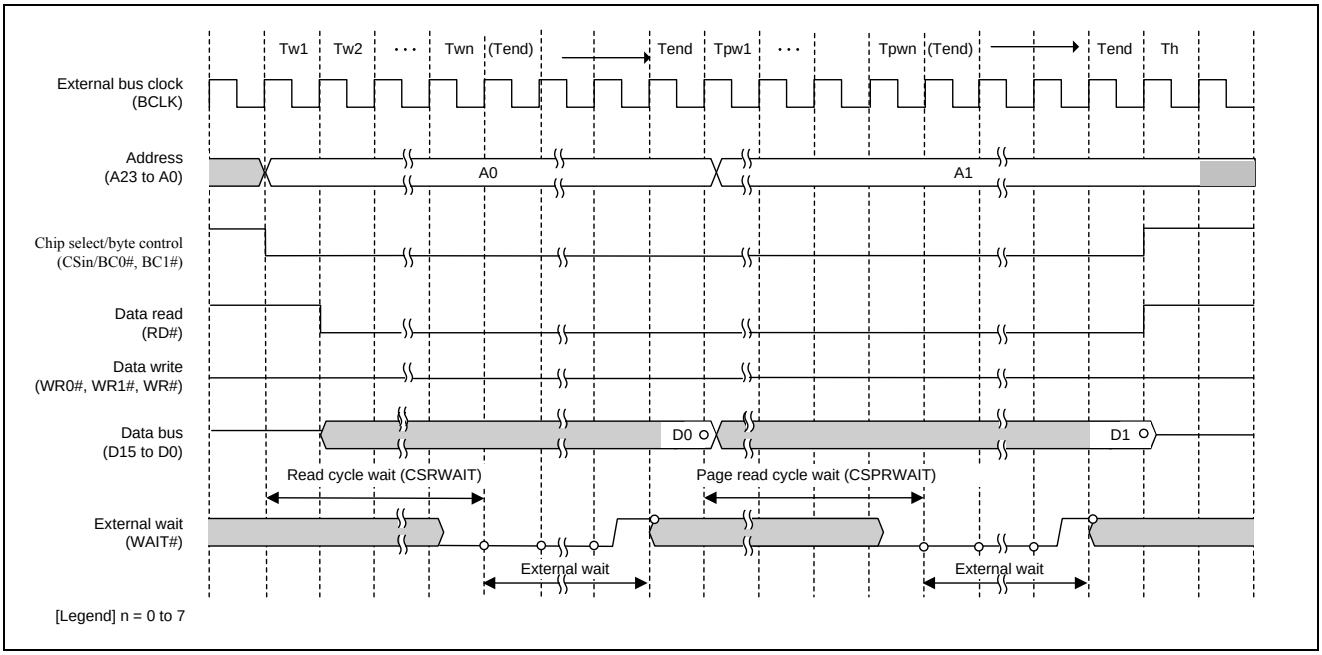



**Figure 11.19 Example of External Wait Timing (Page-Read Access to 16-Bit Bus Space)**



Downloaded from Arrow.com.

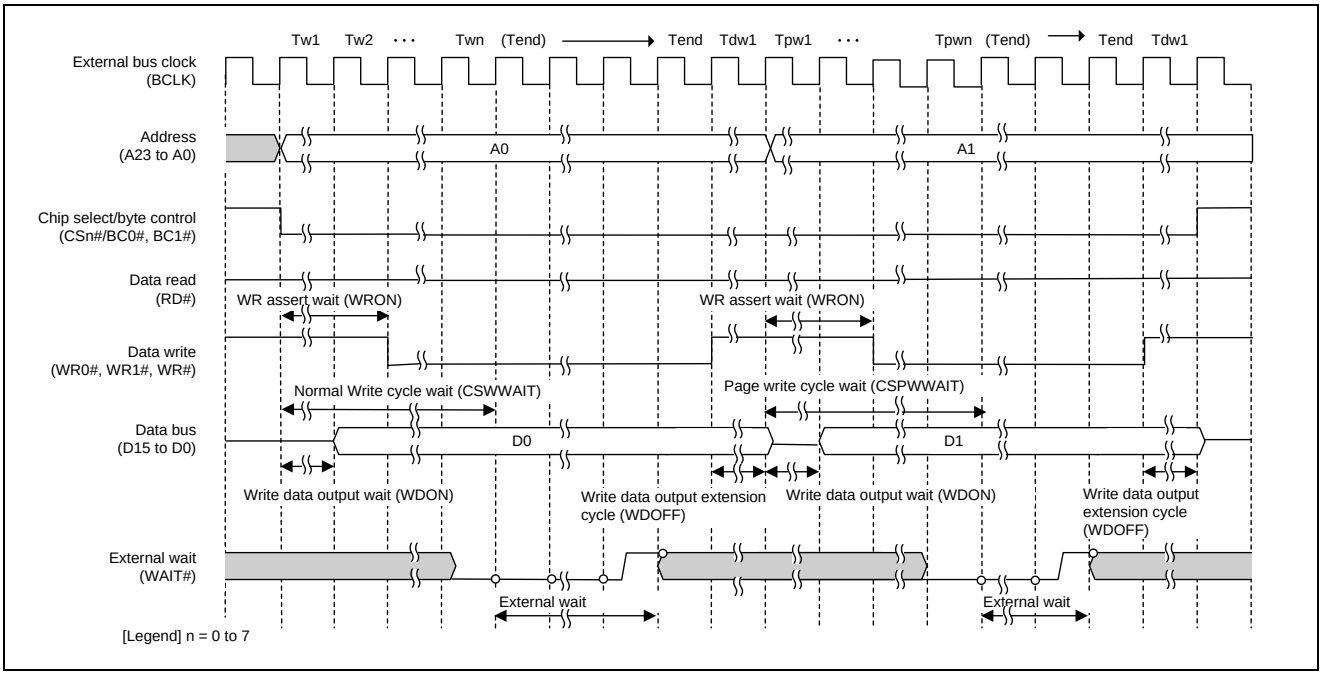



**Figure 11.20 Example of External Wait Timing (Page-Write Access to 16-Bit Bus Space)**



Downloaded from Arrow.com.

### 11.5.3 Insertion of Recovery Cycles

Clock cycles of recovery can be inserted between consecutive rounds of external bus access. Conditions where cycles of recovery can be inserted are described below.

- Write access via the external bus follows read access via the external bus.
- Read access via the external bus follows read access to a different area via the external bus.
- Read access via the external bus follows write access via the external bus.

Clock cycles of recovery are not inserted between write access and subsequent write access.

Separate numbers of clock cycles for recovery can be set up for insertion after cycles of reading or writing. The number of bus-clock cycles to be inserted for recovery following a write cycle is set by the WRCV[3:0] bits in CSiREC for the area to which a write access has been performed in the preceding bus cycle, and the number of bus-clock cycles to be inserted as the recovery cycle that follows a read cycle is set by the RRCV[3:0] bits in CSiREC for the area to which a read access has been performed in the preceding bus cycle. In other words, when a read access is made to CS0 and then to CS1, the number of recovery cycles to be inserted between these two accesses is determined by the PRCV[3:0] bits in CS0REC.

The clock cycles of recovery begin at the end of the preceding bus cycle, i.e. when the CSi# signal (i = 0 to 7) is negated. From this point, the selected period over which the CSi# signal is at the high level (i.e. the recovery period) is inserted.

After the end of the clock cycles of recovery, the CSi# signal is asserted for the next round of bus access after the insertion of no less than one bus-clock cycle as an idle cycle. Even if the next request for access to an external address space is generated during the recovery period, the next round of access over the external bus will only start after one bus-clock cycle has been inserted as an idle cycle following the end of the recovery period.

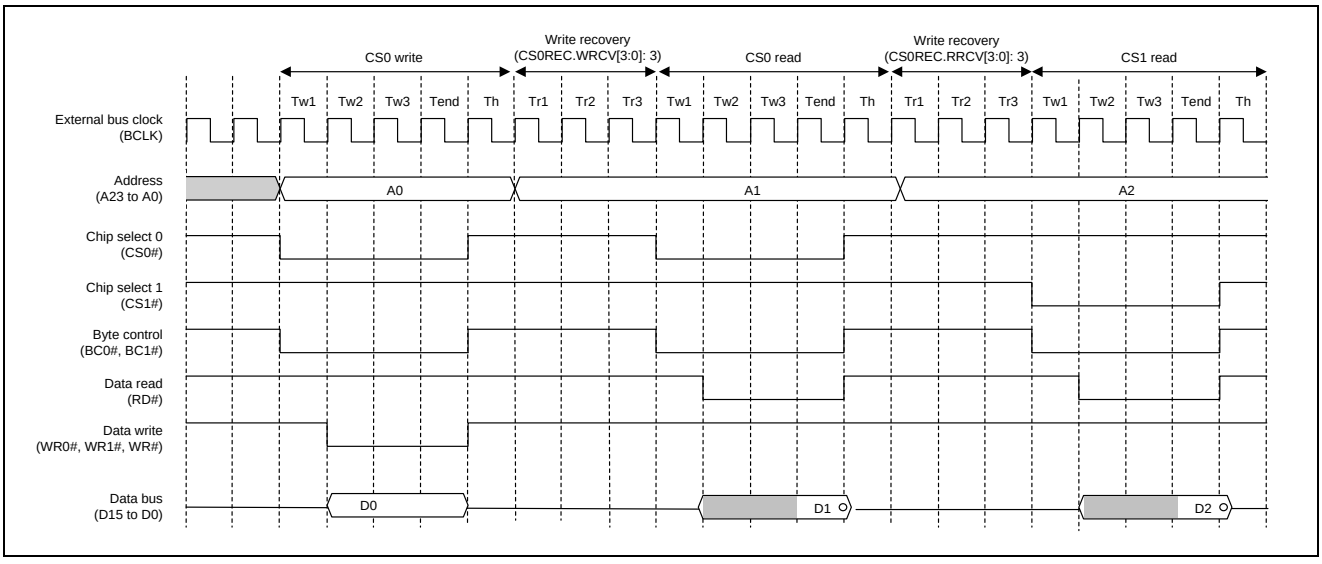



**Figure 11.21 Example of Recovery Timing**



Downloaded from Arrow.com.

### 11.5.4 Write Buffer Function

The internal main bus is released by writing data to the write buffer before the write access is completed, which allows the next round of bus access to start. However, if the following round of bus access is to an external address space or to a control register for the external bus, it is suspended until the external bus operations already in progress are completed.

Figure 11.22 shows an example of operation when the write-buffer function is in use. When this function is in use, if the next operation after an external write is internal access, the internal access (access to on-chip memory or a peripheral module) is executed in parallel with the external write, i.e. without waiting for completion of the latter operation.

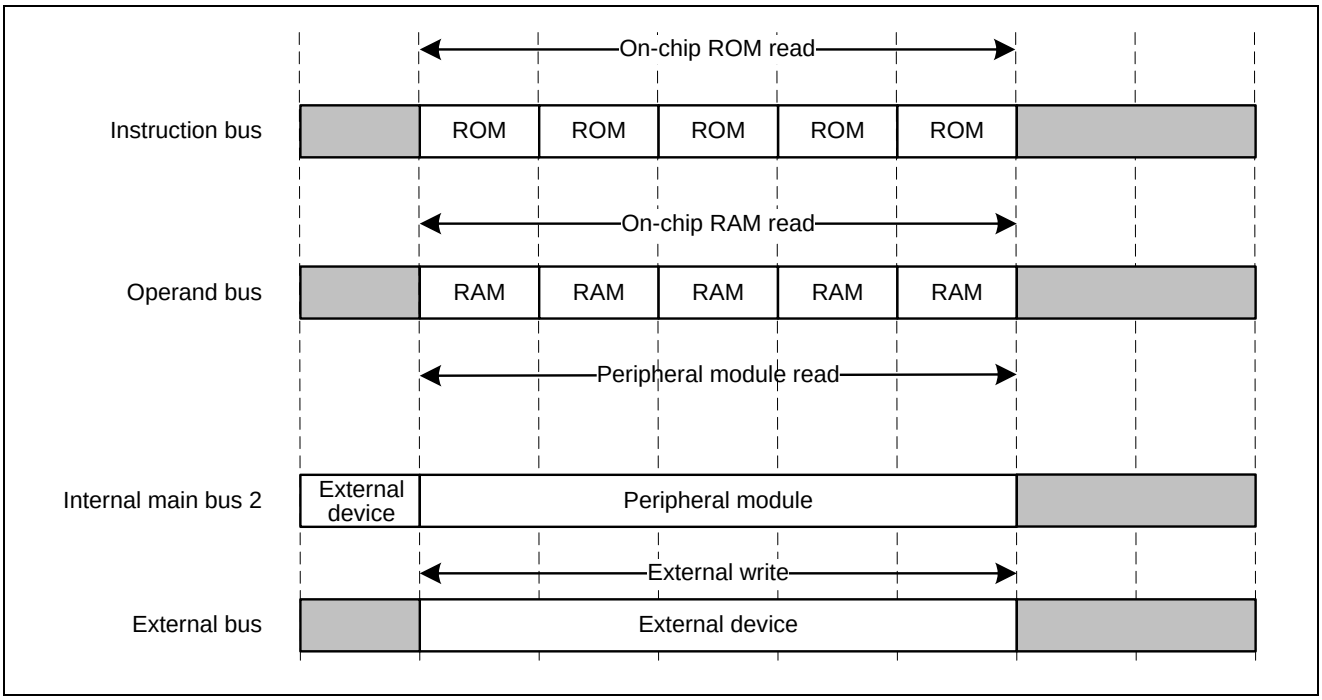



**Figure 11.22 Example of Operation when the Write-Buffer Function is in Use**



Downloaded from Arrow.com.

## 11.5.5 Notes on Usage

### 11.5.5.1 Limitations at the Time of Normal and Page Access

Limitations that apply to various bits of CSi wait control register 1 (CSiWCNT1) and CSi wait control register 2 (CSiWCNT2) at the times of normal and page access are listed in table 11.9.

If the setting of the page-read access enable bit in the CSi mode register or the page-write access enable bit in the CSi mode register selects permission (CSiMOD.PRENB = 1 or CSiMOD.PWENB = 1), the limitations on normal access must be satisfied in the first round of access for page access or in access that does not fall within the scope of page access and thus leads to normal access operations. For details on the situations that are not within the scope of page access, see section 11.5.1, Timing of External Bus Access.

**Table 11.9 Limitations at the Time of Normal and Page Access**

| Limitations at the Time of Normal Access | | Limitations at the Time of Page Access | |
|---|---|---|---|
| **Reading** | **Writing** | **Reading** | **Writing** |
| CSON[2:0] ≤ CSRWAIT | CSON[2:0] ≤ CSWWAIT | CSON[2:0] ≤ CSPRWAIT | CSON[2:0] ≤ CSPWWAIT |
| RDON[2:0] ≤ CSRWAIT | WRON[2:0] ≤ CSWWAIT | RDON[2:0] ≤ CSPRWAIT | WRON[2:0] ≤ CSPWWAIT |
| CSON[2:0] ≤ RDON | WDON[2:0] ≤ CSWWAIT | CSON[2:0] ≤ RDON | WDON[2:0] ≤ CSPWWAIT |
| | WDOFF[2:0] ≤ CSWOFF | | WDOFF[2:0] ≤ CSWOFF |
| | WDON[2:0] ≤ WRON | | WDON[2:0] ≤ WRON |
| | CSON[2:0] ≤ WRON | | CSON[2:0] ≤ WRON |

### 11.5.5.2 Prohibition of Access that Spans Areas of Address Space

Single access operations that span areas of the address space are prohibited, and operation in the case of attempts at such access is not guaranteed. In cases where access to a single word or longword would produce access that crosses a boundary between areas, split the instruction so that separate instructions are used for access to each of the areas.

### 11.5.5.3 Restrictions in Relation to RMPA and String-Manipulation Instructions

- Although the external address space has a per-area ending-switching facility (only for data), the allocation of data to be handled by RMPA or string-manipulation instructions to an area where the endian differs from that of the chip is prohibited, and operation is not guaranteed if this restriction is not observed. If data to be handled by RMPA or string-manipulation instructions are allocated to the external address space, they must be allocated to areas where the endian setting is the same as that for the chip.
- The allocation of data to be handled by RMPA or string-manipulation instructions to I/O registers is prohibited, and operation is not guaranteed if this restriction is not observed.

### 11.5.5.4 Point for Caution Regarding Register Settings

- Note on setting that affects the write data hold time (tWDH)
  To secure the write data hold time, ensure that the setting of the CSiWCNT2.WDOFF[2:0] bits (write data output extension cycle select bits) is at least 1. Take care to ensure that this condition is met, since not meeting the condition may make securing the write data hold time impossible.



Downloaded from Arrow.com.

#### 11.5.5.5 Restriction on Instruction Code

When the endian setting for an area is different from that for the chip, no instruction code can be arranged in the area. The instruction code should be arranged in the external address space whose endian setting is the same as that for the chip.



Downloaded from Arrow.com.

## 11.6 Bus Error Monitoring Section

The bus-error monitoring section monitors the individual areas for bus errors, and generates an interrupt when it detects a bus error.

### 11.6.1 Types of Bus Error

There are two types of bus error: illegal address access and time-out.

Illegal address access is the detection of illegal access to an area, and time-out is the detection of a bus-access operation not being completed within 768 cycles.

#### 11.6.1.1 Illegal Address Access

When the illegal address access detection enable bit in the bus-error enable register is set (BEREN.IGAEN = 1), access of the following types leads to illegal address access errors.

- Access to areas of external address space for which operation has been disabled (CSiCNT.EXENB = 0; i = 0 to 7)
- Access to illegal address ranges other than in areas for which operation has been disabled

The address ranges where access will lead to illegal address access errors are indicated in table 11.10.

#### 11.6.1.2 Time-out

When the time-out detection enable bit in the bus-error enable register is set (BEREN.TOEN = 1), bus access that is not completed within 768 cycles leads to a time-out error. Cycles of the operating clock of the relevant slave are counted as the number of cycles.

- External address space: Once a bus-access operation has started, access is not completed (the WAIT# signal is not negated) within 768 cycles.

Note: In products of the RX610 Group, time-out errors are only generated for access to external address space.

### 11.6.2 Operations When a Bus Error Occurs

Generation of an interrupt (BUSERR) can be selected by setting the bus error notification bit (BERIE.CPEN = 1).



Downloaded from Arrow.com.

## 11.6.3 Conditions Leading to Bus Errors

Table 11.10 lists the types of bus errors for each area in the respective address space.

**Table 11.10 Types of Bus Errors**

<table>
<tr><th rowspan="4">Address</th><th colspan="2" rowspan="3">Type of Area</th><th colspan="4">Type of Error</th></tr>
<tr><th colspan="2">Illegal Address Access</th><th colspan="2">Time-out</th></tr>
<tr><th colspan="2">On-chip ROM Mode</th><th colspan="2">On-chip ROM mode</th><th colspan="2">On-chip ROM Mode</th></tr>
<tr><th>Enabled</th><th>Disabled</th><th>Enabled</th><th>Disabled</th><th>Enabled</th><th>Disabled</th></tr>
<tr><td>0000 0000h to 0001 FFFFh</td><td colspan="2">On-chip RAM</td><td colspan="2" rowspan="2">—</td><td colspan="2" rowspan="2">—</td></tr>
<tr><td>0002 0000h to 0007 FFFFh</td><td colspan="2">Reserved area</td></tr>
<tr><td>0008 0000h to 0008 FFFFh</td><td colspan="2" rowspan="2">Peripheral I/O registers</td><td colspan="2">—</td><td colspan="2">—</td></tr>
<tr><td>0009 0000h to 000F FFFFh</td><td colspan="2">√</td><td colspan="2">—</td></tr>
<tr><td>0010 0000h to 0011 FFFFh</td><td>Data flash</td><td rowspan="10">Reserved area</td><td>—</td><td rowspan="10">√</td><td>—</td><td rowspan="10">—</td></tr>
<tr><td>0012 0000h to 007F 7FFFh</td><td>Reserved area</td><td>√</td><td>—</td></tr>
<tr><td>007F 8000h to 007F 9FFFh</td><td>FCU-RAM</td><td>—</td><td>—</td></tr>
<tr><td>007F A000h to 007F BFFFh</td><td>Reserved area</td><td>√</td><td>—</td></tr>
<tr><td>007F C000h to 007F C4FFh</td><td>Peripheral I/O registers</td><td>—</td><td>—</td></tr>
<tr><td>007F C500h to 007F FBFFh</td><td>Reserved area</td><td>√</td><td>—</td></tr>
<tr><td>007F FC00h to 00FF FFFFh</td><td>Peripheral I/O registers</td><td rowspan="3">—</td><td rowspan="3">—</td></tr>
<tr><td>0080 0000h to 00DF FFFFh</td><td>Reserved area</td></tr>
<tr><td>00E0 0000h to 00FF FFFFh</td><td>On-chip ROM (dedicated area for writing)</td></tr>
<tr><td>0100 0000h to 07FF FFFFh</td><td colspan="2">External address space (CS1 to CS7)</td><td colspan="2">√*1</td><td colspan="2">√*2</td></tr>
<tr><td>0800 0000h to 7FFF FFFFh</td><td colspan="2">Reserved area</td><td colspan="2">√</td><td colspan="2">—</td></tr>
<tr><td>8000 0000h to FEFF FFFFh</td><td rowspan="2">On-chip ROM (dedicated area for reading)</td><td>Reserved area</td><td rowspan="2">—</td><td>√</td><td rowspan="2">—</td><td>—</td></tr>
<tr><td>FF00 0000h to FFFF FFFFh</td><td>External address space (CS0)</td><td>√*1</td><td>√*2</td></tr>
</table>

[Legend] —: A bus error is not produced.
√: A bus error is produced.
1. Access to this area leads to the detection of a bus error if operation has been disabled (CSiCNT.EXENB = 0; i = 0 to 7).
2. Bus access not being completed (the WAIT# signal not being negated) within 768 cycles leads to detection of a bus error.



Downloaded from Arrow.com.

# 12. DMA Controller (DMAC)

The RX610 Group incorporates a 4-channel direct memory access controller (DMAC).

The DMAC is a module to transfer data without the CPU.

When a DMA transfer request is generated, the DMAC transfers data stored at the transfer source address to the transfer destination address.

## 12.1 Overview

Table 12.1 lists the specifications of the DMAC, and figure 12.1 shows a block diagram of the DMAC.

**Table 12.1 Specifications of DMAC**

| Item | | | Description |
|---|---|---|---|
| Number of channels | | | 4 (DMACm (m = 0 to 3)) |
| Transfer space | | | 4 Gbytes (00000000h to FFFFFFFFh excluding reserved areas) |
| Maximum transfer volume | | | 64 Mbytes |
| DMA activation source | | | Software trigger<br>Trigger input to external pin interrupts<br>Interrupt requests from peripheral functions |
| Channel priority | | | Channel 0 > Channel 1 > Channel 2 > Channel 3 (Channel 0: Highest) |
| Transfer data | Single data | | Bit length: 8, 16, 32 bits |
| | Single operand | | Data count: 1, 2, 4, 8, 16, 32, 64, 128 |
| Transfer system | Operand transfer system | Single | Data of a single operand is transferred per DMA transfer request.<br>Channel arbitration is made after a single-operand transfer.<br>A DMA transfer request is necessary at each end of single-operand transfer until the DMA transfer end. |
| | | Consecutive | Data is transferred continuously in operand units until the DMA transfer end per DMA transfer request.<br>Channel arbitration is made after a single-operand transfer.<br>A DMA transfer request is made first only once. |
| | Nonstop transfer system | | Data is transferred continuously until the DMA transfer ends per DMA transfer request.<br>Channel arbitration is made when DMA transfer is completed.<br>A DMA transfer request is made first only once. |
| DMA transfer start condition | | | DMA transfer starts when all the following conditions are met.<br>The DEN bit in DMCRE of DMACm is 1 (DMA transfer enabled).<br>The DMST bit in DMSCNT is 1 (DMAC start).<br>When a DMA transfer request of channel m (DMACm) is generated and the execution authority is obtained by the channel arbitration. |
| DMA transfer end condition | | | When the DMCBC register of DMACm is decremented to 0000000h |
| Interrupt request generation timing | | | When the DMCBC register of DMACm is decremented to 0000000h |
| Single-data transfer time | | | Three bus clock cycles or more |



Downloaded from Arrow.com.

| Item | Description |
|---|---|
| Selective function | Reload function<br>Reloads the values in reload registers (transfer source address, transfer destination address, transfer byte count) to current registers (transfer source address, transfer destination address, transfer byte count) at the end of DMA transfer. |

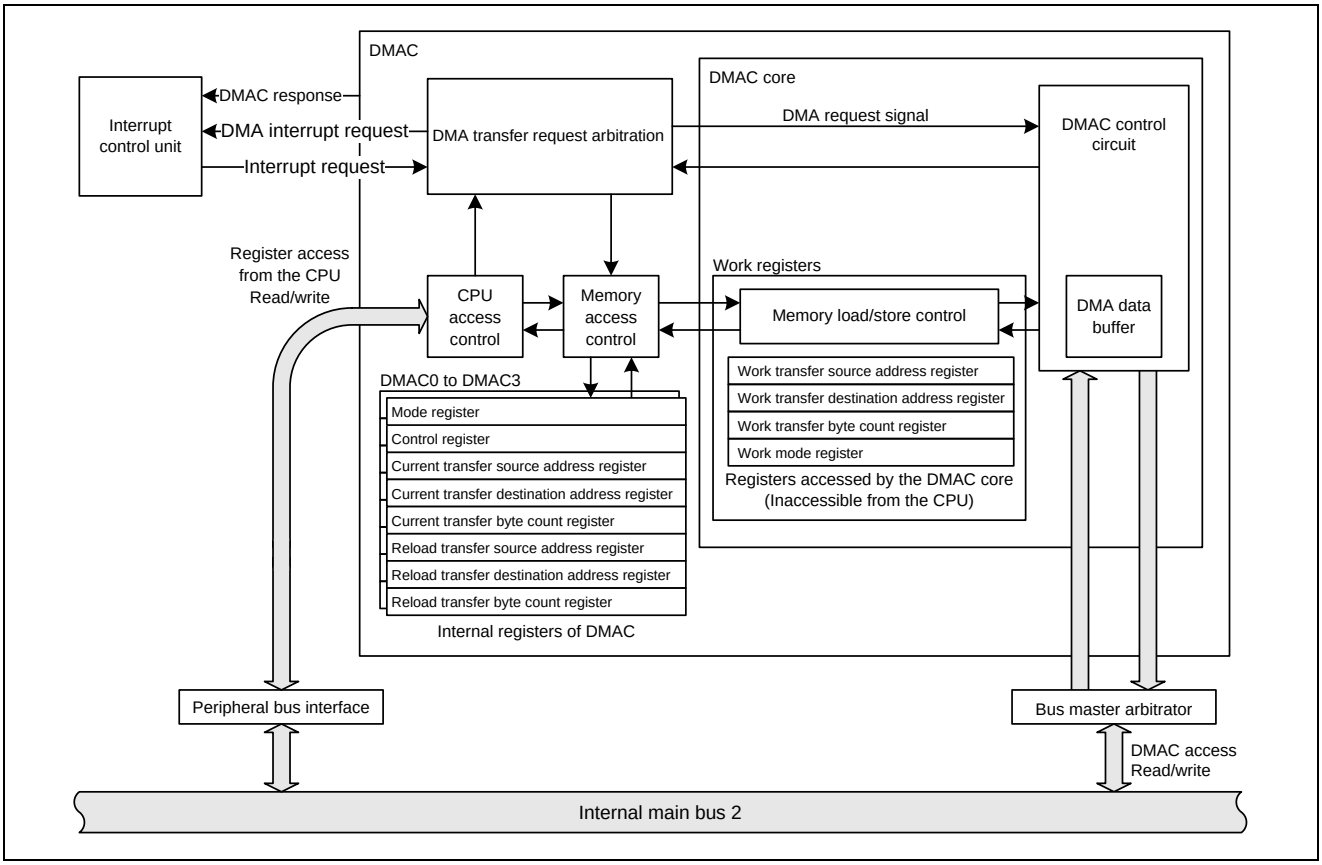



**Figure 12.1 Block Diagram of DMAC**



Downloaded from Arrow.com.

## 12.2 Register Descriptions

Table 12.2 lists the registers of the DMAC. Registers of DMAC0 to DMAC3 have same functions.

**Table 12.2 Registers of DMAC**

| Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| DMAC0 | DMA mode register | DMMOD | 0x0x xx00h | 0008 200Ch | 32 |
| | DMA control register A | DMCRA | 0000 0000h | 0008 2400h | 32 |
| | DMA control register B | DMCRB | 00h | 0008 2404h | 8 |
| | DMA control register C | DMCRC | 00h | 0008 2405h | 8 |
| | DMA control register D | DMCRD | 00h | 0008 2406h | 8 |
| | DMA control register E | DMCRE | 00h | 0008 2407h | 8 |
| | DMA current transfer source address register | DMCSA | xxxx xxxxh | 0008 2000h | 32 |
| | DMA current transfer destination address register | DMCDA | xxxx xxxxh | 0008 2004h | 32 |
| | DMA current transfer byte count register | DMCBC | 0xxx xxxxh | 0008 2008h | 32 |
| | DMA reload transfer source address register | DMRSA | xxxx xxxxh | 0008 2200h | 32 |
| | DMA reload transfer destination address register | DMRDA | xxxx xxxxh | 0008 2204h | 32 |
| | DMA reload transfer byte count register | DMRBC | 0xxx xxxxh | 0008 2208h | 32 |
| DMAC1 | DMA mode register | DMMOD | 0x0x xx00h | 0008 201Ch | 32 |
| | DMA control register A | DMCRA | 0000 0000h | 0008 2408h | 32 |
| | DMA control register B | DMCRB | 00h | 0008 240Ch | 8 |
| | DMA control register C | DMCRC | 00h | 0008 240Dh | 8 |
| | DMA control register D | DMCRD | 00h | 0008 240Eh | 8 |
| | DMA control register E | DMCRE | 00h | 0008 240Fh | 8 |
| | DMA current transfer source address register | DMCSA | xxxx xxxxh | 0008 2010h | 32 |
| | DMA current transfer destination address register | DMCDA | xxxx xxxxh | 0008 2014h | 32 |
| | DMA current transfer byte count register | DMCBC | 0xxx xxxxh | 0008 2018h | 32 |
| | DMA reload transfer source address register | DMRSA | xxxx xxxxh | 0008 2210h | 32 |
| | DMA reload transfer destination address register | DMRDA | xxxx xxxxh | 0008 2214h | 32 |
| | DMA reload transfer byte count register | DMRBC | 0xxx xxxxh | 0008 2218h | 32 |
| DMAC2 | DMA mode register | DMMOD | 0x0x xx00h | 0008 202Ch | 32 |
| | DMA control register A | DMCRA | 0000 0000h | 0008 2410h | 32 |
| | DMA control register B | DMCRB | 00h | 0008 2414h | 8 |
| | DMA control register C | DMCRC | 00h | 0008 2415h | 8 |
| | DMA control register D | DMCRD | 00h | 0008 2416h | 8 |
| | DMA control register E | DMCRE | 00h | 0008 2417h | 8 |
| | DMA current transfer source address register | DMCSA | xxxx xxxxh | 0008 2020h | 32 |
| | DMA current transfer destination address register | DMCDA | xxxx xxxxh | 0008 2024h | 32 |
| | DMA current transfer byte count register | DMCBC | 0xxx xxxxh | 0008 2028h | 32 |
| | DMA reload transfer source address register | DMRSA | xxxx xxxxh | 0008 2220h | 32 |
| | DMA reload transfer destination address register | DMRDA | xxxx xxxxh | 0008 2224h | 32 |
| | DMA reload transfer byte count register | DMRBC | 0xxx xxxxh | 0008 2228h | 32 |



Downloaded from Arrow.com.

| Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| DMAC3 | DMA mode register | DMMOD | 0x0x xx00h | 0008 203Ch | 32 |
| | DMA control register A | DMCRA | 0000 0000h | 0008 2418h | 32 |
| | DMA control register B | DMCRB | 00h | 0008 241Ch | 8 |
| | DMA control register C | DMCRC | 00h | 0008 241Dh | 8 |
| | DMA control register D | DMCRD | 00h | 0008 241Eh | 8 |
| | DMA control register E | DMCRE | 00h | 0008 241Fh | 8 |
| | DMA current transfer source address register | DMCSA | xxxx xxxxh | 0008 2030h | 32 |
| | DMA current transfer destination address register | DMCDA | xxxx xxxxh | 0008 2034h | 32 |
| | DMA current transfer byte count register | DMCBC | 0xxx xxxxh | 0008 2038h | 32 |
| | DMA reload transfer source address register | DMRSA | xxxx xxxxh | 0008 2230h | 32 |
| | DMA reload transfer destination address register | DMRDA | xxxx xxxxh | 0008 2234h | 32 |
| | DMA reload transfer byte count register | DMRBC | 0xxx xxxxh | 0008 2238h | 32 |
| DMAC0 to DMAC3 | DMA interrupt control register | DMICNT | 00h | 0008 250Bh | 8 |
| | DMA start control register | DMSCNT | 00h | 0008 2502h | 8 |
| | DMA arbitration status register | DMASTS | 00h | 0008 251Bh | 8 |
| | DMA transfer end detect register | DMEDET | 00h | 0008 2517h | 8 |

Note: x: Undefined



Downloaded from Arrow.com.

## 12.2.1 DMA Mode Register (DMMOD)

Addresses: DMAC0.DMMOD 0008 200Ch, DMAC1.DMMOD 0008 201Ch
DMAC2.DMMOD 0008 202Ch, DMAC3.DMMOD 0008 203Ch

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | OPSEL[3:0] | | | | — | — | — | — | — | SZSEL[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | x | x | x | x | 0 | 0 | 0 | 0 | 0 | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SMOD[2 0] | | | — | DMOD[2:0] | | | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | x | x | x | 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b10 to b8 | DMOD[2:0] | Transfer Destination Address Addition Direction Select | b10 b9 b8<br>0 0 0: Fixed<br>0 0 1: Plus<br>0 1 0: Minus<br>0 1 1: Rotate<br>Do not write other values. | R/W |
| b11 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b14 to b12 | SMOD[2:0] | Transfer Source Address Addition Direction Select | b14 b13 b12<br>0 0 0: Fixed<br>0 0 1: Plus<br>0 1 0: Minus<br>0 1 1: Rotate<br>Do not write other values. | R/W |
| b15 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b18 to b16 | SZSEL[2:0] | Transfer Data Size Select | b18 b17 b16<br>0 0 0: 8 bits<br>0 0 1: 16 bits<br>0 1 0: 32 bits<br>Do not write other values. | R/W |
| b23 to b19 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b27 to b24 | OPSEL[3:0] | Operand Transfer Data Count Select | b27 b26 b25 b24<br>0 0 0 0: Single data<br>0 0 0 1: 2 data<br>0 0 1 0: 4 data<br>0 0 1 1: 8 data<br>0 1 0 0: 16 data<br>0 1 0 1: 32 data<br>0 1 1 0: 64 data<br>0 1 1 1: 128 data<br>Do not write other values. | R/W |
| b31 to b28 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMMOD is used to set the addition direction of transfer source and destination addresses and to set the size of transfer data.

Do not set DMMOD during data transfer, but set it while the DMAC is not active or DMA transfer is disabled.

Access this register with 32 bits.

### DMOD[2:0] Bits (Transfer Destination Address Addition Direction Select)
### SMOD[2:0] Bits (Transfer Source Address Addition Direction Select)

These bits select the addition direction of transfer source and destination addresses during DMA transfer.

When "rotate" is selected, address is added in the plus direction and is returned to the value specified at the beginning of DMA transfer when single-operand transfer is completed.

Transfer source and destination addresses increase or decrease according to the addition direction and data size settings as shown in table 12.3.

**Table 12.3 Address Increase/Decrease According to Addition Direction and Data Size**

| | SMOD[2:0] Bits/DMOD[2:0] Bits | | | |
|---|---|---|---|---|
| **SZSEL[2:0] Bits** | **000b (Fixed)** | **001b (Plus)** | **010b (Minus)** | **011b (Rotate)** |
| 000b (8 bits) | ±0 | +1 | -1 | +1 |
| 001b (16 bits) | ±0 | +2 | -2 | +2 |
| 010b (32 bits) | ±0 | +4 | -4 | +4 |

### SZSEL[2:0] Bits (Transfer Data Size Select)

The SZSEL[2:0] bits select the bit length of transfer data.

### OPSEL[3:0] Bits (Operand Transfer Data Count Select)

The OPSEL[3:0] bits select the data count of single-operand transfer.

When the operand transfer system is used, data of the volume specified by the OPSEL[3:0] bits is continuously transferred as a single operand.

When the nonstop transfer system is used, data of the volume specified by the DMA current transfer byte count register (DMCBC) of DMACm is continuously transferred independently of the OPSEL[3:0] setting.



Downloaded from Arrow.com.

## 12.2.2 DMA Control Register A (DMCRA)

Addresses: DMAC0.DMCRA 0008 2400h, DMAC1.DMCRA 0008 2408h
DMAC2.DMCRA 0008 2410h, DMAC3.DMCRA 0008 2418h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DSEL[1:0] | | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | BRLOD | SRLOD | DRLOD | — | — | DCTG[5:0] | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b5 to b0 | DCTG[5:0] | DMA Activation Source Select | Select a DMA activation source. (See table 12.4.) | R/W |
| b7, b6 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b8 | DRLOD | Transfer Destination Address Reload Function Select | 0: Transfer destination address reload function is not used<br>1: Transfer destination address reload function is used | R/W |
| b9 | SRLOD | Transfer Source Address Reload Function Select | 0: Transfer source address reload function is not used<br>1: Transfer source address reload function is used | R/W |
| b10 | BRLOD | Transfer Byte Count Reload Function Select | 0: Transfer byte count reload function is not used<br>1: Transfer byte count reload function is used | R/W |
| b23 to b11 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b25, b24 | DSEL[1:0] | Transfer System Select | b25 b24<br>0 0: Single-operand transfer<br>0 1: Consecutive-operand transfer<br>1 0: Setting prohibited<br>1 1: Nonstop transfer | R/W |
| b31 to b26 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCRA is used to control DMAC functions.

### DCTG[5:0] Bits (DMA Activation Source Select)

The DCTG[5:0] bits select a DMA activation source.

Do not set these bits during data transfer, but set them while the DMAC is not active or DMA transfer is disabled. After these bits are set, be sure to set the DREQ bit in DMCRD of DMACm to 0 (no DMA transfer request) and then activate the DMAC and enable DMA transfer.

Table 12.4 shows the setting of the DCTG[5:0] bits.



Downloaded from Arrow.com.

**Table 12.4 Setting of DCTG[5:0] Bits**

| | DMA Activation Source | | | |
|---|---|---|---|---|
| **DCTG[5:0] Bits** | **DMA0** | **DMA1** | **DMA2** | **DMA3** |
| 000000 | Software trigger | | | |
| 000001 | CMI0 (CMT0 compare match interrupt of compare match timer unit 0) | | | |
| 000010 | CMI1 (CMT1 compare match interrupt of compare match timer unit 0) | | | |
| 000011 | CMI2 (CMT2 compare match interrupt of compare match timer unit 1) | | | |
| 000100 | CMI3 (CMT3 compare match interrupt of compare match timer unit 1) | | | |
| 000101 | IRQ0 (external pin interrupt) | | | |
| 000110 | IRQ1 (external pin interrupt) | | | |
| 000111 | IRQ2 (external pin interrupt) | | | |
| 001000 | IRQ3 (external pin interrupt) | | | |
| 001001 | ADI0 (ADC interrupt of A/D converter unit 0) | | | |
| 001010 | ADI1 (ADC interrupt of A/D converter unit 1) | | | |
| 001011 | ADI2 (ADC interrupt of A/D converter unit 2) | | | |
| 001100 | ADI3 (ADC interrupt of A/D converter unit 3) | | | |
| 001101 | TGI0A (TPU0 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 001110 | TGI1A (TPU1 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 001111 | TGI2A (TPU2 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 010000 | TGI3A (TPU3 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 010001 | TGI4A (TPU4 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 010010 | TGI5A (TPU5 input capture/compare match interrupt of 16-bit timer pulse unit 0) | | | |
| 010011 | TGI6A (TPU6 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 010100 | TGI7A (TPU7 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 010101 | TGI8A (TPU8 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 010110 | TGI9A (TPU9 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 010111 | TGI10A (TPU10 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 011000 | TGI11A (TPU11 input capture/compare match interrupt of 16-bit timer pulse unit 1) | | | |
| 011001 | RXI0 (receive data full interrupt of serial communications interface SCI0) | | | |
| 011010 | TXI0 (transmit data empty interrupt of serial communications interface SCI0) | | | |
| 011011 | RXI1 (receive data full interrupt of serial communications interface SCI1) | | | |
| 011100 | TXI1 (transmit data empty interrupt of serial communications interface SCI1) | | | |
| 011101 | RXI2 (receive data full interrupt of serial communications interface SCI2) | | | |
| 011110 | TXI2 (transmit data empty interrupt of serial communications interface SCI2) | | | |
| 011111 | RXI3 (receive data full interrupt of serial communications interface SCI3) | | | |
| 100000 | TXI3 (transmit data empty interrupt of serial communications interface SCI3) | | | |
| 100001 | RXI4 (receive data full interrupt of serial communications interface SCI4) | | | |
| 100010 | TXI4 (transmit data empty interrupt of serial communications interface SCI4) | | | |
| 100011 | RXI5 (receive data full interrupt of serial communications interface SCI5) | | | |
| 100100 | TXI5 (transmit data empty interrupt of serial communications interface SCI5) | | | |
| 100101 | RXI6 (receive data full interrupt of serial communications interface SCI6) | | | |
| 100110 | TXI6 (transmit data empty interrupt of serial communications interface SCI6) | | | |
| 100111 | ICRXI0 (receive data full interrupt of $I^2C$ bus interface RIIC0) | | | |
| 101000 | ICTXI0 (transmit data empty interrupt of $I^2C$ bus interface RIIC0) | | | |
| 101001 | ICRXI1 (receive data full interrupt of $I^2C$ bus interface RIIC1) | | | |
| 101010 | ICTXI1 (transmit data empty interrupt of $I^2C$ bus interface RIIC1) | | | |
| 101011 to 111111 | (Nothing is assigned. These settings are prohibited.) | | | |

To make an interrupt request effective as a source of DMA requests, the corresponding bit of the relevant interrupt



Downloaded from Arrow.com.

enabling register in the ICU (IERi; i = 02h to 1Fh) must be set to 1, and the relevant interrupt destination setting register of the ICU (ISELRi, where n is the interrupt vector number) must be set to select the DMAC as the destination of the interrupt signal. For details, see the sections on the interrupt controller and various peripheral modules listed below.

- Section 10, Interrupt Control Unit (ICU)
- Section 15, 16-Bit Timer Pulse Unit (TPU)
- Section 18, Compare Match Timer (CMT)
- Section 20, Serial Communications Interface (SCI)
- Section 22, $I^2C$ Bus Interface Unit (RIIC)
- Section 23, A/D Converter

#### DRLOD Bit (Transfer Destination Address Reload Function Select)

This bit controls the transfer destination address reload function.

When the DRLOD bit is set to 1, the value of the DMA reload transfer destination address register (DMRDA) of DMACm is reloaded to the DMA current transfer destination address register (DMCDA) of DMACm at the end of DMA transfer.

When this reload function is not used, set the ECLR bit in DMCRC of DMACm to 1 (the DEN bit is cleared to 0 at the end of DMA transfer) to clear the DEN bit in DMCRE of DMACm to 0 (DMA transfer disabled).

#### SRLOD Bit (Transfer Source Address Reload Function Select)

This bit controls the transfer source address reload function.

When the SRLOD bit is set to 1, the value of the DMA reload transfer source address register (DMRSA) of DMACm is reloaded to the DMA current transfer source address register (DMCSA) of DMACm at the end of DMA transfer.

When this reload function is not used, set the ECLR bit in DMCRC of DMACm to 1 to clear the DEN bit in DMCRE of DMACm to 0.

#### BRLOD Bit (Transfer Byte Count Reload Function Select)

This bit controls the transfer byte count reload function.

When the BRLOD bit is set to 1, the value of the DMA reload transfer byte count register (DMRBC) of DMACm is reloaded to the DMA current transfer byte count register (DMCBC) of DMACm at the end of DMA transfer.

When this reload function is not used, set the ECLR bit in DMCRC of DMACm to 1 to clear the DEN bit in DMCRE of DMACm to 0.

#### DSEL[1:0] Bits (Transfer System Select)

The DSEL[1:0] bits select a transfer system.

Do not set these bits during data transfer, but set them while the DMAC is not active or DMA transfer is disabled.



Downloaded from Arrow.com.

### 12.2.3 DMA Control Register B (DMCRB)

Addresses: DMAC0.DMCRB 0008 2404h, DMAC1.DMCRB 0008 240Ch
DMAC2.DMCRB 0008 2414h, DMAC3.DMCRB 0008 241Ch

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DSCLR |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DSCLR | DMAC Internal Status Clear | Writing 1 to this bit initializes the DMAC internal status.<br>Do not write 0 to this bit. This bit is always read as 0. | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCRB is used to control DMA transfer.

**DSCLR Bit (DMAC Internal Status Clear)**

This bit initializes the internal status of the DMAC.

Setting the DSCLR bit to 1 when DMA has been suspended cancels the remainder of the DMA transfer and initializes the DMAC's internal transfer state. However, no registers are initialized. Since a written "1" is not retained, this bit is always read as 0. Writing 0 has no effect.

Do not set the DSCLR bit during data transfer. Only set it while the DMAC is not active or DMA transfer is disabled.



Downloaded from Arrow.com.

### 12.2.4 DMA Control Register C (DMCRC)

Addresses: DMAC0.DMCRE 0008 2407h, DMAC1.DMCRE 0008 240Fh
DMAC2.DMCRE 0008 2417h, DMAC3.DMCRE 0008 241Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | DEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ECLR | DMA Transfer Enable Clear | 0: The DEN bit is not cleared to 0 at the end of DMA transfer.<br>1: The DEN bit is cleared to 0 at the end of DMA transfer. | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCRC is used to control DMA transfer.

#### ECLR Bit (DMA Transfer Enable Clear)

The ECLR bit controls behavior of the DEN bit in DMCRE of the given DMACm at the end of DMA transfer.

When the value of the ECLR bit is 1, the DEN bit is cleared to 0 at the end of DMA transfer, disabling subsequent DMA transfer on the given channel.

When reloading is not in use, set this bit to 1 so that the DEN bit is cleared to 0 at the end of DMA transfer.

Do not set the ECLR bit during data transfer.



Downloaded from Arrow.com.

## 12.2.5 DMA Control Register D (DMCRD)

Addresses: DMAC0.DMCRD 0008 2406h, DMAC1.DMCRD 0008 240Eh
DMAC2.DMCRD 0008 2416h, DMAC3.DMCRD 0008 241Eh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DREQ |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DREQ | DMA Transfer Request | 0: No DMA transfer request is generated<br>1: A DMA transfer request is generated | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCRD is used to control DMA transfer.

**DREQ Bit (DMA Transfer Request)**

The DREQ bit indicates whether a DMA transfer request is present or not.

The value of the DREQ bit varies according to the presence or absence of a DMA transfer request even when the DMAC is stopped or DMA transfer is disabled.

If software triggering is specified as a source of DMA transfer requests, the program generates a DMA transfer request by writing 1 to this bit.

If software triggering is not specified as a source of DMA transfer requests, the program should not write 1 to this bit.

Write 0 to the DREQ bit while data transfer is not in progress, i.e. while the DMAC is stopped or DMA transfer is disabled. Writing of 1 can proceed regardless of the state of DMA transfer.

The DREQ bit is set to 1 or cleared to 0 according to the source of DMA transfer requests as described below.

(1) When the source of DMA transfer requests is a software trigger

[Setting condition]

- The program writing 1 to the bit

[Clearing conditions]

- The program writing 0 to the bit
- The start of data transfer following acceptance of a DMA transfer request

(2) When the source of DMA transfer requests is other than a software trigger

[Setting condition]

- Detection of a source of DMA transfer requests that has been selected in the corresponding DMACm.DMCRA.DCTG[5:0] bits

[Clearing conditions]

- The program writing 0 to the bit
- The start of data transfer following acceptance of a DMA transfer request



Downloaded from Arrow.com.

## 12.2.6 DMA Control Register E (DMCRE)

Addresses: DMAC0.DMCRE 0008 2407h, DMAC1.DMCRE 0008 240Fh
DMAC2.DMCRE 0008 2417h, DMAC3.DMCRE 0008 241Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DEN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DEN | DMA Transfer Enable | 0: DMA transfer is disabled<br>1: DMA transfer is enabled | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCRE is used to control DMA transfer.

**DEN Bit (DMA Transfer Enable)**

The DEN bit enables DMA transfer.

While the ECLR bit is 1, the DEN bit is automatically cleared to 0 at the end of DMA transfer.

When the DEN bit is cleared to 0 in operand transfer mode, DMA transfer on the given channel is suspended after the current single-operand transfer is completed. The DMA transfer is restarted by setting the DEN bit to 1 again.

In nonstop transfer mode, DMA transfer is not suspended by clearing the DEN bit to 0. Instead, DMA transfer continues until it is completed.



Downloaded from Arrow.com.

## 12.2.7 DMA Current Transfer Source Address Register (DMCSA)

Addresses: DMAC0.DMCSA 0008 2000h, DMAC1.DMCSA 0008 2010h
DMAC2.DMCSA 0008 2020h, DMAC3.DMCSA 0008 2030h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b31 to b0 | Transfer source start address | 00000000h to FFFFFFFFh (4 Gbytes) | R/W |

DMCSA is used to set the start address of the transfer source.

Do not set the DMCSA register of DMACm during data transfer, but set it while the DMAC is not active or DMA transfer is disabled.

Access the DMCSA register of DMACm with 32 bits.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register so that b31 to b0 correspond to A31 to A0.

The value written to this register is transferred to the work register in the DMAC core at the beginning of DMA transfer, and the work register value is reloaded at the end of single-operand transfer or DMA transfer. However, when the SMOD[2:0] value in DMMOD of DMACm is 011b (rotate), the work register value is not reloaded and the DMCSA register value remains unchanged retaining the value that was set at the beginning of DMA transfer. When the SRLOD bit in DMCRA of DMACm is set to 1 (the transfer source address reload function is used), the value of the DMRSA register of DMACm is reloaded at the end of DMA transfer.



Downloaded from Arrow.com.

## 12.2.8 DMA Current Transfer Destination Address Register (DMCDA)

Addresses: DMAC0.DMCDA 0008 2004h, DMAC1.DMCDA 0008 2014h
DMAC2.DMCDA 0008 2024h, DMAC3.DMCDA 0008 2034h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b31 to b0 | Transfer destination start address | 00000000h to FFFFFFFFh (4 Gbytes) | R/W |

DMCDA is used to set the start address of the transfer destination.

Do not set the DMCDA register of DMACm during data transfer, but set it while the DMAC is not active or DMA transfer is disabled.

Access the DMCDA register of DMACm with 32 bits.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register so that b31 to b0 correspond to A31 to A0.

The value written to this register is transferred to the work register in the DMAC core at the beginning of DMA transfer, and the work register value is reloaded at the end of single-operand transfer or DMA transfer. However, when the DMOD[2:0] value in DMMOD of DMACm is 011b (rotate), the work register value is not reloaded and the DMCSA register value remains unchanged retaining the value that was set at the beginning of DMA transfer. When the DRLOD bit in DMCRA of DMACm is set to 1 (the transfer destination address reload function is used), the value of the DMRDA register of DMACm is reloaded at the end of DMA transfer.



Downloaded from Arrow.com.

## 12.2.9 DMA Current Transfer Byte Count Register (DMCBC)

Addresses: DMAC0.DMCBC 0008 2008h, DMAC1.DMCBC 0008 2018h
DMAC2.DMCBC 0008 2028h, DMAC3.DMCBC 0008 2038h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | | | | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b25 to b0 | Number of DMA transfer bytes | 0000000h to 3FFFFFFh | R/W |
| b31 to b26 | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMCBC is used to set the number of DMA transfer bytes.

Do not set the DMCBC register of DMACm during data transfer, but set it while the DMAC is not active or DMA transfer is disabled.

Access the DMCBC register of DMACm with 32 bits.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register. Writing 0000000h to this register makes the transfer byte count 64 Mbytes.

The value written to this register is transferred to the work register in the DMAC core at the beginning of DMA transfer. The work register value decreases by the transfer byte count (1 for 8-bit data size, 2 for 16-bit data size, or 4 for 32-bit data size) each time a single data block is transferred. When the work register value decreases to 0000000h, the DMA transfer ends. The DMCBC register of DMACm is reloaded with the value from the working register on completion of a single-operand transfer or of DMA transfer. However, when the BRLOD bit in DMCRA of DMACm is set to 1 (the transfer byte count reload function is used), the value of the DMRBC register of DMACm is reloaded at the end of DMA transfer.



Downloaded from Arrow.com.

## 12.2.10 DMA Reload Transfer Source Address Register (DMRSA)

Addresses: DMAC0.DMRSA 0008 2200h, DMAC1.DMRSA 0008 2210h
DMAC2.DMRSA 0008 2220h, DMAC3.DMRSA 0008 2230h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b31 to b0 | Set an address value to be reloaded to the DMCSA register of DMACm | 00000000h to FFFFFFFFh (4 Gbytes) | R/W |

DMRSA is used to set an address value to be reloaded to the DMCSA register of DMACm.

Access the DMRSA register of DMACm with 32 bits.

When the SRLOD bit in DMCRA of DMACm is set to 1 (the transfer source address reload function is used), the value of the DMRSA register of DMACm is reloaded to the DMCSA register of DMACm at the end of DMA transfer.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register so that b31 to b0 correspond to A31 to A0.

## 12.2.11 DMA Reload Transfer Destination Address Register (DMRDA)

Addresses: DMAC0.DMRDA 0008 2204h, DMAC1.DMRDA 0008 2214h
DMAC2.DMRDA 0008 2224h, DMAC3.DMRDA 0008 2234h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b31 to b0 | Set an address value to be reloaded to the DMCDA register of DMACm | 00000000h to FFFFFFFFh (4 Gbytes) | R/W |

DMRDA is used to set an address value to be reloaded to the DMCDA register of DMACm.

Access the DMRDA register of DMACm with 32 bits.

When the DRLOD bit in DMCRA of DMACm is set to 1 (the transfer destination address reload function is used), the value of the DMRDA register of DMACm is reloaded to the DMCDA register of DMACm at the end of DMA transfer.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register so that b31 to b0 correspond to A31 to A0.



Downloaded from Arrow.com.

## 12.2.12 DMA Reload Transfer Byte Count Register (DMRBC)

Addresses: DMAC0.DMRBC 0008 2208h, DMAC1.DMRBC 0008 2218h
DMAC2.DMRBC 0008 2228h, DMAC3.DMRBC 0008 2238h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | | | | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

Note: x: Undefined

| Bit | Description | Setting Range | R/W |
|---|---|---|---|
| b25 to b0 | Set the number of DMA transfer bytes to be reloaded to the DMCBC register of DMACm | 0000000h to 3FFFFFFh | R/W |
| b31 to b26 | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMRBC is used to set the number of DMA transfer bytes to be reloaded to the DMCBC register of DMACm.

Access the DMRBC register of DMACm with 32 bits.

When the BRLOD bit in DMCRA of DMACm is set to 1 (the transfer byte count reload function is used), the value of the DMRBC register of DMACm is reloaded to the DMCBC register of DMACm at the end of DMA transfer.

Write a multiple of 2 (for 16-bit data size) or a multiple of 4 (for 32-bit data size) to this register. Writing 0000000h to this register makes the transfer byte count 64 Mbytes.



Downloaded from Arrow.com.

## 12.2.13 DMA Interrupt Control Register (DMICNT)

Address: 0008 250Bh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DINTM0 | DINTM1 | DINTM2 | DINTM3 | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4 | DINTM3 | DMA3 Interrupt Enable | 0: DMAm interrupts are disabled | R/W |
| b5 | DINTM2 | DMA2 Interrupt Enable | 1: DMAm interrupts are enabled | R/W |
| b6 | DINTM1 | DMA1 Interrupt Enable | | R/W |
| b7 | DINTM0 | DMA0 Interrupt Enable | | R/W |

DMICNT is used to enable a DMAm (m = 0 to 3) interrupt request (DMTENDm).

**DINTMn Bits (DMAm Interrupt Enable) (m = 0 to 3)**

Setting the DINTMm bit to 1 enables a DMAm interrupt request (DMTENDm) to occur at the end of DMA transfer of channel m. When the DINTMm bit is set to 0, no DMAm interrupt request (DMTENDm) is generated.



Downloaded from Arrow.com.

## 12.2.14 DMA Start Register (DMSCNT)

Address: 0008 2502h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DMST |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DMST | DMAC Start | 0: DMAC stop<br>1: DMAC start | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DMSCNT is used to start the DMAC.

### DMST Bit (DMAC Start)

When the DMST bit is set to 1, the DMAC is activated.

When the DMST bit is cleared to 0 during DMA transfer in operand transfer mode, DMA transfer of all channels is suspended after the ongoing single-operand transfer is completed. The DMA transfer is restarted by setting this bit to 1 again.

In nonstop transfer mode, DMA transfer is not suspended by clearing the DMST bit to 0 and continues until the DMA transfer is completed.

To make transitions to the module-stop function for the DMAC, all-module clock-stop mode, software-standby mode, or deep software-standby mode, set the DMST bit to 0 (DMAC stop).

See section 8, Power-Down Modes, for further information on the module-stop function for the DMAC, all-module clock-stop mode, software-standby mode, and deep software-standby mode.



Downloaded from Arrow.com.

## 12.2.15 DMA Arbitration Status Register (DMASTS)

Address: 0008 251Bh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DASTS0 | DASTS1 | DASTS2 | DASTS3 | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 0. Do not write to this bit. | R |
| b4 | DASTS3 | Channel 3 Arbitration Status Flag | 0: Data transfer is not in progress | R |
| b5 | DASTS2 | Channel 2 Arbitration Status Flag | 1: Data transfer is in progress | R |
| b6 | DASTS1 | Channel 1 Arbitration Status Flag | (during operand transfer or nonstop transfer) | R |
| b7 | DASTS0 | Channel 0 Arbitration Status Flag | | R |

DMASTS indicates data transfer status of each channel.

**DASTSm Flag (Channel m Arbitration Status Flag) (m = 0 to 3)**

When data transfer (single-operand transfer or nonstop transfer) of channel m is started, the corresponding DASTSm flag is set to 1 and is cleared to 0 when the data transfer is completed.

[Setting conditions]

- In the case of operand transfer, the start of transfer for an operand
- In the case of non-stop transfer, the start of DMA transfer

[Clearing condition]

- The completion of transfer for a single operand or of DMA transfer



Downloaded from Arrow.com.

## 12.2.16 DMA Transfer End Detect Register (DMEDET)

Address: 0008 2517h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DEDET0 | DEDET1 | DEDET2 | DEDET3 | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4 | DEDET3 | Channel 3 DMA Transfer End Detect Flag | Read:<br>0: Not detected<br>1: Detected<br><br>Write:<br>0: Disabled<br>1: The DEDETm flag (m = 0 to 3) is cleared to 0 | R/W |
| b5 | DEDET2 | Channel 2 DMA Transfer End Detect Flag | | R/W |
| b6 | DEDET1 | Channel 1 DMA Transfer End Detect Flag | | R/W |
| b7 | DEDET0 | Channel 0 DMA Transfer End Detect Flag | | R/W |

DMEDET indicates the DMA transfer end of each channel.

**DEDETm Flags (Channel m DMA Transfer End Detect Flags) (m = 0 to 3)**

Upon completion of DMA transfer of channel m, the DEDETm flag is set to 1. Once the DEDETm flag is set to 1, it is not cleared to 0 automatically. To clear the DEDETm flag to 0, write 1 to the flag by the program. This written value of 1 is not retained. Writing 0 to these bits has no effect.

To use a DMAm interrupt (DMTENDm), write 1 to the DEDETm flag of a channel in which a DMA interrupt request is generated in the interrupt routine.

[Setting condition]

- Completion of DMA transfer

[Clearing condition]

- Writing of a "1" to the relevant DEDETm bit

Do not use any bit manipulation instruction such as BSET instruction to clear the DEDETm flags. To clear the DEDETm flags, set only the bit of a channel to be cleared to 1 using the MOV instruction, and write to the DMEDET register.



Downloaded from Arrow.com.

## 12.3 Operation

### 12.3.1 Bus Mastership Release Timing

The DMAC must release bus mastership for an interval of at least one cycle per single operation of data reading and writing. The bus is thus accessible by another master (CPU or DTC) during this interval.

Furthermore, access by the CPU (except with the DMAC as the target) is possible during access by the DMAC. However, when the external bus is set as the source or destination for transfer by the DMAC, access over the internal bus by the CPU and DTC may become impossible due to the relation between the clock and the timing of access. In such cases, divided up the data for transfer by the DMAC into smaller units and handle transfer in these units so that the CPU and DTC become able to accept access requests when transfer-completed interrupts are conveyed to the CPU and DTC. For details, refer to section 11, Bus.

Figure 12.2 shows an example of how bus mastership passes between the DMAC and other bus masters.

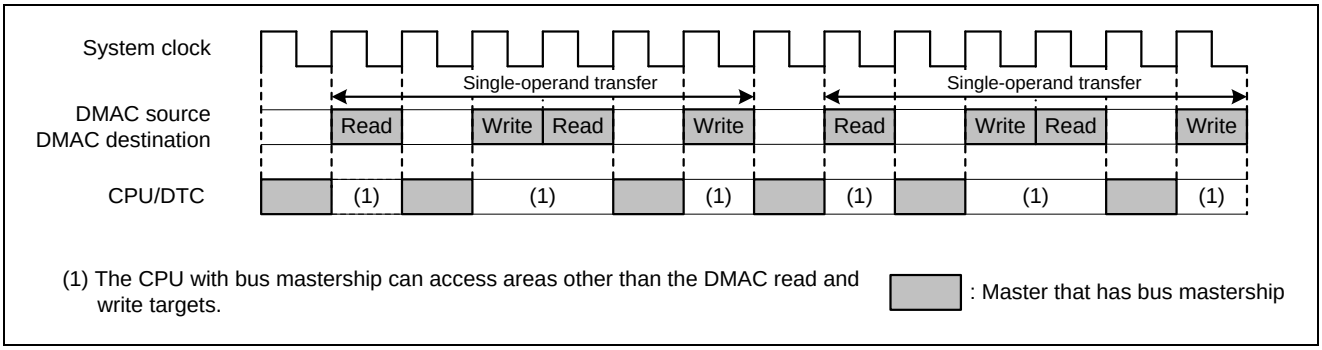



**Figure 12.2 Example of how Bus Mastership Passes between the DMAC and Other Bus Masters**



Downloaded from Arrow.com.

### 12.3.2 Transfer System

The DMA transfer system includes operand transfer and nonstop transfer.

The operand transfer system includes single-operand transfer and consecutive-operand transfer. The single-operand transfer allows transfer of a single operand per DMA transfer request, and the consecutive-operand transfer allows transfer in operand units upon a DMA transfer request until the DMA transfer ends.

The nonstop transfer allows continuous data transfer upon a DMA transfer request until the DMA transfer ends.

Regardless of transfer system, data of the byte count specified in the DMACm.DMCBC register is transferred during a single DMA transfer. When the DMCBC register value of DMACm is decreased to 0000000h, the DMA transfer is completed.

Table 12.5 lists transfer systems.

**Table 12.5 Transfer Systems**

| DSEL[1:0] Bits in DMCRA of DMACm | Transfer System | Byte Count Transferred per DMA Transfer Request |
|---|---|---|
| 00b<br>(Single-operand transfer) | • Single-operand data is transferred during DMA transfer.<br>• Single-operand data is transferred at each DMA transfer request until DMA transfer ends.<br>• Channel arbitration is made at the end of single-operand transfer.<br>• DMA transfer request is necessary each time single-operand transfer is completed. | Byte count corresponding to single-operand data count × bit length |
| 01b<br>(Consecutive-operand transfer) | • Single-operand data is transferred during DMA transfer.<br>• Data is transferred in operand units until DMA transfer ends.<br>• Channel arbitration is made each time single-operand transfer is completed.<br>• A DMA transfer request is generated first only once. | Byte count specified by the DMCBC register of DMACm |
| 11b<br>(Nonstop transfer) | • Data is transferred continuously during DMA transfer.<br>• Data is transferred continuously until DMA transfer ends.<br>• Channel arbitration is made at the end of DMA transfer.<br>• A DMA transfer request is generated first only once. | Byte count specified by the DMCBC register of DMACm |

In the operand transfer system, when there is a DMA transfer request of higher-priority channel in the channel arbitration that is made at the end of single-operand transfer, DMA transfer of the higher-priority channel is performed. When there is no such DMA transfer request, the following operand is transferred continuously. However, when there is no DMA transfer request in single-operand transfer mode, the following operand is not transferred.

In the nonstop transfer system, data is transferred continuously from the start to the end of DMA transfer. Therefore, even if a DMA transfer request of higher-priority channel is generated during DMA transfer, the request is not accepted.

Figure 12.3 shows examples of DMA transfer in each transfer system.



Downloaded from Arrow.com.

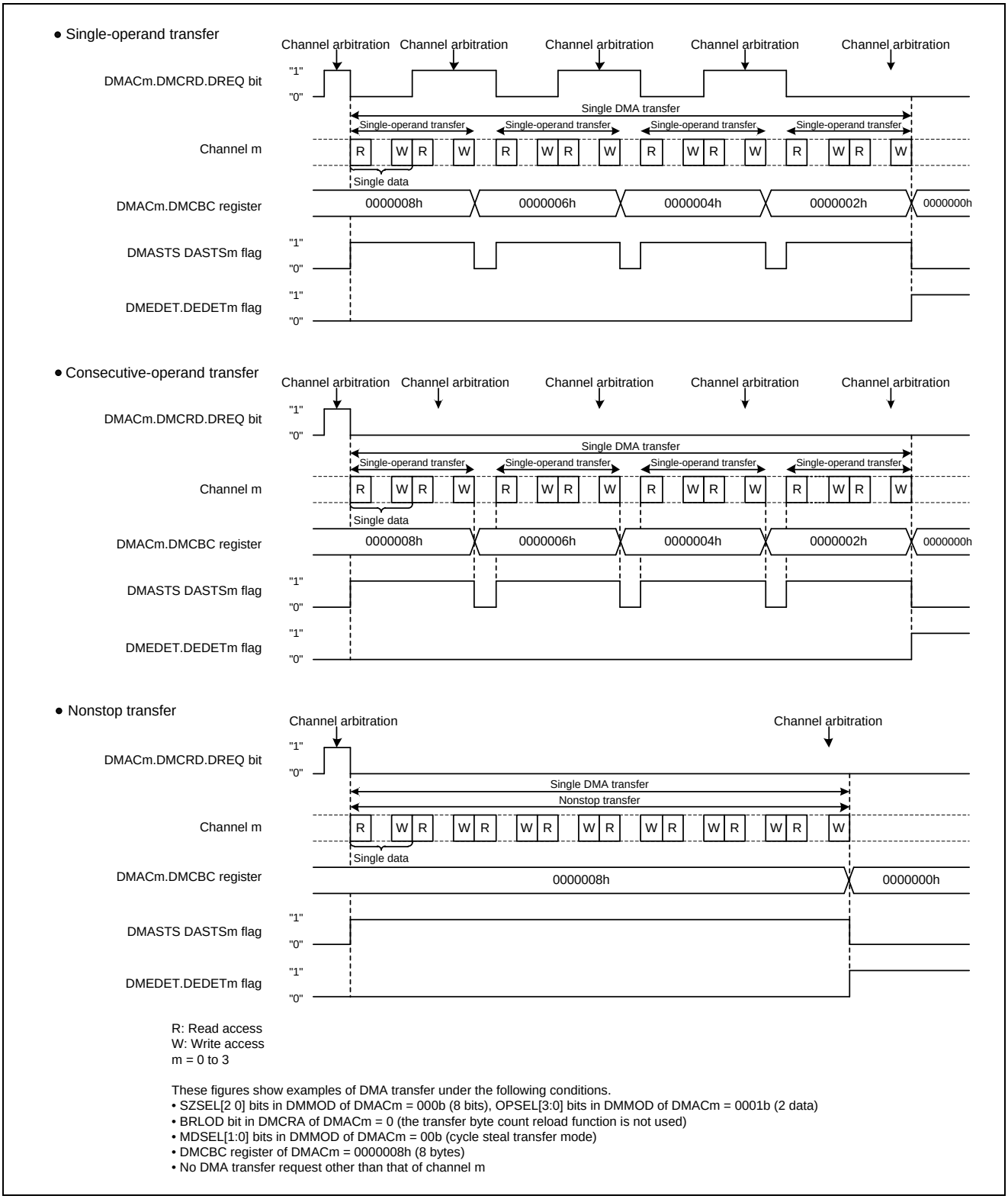



**Figure 12.3 Examples of DMA Transfer in Each Transfer System**



Downloaded from Arrow.com.

## 12.3.3 Activating the DMAC

Figure 12.4 shows the register setting procedure.

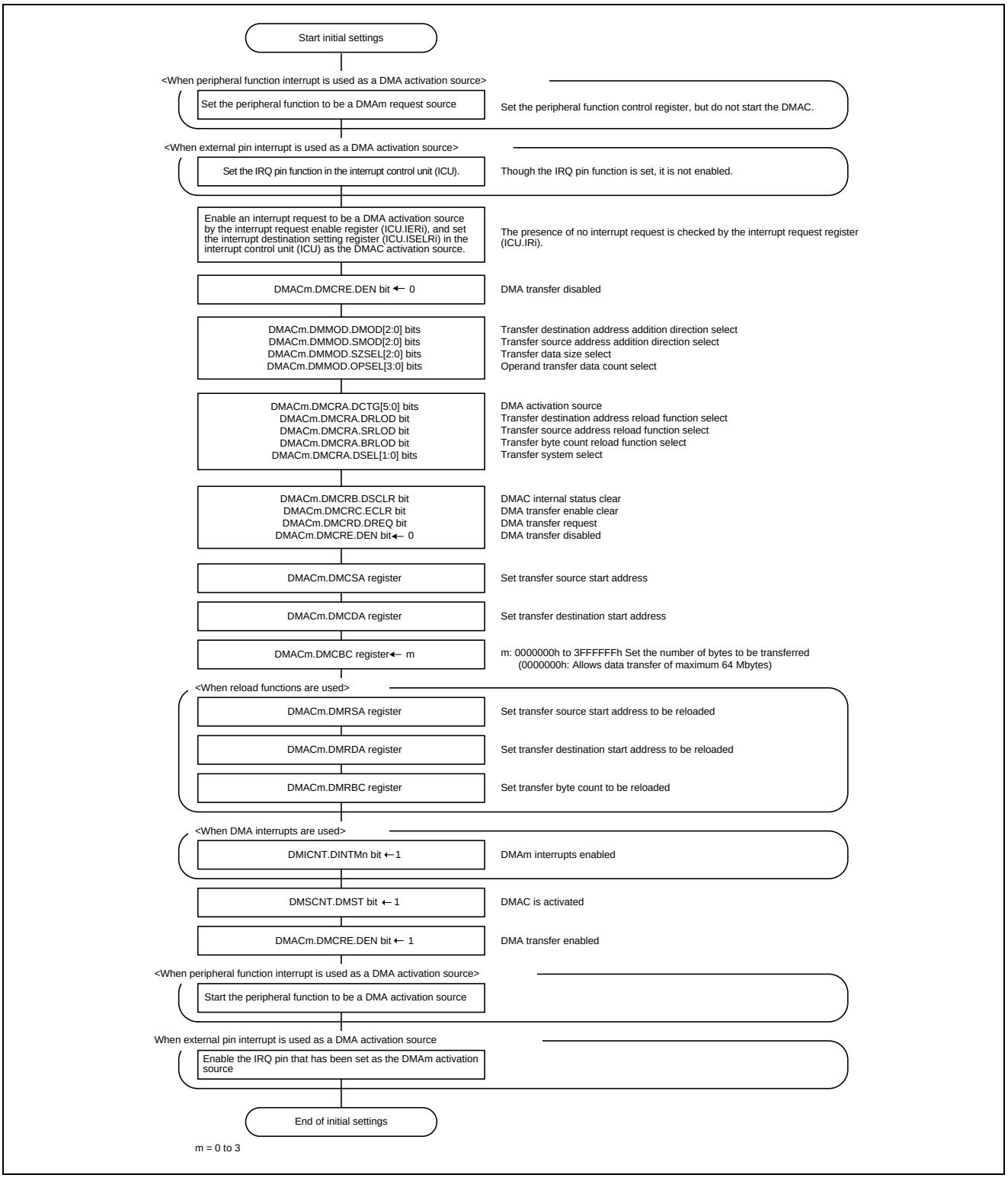



**Figure 12.4 Register Setting Procedure**



Downloaded from Arrow.com.

### 12.3.4 Starting DMA Transfer

Setting the DEN bit in DMCRE of DMACm to 1 (DMA transfer enabled) and setting the DMST bit in DMSCNT to 1 (DMAC start) enable DMA transfer of channel m (m = 0 to 3).

When DMA transfer requests are generated, channel arbitration is made where a DMA transfer request of higher-priority channel is accepted and DMA transfer of the channel starts. When a DMA transfer request is accepted and DMA transfer starts, the DASTSm flag in DMASTS is set to 1 (data transfer is in progress).

### 12.3.5 Ending DMA Transfer

When the DMCBC register value of DMACm is decreased to 0000000h, DMA transfer of channel m (m = 0 to 3) ends and the DMAC performs the following processing.

- The DEDETm flag in DMEDET is set to 1 (DMA transfer end detected).
- When the DINTMm bit in DMICNT is 1 (interrupts enabled), a DMAm interrupt request (DMTENDm) request occurs.
- When the value of the ECLR bit in DMCRC for DMACm is "1" causing the DEN bit in DMCRE for DMACm to be cleared to "0" on completion of DMA transfer, subsequent DMA transfer on channel m does not proceed once the value of the DEN bit has become "1" (disabling DMA transfer).
- When the reload functions are used, reload register values are reloaded to the current registers respectively.



Downloaded from Arrow.com.

### 12.3.6 Suspending, Restarting, and Canceling DMA Transfer

#### (1) Suspending DMA transfer

DMA transfer by operand transfer can be suspended by clearing the DMST (DMAC stop) bit in DMSCNT or the DEN bit in DMCRE for DMACm to 0 (disabling DMA transfer on that channel).

Suspension applies to all channels when the DMST bit is 0 or the corresponding channel n when DEN bit is 0, and becomes effective on completion of the transfer for the operand for which transfer was in progress.

Nonstop DMA transfer is not suspended even when the DMST or DEN bit is cleared to 0; instead, DMA transfer continues until it is completed.

In advance of transitions of the DMAC to the module-stop state or of the overall device to all-module clock stop mode, software standby mode, or deep software standby mode, clear the DMST bit to 0 (stopping the DMAC).

#### (2) Restarting DMA transfer

Suspended DMA transfer on a channel is restarted by setting the DMST bit in DMSCNT or the DEN bit in DMCRE of the given DMACm to 1.

After returning the DMAC from the module-stop state or the overall device from all-module clock stop mode, software standby mode, or deep software standby mode, setting the DMST bit to 1 restarts DMA transfer on channels where transfer had been suspended.

#### (3) Canceling DMA transfer

If the DSCLR bit in DMCRB of DMACm is set to 1 with DMA transfer of the channel suspended, the DMA transfer is canceled and the internal state of the DMAC is initialized. However, only the transfer state of the DMAC's internal circuits is initialized at this time; that is, all registers are not initialized.



Downloaded from Arrow.com.

### 12.3.7 DMA Activation Source

Interrupt signals from peripheral functions and interrupt signals on external pins for which the appropriate software-trigger and interrupt control unit (ICU) settings have been made are selectable as DMA activation sources.

#### 12.3.7.1 Software Trigger

When software trigger is selected as a DMA activation source and the DREQ bit in DMCRD of DMACm is set to 1 (a DMA transfer request is generated) by the program, a DMA transfer request is generated. The DREQ bit can be set to 1 independently of the DMA transfer status, but be sure to clear the DREQ bit to 0 (no DMA transfer request is generated) while the DMAC is not active or DMA transfer is disabled (except during DMA transfer).

#### 12.3.7.2 Interrupt Signals on External Pins and Peripheral Function Interrupts

When the corresponding bit of the relevant interrupt enabling register in the ICU (IERi; i = 02h to 1Fh) has been set to 1 (enabling an interrupt signal on an external pin or an interrupt signal from a peripheral module), the interrupt destination setting bits (ISEL[1:0]) in the interrupt destination setting register (ISELR) of the interrupt control unit (ICU) have been set to select the DMAC as the destination, and the DTCG[5:0] bits in DMCRA of DMACm have been set so that the corresponding source of interrupt requests acts as a source of DMA transfer requests, generation of the conditions for the selected request leads to the generation of a DMA transfer request.

When a DMA transfer request is detected, the DREQ bit in DMCRD of DMACm is set to "1" to indicate the presence of the DMA transfer request, and the value of the DREQ remains unchanged even if the input level subsequently changes until a "0" (indicating that there is not DMA transfer request) is written to this bit by the program or until the DMA transfer request is accepted.

Since the value of the DREQ bit that indicates the DMA transfer request is retained, a DMA transfer request that is issued while the value of the DREQ bit is still "1" will be ignored.

Figure 12.5 shows the timing of the DREQ bit.

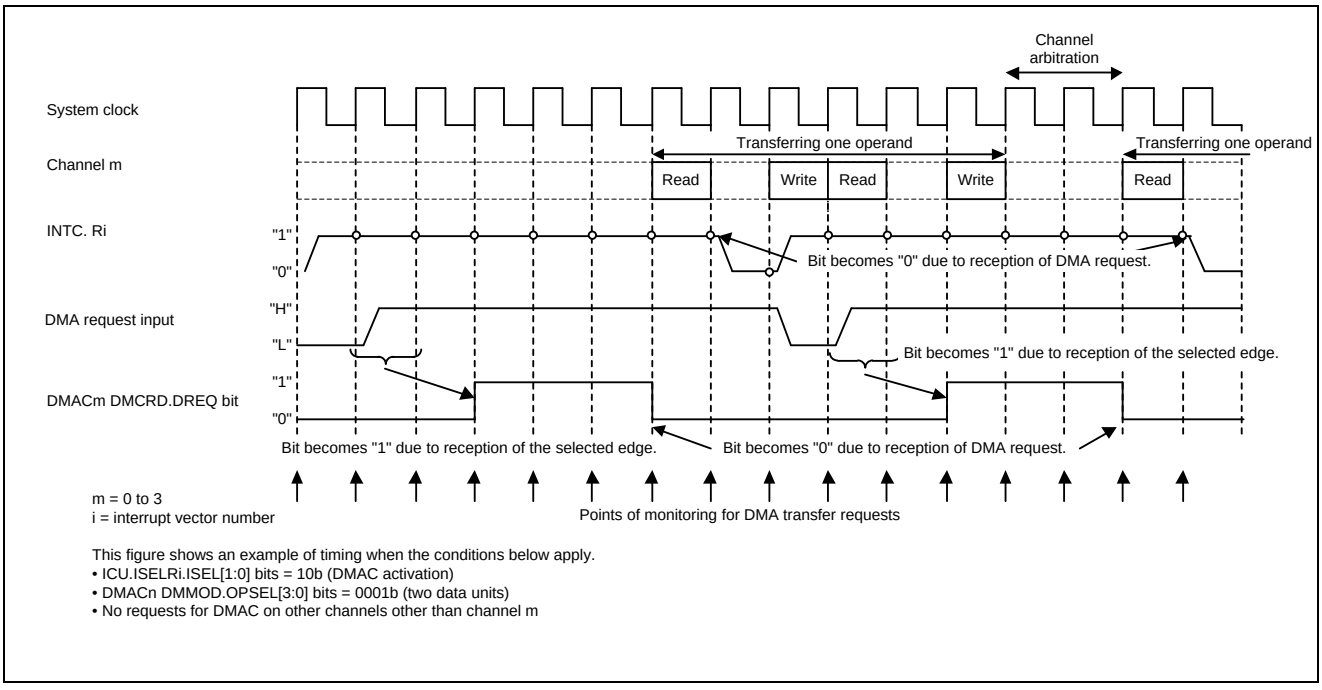



**Figure 12.5 Timing of DMACm.DMCRD.DREQ Bit Transitions**



Downloaded from Arrow.com.

### 12.3.8 Channel Arbitration

When multiple DMA transfer requests are present, the DMAC determines the priority of channels that have DMA transfer requests. The channel priority is fixed as channel 0 > channel 1 > channel 2 > channel 3 (channel 0: highest).

When a DMA transfer request is generated during data transfer, channel arbitration is started at the beginning of final data write access. Therefore, when another DMA transfer request of higher-priority channel is generated during the data transfer, DMA transfer of the higher-priority channel starts after the current data transfer is completed.

### 12.3.9 Reload Function

The reload function reloads values of the reload registers (DMRSA, DMRDA, and DMRBC of DMACm) to the current registers (DMCSA, DMCDA, and DMCBC of DMACm) respectively at the end of DMA transfer, and is available for reloading transfer source addresses, transfer destination addresses, and transfer byte counts. The reload function allows continuous transfer of separately allocated blocks. That is, multiple transfer blocks with different transfer area and/or data size can be transferred continuously in the same channel. Furthermore, by writing a value to a reload register before the data transfer ends, the next transfer can be prepared without affecting the current register engaged in DMA transfer.

To use the reload function, write data to the corresponding reload registers and current registers. Set reload registers to be used before the final data transfer starts. If they are set after the final data transfer starts, the settings may not be reflected in the reload operation after the DMA transfer ends.

When the reload function is not used, set the ECLR bit in DMCRC of DMACm to 1 (the DEN bit is cleared to 0 at the end of DMA transfer) to clear the DEN bit in DMCRE of DMACm.

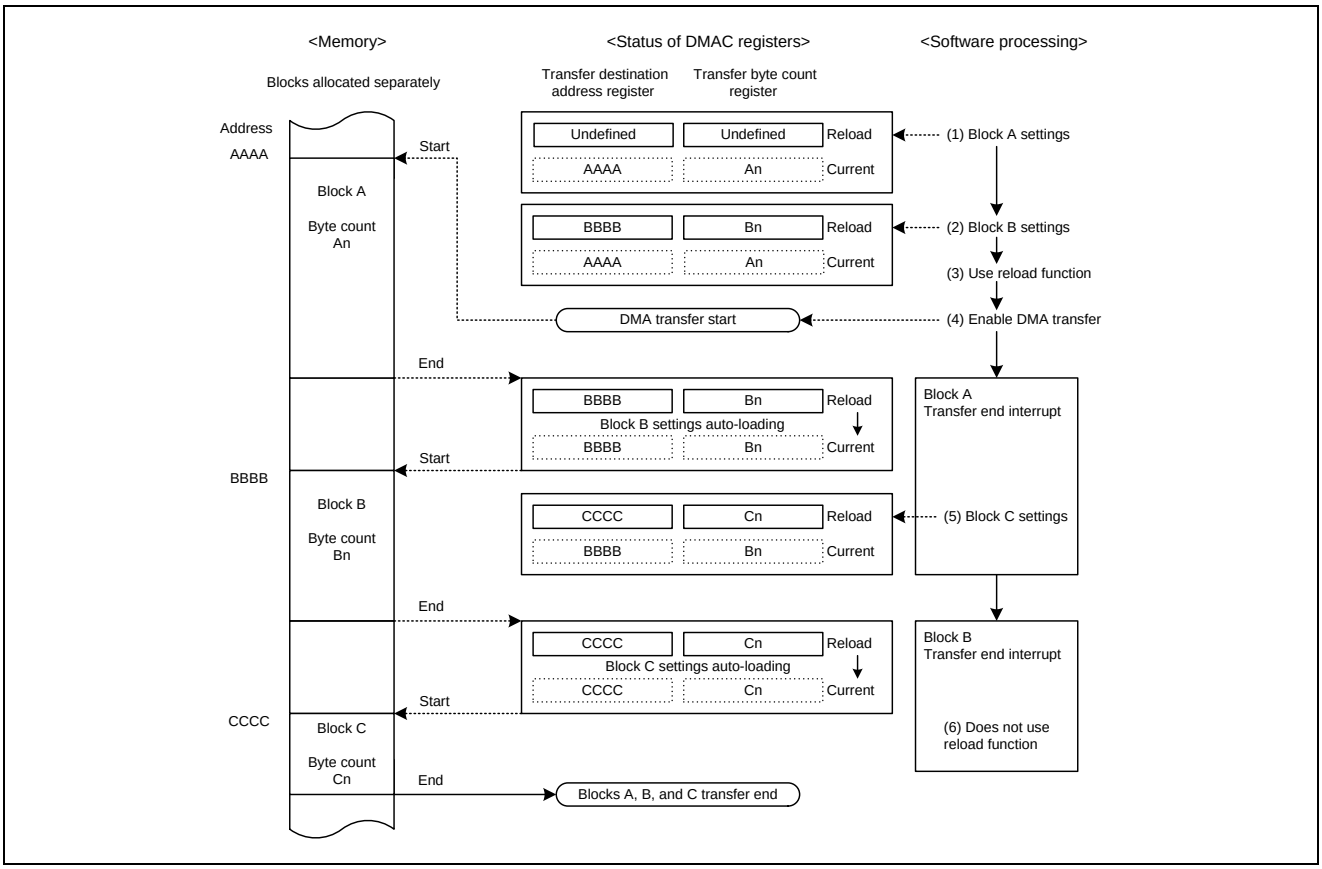



**Figure 12.6 Example of DMA Transfer Using the Reload Function**



Downloaded from Arrow.com.

### 12.3.10 Rotate

When "rotate" is selected with the DMOD[2:0] and SMOD[2:0] bits in DMMOD of DMACm, transfer destination and source addresses are automatically increased during data transfer. The values written at the beginning of DMA transfer are reloaded to the address registers when single-operand transfer is completed.

Figure 12.7 shows an example of "rotate" transfer.

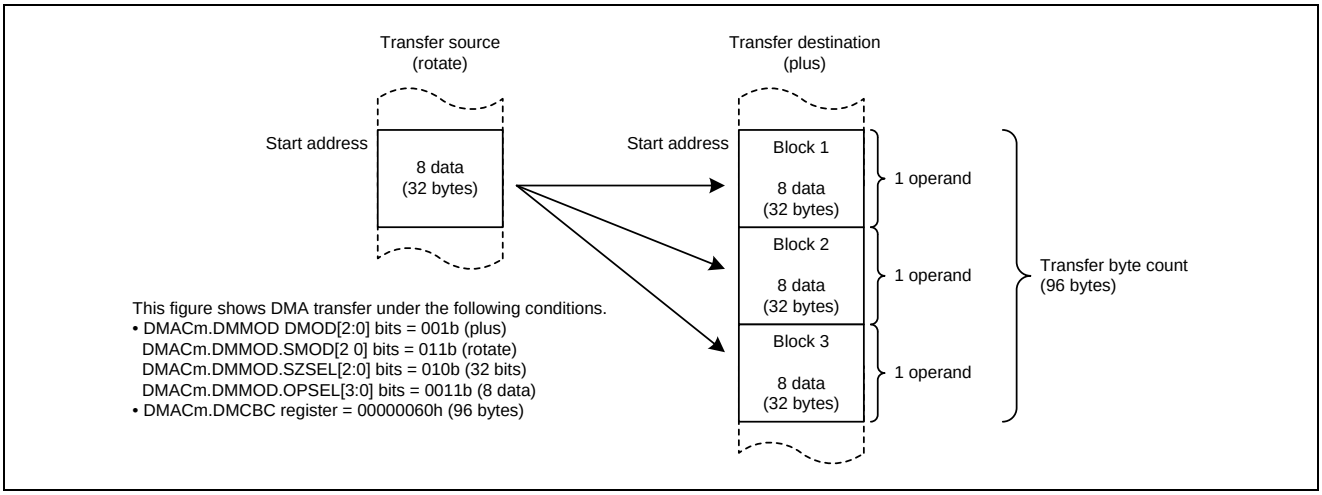



**Figure 12.7 Example of "Rotate" Transfer**

## 12.4 Interrupts

When the DINTMm bit (m = 0 to 3) in DMICNT is set to 1 (DMAm interrupts enabled), a DMAm interrupt request (DMTENDm) is generated upon completion of channel-m DMA transfer. Figure 12.8 shows the schematic logic diagram of interrupt outputs.

To use a DMAm interrupt, write 1 to the DEDETm flag in DMEDET of a channel in which a DMA interrupt request is generated in the interrupt routine. Setting the DINTMm bit in DMICNT to 1 from 0 while the DEDETm flag in DMEDET is set to 1 generates a DMAm interrupt request (DMTENDm) regardless of the DMAC transfer status.

The interrupt control unit (ICU) can be set to select a DMAm interrupt as a source for activation of the data transfer controller (DTC). For details, refer to section 10, Interrupt Control Unit (ICU).

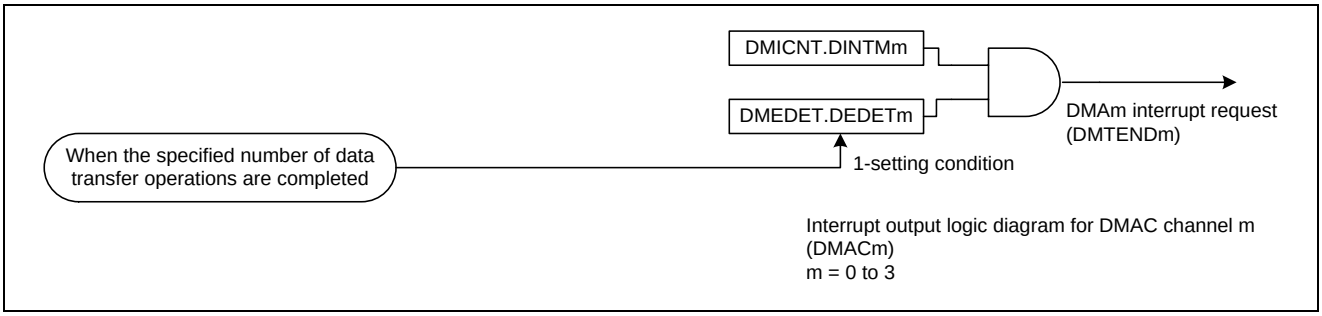



**Figure 12.8 Schematic Logic Diagram of Interrupt Outputs**



Downloaded from Arrow.com.

## 12.5 Low-Power Consumption

If the DMAC is to be placed in any low power-consumption state (module-stop state, all-module clock-stop mode, software-standby mode, or deep software-standby mode), DMAC transfer in progress when the request for transition to the low-power state was accepted must be suspended.

Specifically, clear the DMST bit in DMSCNT to 0 (stopping the DMAC) in advance of transitions of the DMAC to the module-stop state or of the overall device to all-module clock stop mode, software standby mode, or deep software standby mode.

### (1) Module Stop

After "0" has been written to the DMSCNT.DMST (DMAC stop) bit, writing a "1" to the MSTPA28 bit (transition to the module-stop state) in MSTPCRA places the DMAC in the module-stop state. If DMA transfer is in progress at the time a "1" is written to the MSTPA28 bit, the transition to the module-stop state proceeds after DMAC transfer has been suspended. Furthermore, the internal registers of the DMAC become inaccessible at the time a "1" is written to the MSTPA28 bit, regardless of the state of DMA transfer.

Writing a "0" to the MSTPA28 bit releases the DMAC from the module-stop state. The registers of the DMAC become accessible again once the DMAC is supplied with a clock signal.

### (2) All-Module Clock-Stop Mode

After writing "0" to the DMSCNT.DMST (DMAC stop) bit, and then writing a "1" to the ACSE bit (permitting all-module clock-stop mode) in MSTPCRA, writing "1" to all bits in MSTPCRA and MSTPCRB, including the MSTPA28 bit (module-stop bit for the DMAC), and confirming that "1" has been written to all bits of the MSTPCRA and MSTPCRB registers, executing a WAIT instruction causes a transition to the all-module clock-stop mode. However, if DMA transfer is in progress at the time the WAIT instruction is executed, the transition to all-module clock stop mode becomes possible after the suspension of DMA transfer.

Release from all-module clock stop mode is triggered by an external interrupt (the NMI or any of IRQ0 to IRQ15), a reset by the signal on the RES# pin, or an internal interrupt (from an 8-bit timer or the watchdog timer).

### (3) Software Standby and Deep Software Standby Modes

After writing "0" to the DMSCNT.DMST (DMAC stop) bit, and then writing a "1" to the SBYCR.SSBY bit (selecting a transition to software standby mode following the execution of a WAIT instruction) and a "0" to the DPSBYCR.DPSBY bit (so that the transition is not to deep software standby mode following the execution of a WAIT instruction), executing a WAIT instruction places the chip in software standby mode.

However, if DMA transfer is in progress at the time the WAIT instruction is executed, the transition to software-standby mode becomes possible after the suspension of DMA transfer.

Release from software-standby mode is triggered by an external interrupt (the NMI or any of IRQ0 to IRQ15), a reset by the signal on the RES# pin, or an internal interrupt (from an eight-bit timer or the watchdog timer).

If the setting of the DPSBY bit is "1" (selecting a transition to deep software standby mode after execution of a WAIT instruction), the transition is to deep software standby mode.



Downloaded from Arrow.com.

## 12.6 Usage Notes

### 12.6.1 Register Settings

(1) When setting the bits or registers below, set them when the DASTSm flag (m = 0 to 3) in DMASTS is 0 (data transfer is not in progress) and the DEN bit in DMCRE of DMACm is 0 (DMA transfer disabled) or when the DMST bit in DMSCNT is 0 (DMAC stop) for the target channel.

Registers: DMMOD, DMCRB, DMCRC, DMCDA, and DMCBC of DMACm
Bits: DSEL[1:0] bits and DCTG[5:0] bits in the DMCRA register of DMACm

Write 0 to the DREQ bit in DMCRD of DMACm (However, the DREQ bit can be set to 1 independently of the DMA transfer status.)

(2) Access the following registers with 32 bits.
DMMOD, DMCSA, DMCDA, DMCBC, DMRSA, and DMRDA of DMACm

(3) Modify the ECLR bit in DMCRC of DMACm while the DASTSm flag (m = 0 to 3) in DMASTS is 0 (data transfer is not in progress). When the reload function is not used, set the ECLR bit to 1 (the DEN bit is cleared to 0 at the end of DMA transfer) to clear the DEN bit in DMCRE of DMACm.

(4) When the DCTG[5:0] in DMCRA of DMACm is set to 1, be sure to clear the DREQ bit in DMCRD of DMACm and then set the DMST bit in DMSCNT to 1 (DMAC start) and the DEN bit in DMCRE of DMACm to 1 (DMA transfer enabled) for the selected channel.

(5) The DREQ bit in DMCRD of DMACm varies with the presence or absence of DMA transfer request regardless of the setting of the DMST bit in DMSCNT and the DEN bit in DMCRE of DMACm. Unless software trigger is selected as a DMA activation source, do not write 1 to the DREQ bit by the program.

(6) Set each address register and transfer byte count register to an aligned value according to data size. Table 12.6 lists alignment and the lower 2 bits of each register according to data size.

(7) Do not use any bit manipulation instruction such as BSET instruction to clear the DEDETm flags in DMEDET. To clear the DEDETm flags, set only the bit of a channel to be cleared to 1 using the MOV instruction, and write to the DMEDET register.

**Table 12.6 Alignment and the Lower 2 Bits of Each Register According to Data Size**

| SZSEL[2:0] Bits in DMMOD of DMACm | Alignment | Address Registers | | Transfer Byte Count Registers | |
|---|---|---|---|---|---|
| | | b1 | b0 | b1 | b0 |
| 000b (8 bits) | Integer multiple | — | — | — | — |
| 001b (16 bits) | Multiple of 2 | — | 0 | — | 0 |
| 010b (32 bits) | Multiple of 4 | 0 | 0 | 0 | 0 |

[Legend] —: Either 0 or 1

### 12.6.2 DMA Transfer to External Devices

When DMA transfer to an external device is performed, the DASTSm flag (m = 0 to 3) in DMASTS is cleared to 0 (data transfer is not in progress) before the external bus access ends after the final data write is started in some cases.



Downloaded from Arrow.com.

# 13. Data Transfer Controller (DTC)

The RX610 Group incorporates a data transfer controller (DTC).

The DTC is activated by an interrupt request and controls data transfer.

## 13.1 Overview

Table 13.1 lists the specifications of the DTC, and figure 13.1 shows a block diagram of the DTC.

**Table 13.1 DTC Specifications**

| Item | Description |
|---|---|
| Transfer mode | • Normal transfer mode<br>• Repeat transfer mode<br>• Block transfer mode |
| Transfer of arbitrary channels | • Data of multiple channels can be transferred on a single startup source (chain transfer).<br>• Execution of chain transfer after data transfer can be set. |
| Short-address mode/ Full-address mode | • Transfer data is packed in 3 longwords (short-address mode) or in 4 longwords (full-address mode).<br>• In short-address mode, transfer source address and transfer destination address can be specified with 24 bits, and a 16-Mbyte address space can be directly specified for repeat transfer.<br>• In full-address mode, transfer source address and transfer destination address can be specified with 32 bits, and a 4-Gbyte address space can be directly specified for normal transfer. |
| Data transfer units | • Byte, word, or longword<br>Transfer of units that are not aligned with the corresponding boundary is possible. However, see selection 11, Buses, for details on the allowable range for the transfer of a word or longword unit for which an odd address is specified, or of a longword unit with an address of the form 4n + 2 (n = 0, 1, 2 …). |
| Interrupt source | • An interrupt request is output to the CPU after a single data transfer.<br>• An interrupt request is output to the CPU after data transfer of specified volume. |



Downloaded from Arrow.com.

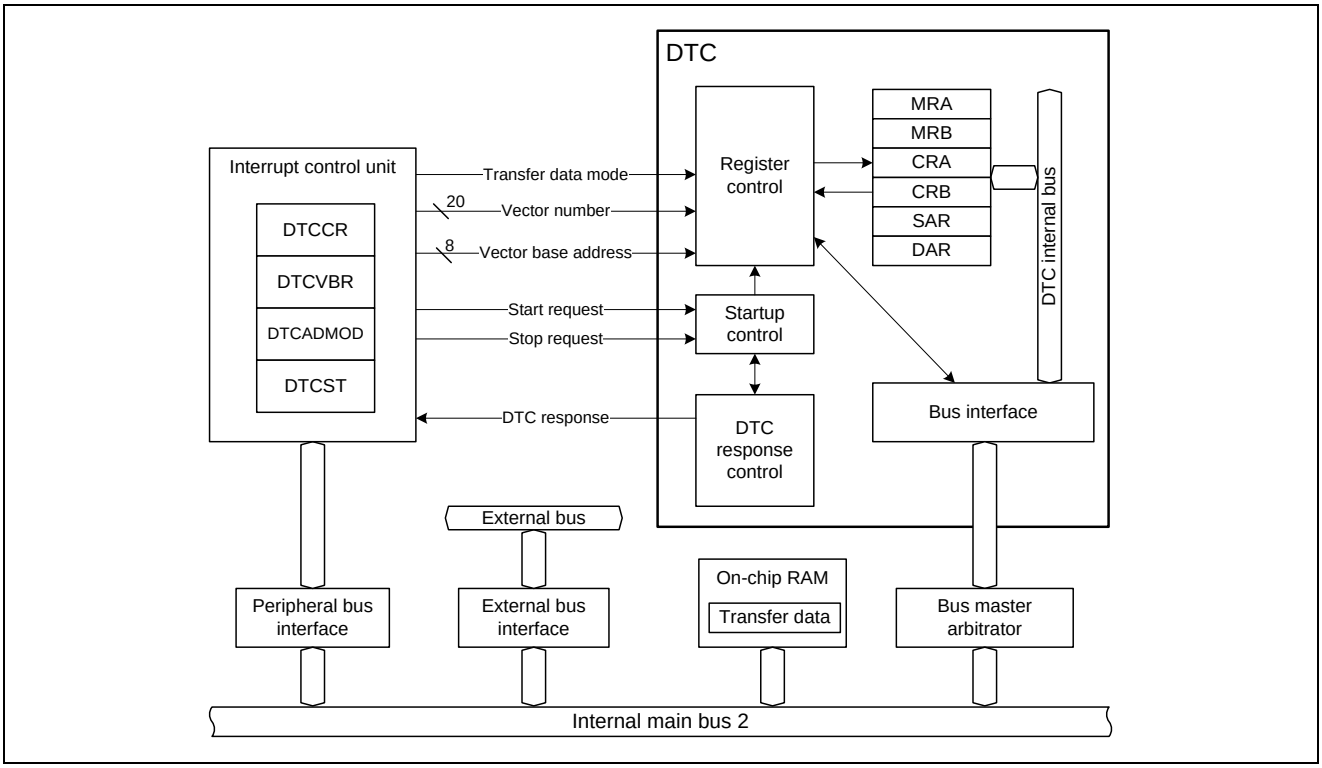



**Figure 13.1 Block Diagram of the DTC**



Downloaded from Arrow.com.

## 13.2 Register Descriptions

Table 13.2 lists the registers of the DTC.

Registers MRA, MRB, SAR, DAR, CRA, and CRB cannot be accessed directly from the CPU. They are located in the data area as transfer data.

When a DTC activation request is generated, the transfer data start address is read according to the vector address determined for each startup source, and arbitrary transfer data is transferred in the DTC. Upon completion of the data transfer, the values of these registers are written back.

**Table 13.2 Registers of the DTC**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| DTC mode register A | MRA | xxh | — | 8 bits |
| DTC mode register B | MRB | xxh | — | 8 bits |
| DTC source address register | SAR | xxxxxxxxh | — | 32 bits |
| DTC destination address register | DAR | xxxxxxxxh | — | 32 bits |
| DTC transfer count register A | CRA | xxxxh | — | 16 bits |
| DTC transfer count register B | CRB | xxxxh | — | 16 bits |
| DTC control register | DTCCR | 00h | 0008 7400h | 8 bits |
| DTC vector base register | DTCVBR | 00000000h | 0008 7404h | 32 bits |
| DTC address mode register | DTCADMOD | 00h | 0008 7408h | 8 bits |
| DTC module start register | DTCST | 00h | 0008 740Ch | 8 bits |

Note: To activate the DTC, a setting of the ISELRi.ISEL[1:0] and IERi.IENj bits in the interrupt control unit (ICU) is required. For details, refer to section 10, Interrupt Control Unit (ICU).

[Legend]

x: Undefined value



Downloaded from Arrow.com.

### 13.2.1 DTC Mode Register A (MRA)

Address (inaccessible directly from the CPU)

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | MD[1:0] | | SZ[1:0] | | SM[1:0] | | – | – |
| Value after reset: | x | x | x | x | x | x | x | x |

[Legend] x: Undefined

| Bit | Symbol | Bit | Symbol | R/W |
|---|---|---|---|---|
| b1, b0 | — | Reserved | The read data is undefined. The write value should be 0. | — |
| b3, b2 | SM[1:0] | SAR Transfer Source Address Addressing Mode | b3 b2<br>0 0: SAR address is fixed (Write-back to SAR is skipped)<br>0 1: SAR address is fixed (Write-back to SAR is skipped)<br>1 0: SAR value is incremented after data transfer (+1 when SZ[1:0] bits = 00b, +2 when SZ[1:0] bits = 01b, +4 when SZ[1:0] bits = 10b)<br>1 1: SAR value is decremented after data transfer (-1 when SZ[1:0] bits = 00b, -2 when SZ[1:0] bits = 01b, -4 when SZ[1:0] bits = 10b) | — |
| b5, b4 | SZ[1:0] | DTC Data Transfer Size | b5 b4<br>0 0: Byte transfer<br>0 1: Word transfer<br>1 0: Longword transfer<br>1 1: Setting prohibited | — |
| b7, b6 | MD[1:0] | DTC Mode | b7 b6<br>0 0: Normal transfer mode<br>0 1: Repeat transfer mode<br>1 0: Block transfer mode<br>1 1: Setting prohibited | — |

MRA is used to select the operating mode of the DTC.

MRA cannot be accessed directly from the CPU.

#### SM[1:0] Bits (SAR Transfer Source Address Addressing Mode)

These bits specify the SAR operation after data transfer.

#### SZ[1:0] Bits (DTC Data Transfer Size)

These bits specify the transfer data size.

#### MD[1:0] Bits (DTC Mode)

These bits specify the transfer mode of the DTC.



Downloaded from Arrow.com.

## 13.2.2 DTC Mode Register B (MRB)

Address (inaccessible directly from the CPU)

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CHNE | CHNS | DISEL | DTS | DM[1:0] | | – | – |
| Value after reset: | x | x | x | x | x | x | x | x |

[Legend] x: Undefined

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | — | Reserved | The read data is undefined. The write value should be 0. | — |
| b3, b2 | DM[1:0] | DAR Transfer Destination Address Addressing Mode | b3 b2<br>0 0: DAR address is fixed (Write-back to DAR is skipped)<br>0 1: DAR address is fixed (Write-back to DAR is skipped)<br>1 0: DAR value is incremented after data transfer (+1 when SZ[1:0] bits in MRA = 00b, +2 when SZ[1:0] bits = 01b, +4 when SZ[1:0] bits = 10b)<br>1 1: DAR value is decremented after data transfer (-1 when SZ[1:0] bits in MRA = 00b, -2 when SZ[1:0] bits = 01b, -4 when SZ[1:0] bits = 10b) | — |
| b4 | DTS | DTC Transfer Mode Select | 0: Destination side is repeat area or block area<br>1: Source side is repeat area or block area | — |
| b5 | DISEL | DTC Interrupt Select | 0: An interrupt request to the CPU is generated when specified data transfer is completed<br>1: An interrupt request to the CPU is generated each time DTC data transfer is performed | — |
| b6 | CHNS | DTC Chain Transfer Select | 0: Chain transfer is performed continuously<br>1: Chain transfer is performed only when the transfer counter is 0 | — |
| b7 | CHNE | DTC Chain Transfer Enable | 0: Chain transfer is disabled<br>1: Chain transfer is enabled | — |

MRB is used to select the operating mode of the DTC.

MRB cannot be accessed directly from the CPU.

### DM[1:0] Bits (DAR Transfer Destination Address Addressing Mode)

These bits specify the DAR operation after data transfer.

### DTS Bit (DTC Transfer Mode Select)

The DTS bit specifies the side (transfer source or destination) to be a repeat area or block area in repeat transfer mode or block transfer mode.

### DISEL Bit (DTC Interrupt Select)

The DISEL bit specifies whether to generate an interrupt request to the CPU each time DTC data transfer is performed or when specified data transfer is completed.



Downloaded from Arrow.com.

**CHNS Bit (DTC Chain Transfer Select)**

The CHNS bit selects the chain transfer condition.

When the next transfer is chain transfer, completion of specified transfer count is not checked and the startup source flag is not cleared. Moreover, an interrupt request to the CPU is not generated.

**CHNE Bit (DTC Chain Transfer Enable)**

The CHNE bit enables or disables chain transfer.

The chain transfer condition is selected by the CHNS bit.

For details of chain transfer, see section 13.4.6, Chain Transfer.

## 13.2.3 DTC Source Address register (SAR)

Address (inaccess ble directly from the CPU)

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

[Legend] x: Undefined

SAR is used to set the transfer source start address.

In full-address mode, 32 bits are valid.

In short-address mode, lower 24 bits are valid and upper 8 bits (b31 to b24) are ignored. The address of this register is extended by the value specified by b23.

SAR cannot be accessed directly from the CPU.

## 13.2.4 DTC Destination Address Register (DAR)

Address (inaccessible directly from the CPU)

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

[Legend] x: Undefined

DAR is used to set the transfer destination start address.

In full-address mode, 32 bits are valid.

In short-address mode, lower 24 bits are valid and upper 8 bits (b31 to b24) are ignored. The address of this register is extended by the value specified by b23.

DAR cannot be accessed directly from the CPU.

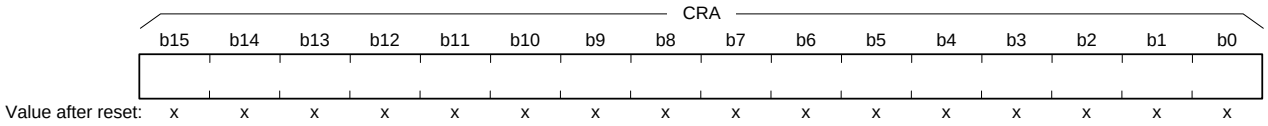


Downloaded from Arrow.com.

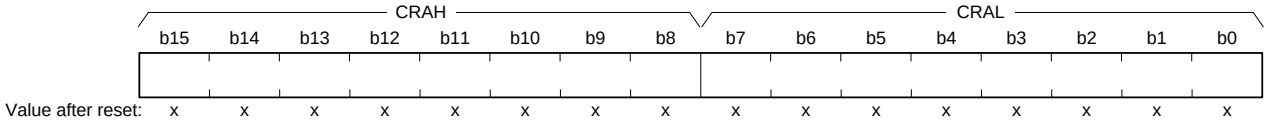
## 13.2.5 DTC Transfer Count Register A (CRA)

Address (inaccessible directly from the CPU)

- Normal transfer mode




- Repeat transfer mode/block transfer mode




Notes: 1. The function depends on transfer mode.
2. x: Undefined

| Symbol | Register Name | Description | R/W |
|---|---|---|---|
| CRAL | Transfer Counter A Lower Register | Set transfer count. | — |
| CRAH | Transfer Counter A Upper Register | | — |

Note: Set CRAH and CRAL to the same value in repeat transfer mode and block transfer mode.

CRA is used to set the transfer count of the DTC.

The function of this register depends on transfer mode.

CRA cannot be accessed directly from the CPU.

### (1) Normal transfer mode (MD[1:0] bits in MRA = 00b)

CRA functions as a 16-bit transfer counter in normal transfer mode.

The transfer count is 1, 65535, and 65536 when the set value is 0001h, FFFFh, and 0000h, respectively.

The CRA value is decremented (-1) at each data transfer.

### (2) Repeat transfer mode (MD[1:0] bits in MRA = 01b)

The CRAH register retains transfer count and the CRAL register functions as an 8-bit transfer counter.

The transfer count is 1, 255, and 256 when the set value is 01h, FFh, and 00h, respectively.

The CRAL value is decremented (-1) at each data transfer. When it reaches 00h, the CRAH value is transferred to CRAL.

### (3) Block transfer mode (MD[1:0] bits in MRA = 10b)

The CRAH register retains block size and the CRAL register functions as an 8-bit block size counter.

The transfer count is 1, 255, and 256 when the set value is 01h, FFh, and 00h, respectively.

The CRAL value is decremented (-1) at each data transfer. When it reaches 00h, the CRAH value is transferred to CRAL.



Downloaded from Arrow.com.

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

Address (inaccessible directly from the CPU)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

[Legend] x: Undefined

CRB is used to set the block transfer count for block transfer mode.

The transfer count is 1, 65535, and 65536 when the set value is 0001h, FFFFh, and 0000h, respectively. The CRB value is decremented (-1) at each data transfer.

When normal transfer mode or repeat transfer mode is selected, this register is not used and the set value is ignored.

CRB cannot be accessed directly from the CPU.

### 13.2.7 DTC Control Register (DTCCR)

Address: 0008 7400h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | RRS | RCHNE | – | – | ERR |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ERR* | Transfer Stop Flag | 0: No DTC transfer stop request is generated<br>1: A DTC transfer stop request is generated | R |
| b2, b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b3 | RCHNE | Chain Transfer Enable after DTC Repeat Transfer | 0: Chain transfer after repeat transfer is disabled<br>1: Chain transfer after repeat transfer is enabled | R/W |
| b4 | RRS | DTC Transfer Data Read Skip Enable | 0: Transfer data read is not skipped<br>1: Transfer data read is skipped when vector numbers match. | R/W |
| b7 to b5 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |

Note: * The DTC will not be activated as long as the value of the ERR bit is 1.
To activate the DTC when transfer has stopped because of a nonmaskable interrupt, clear the interrupt source flag. When transfer has stopped because of the bus error generation, which is not a nonmaskable interrupt, clear the interrupt source by the bus error source clear register (BERCLR) in the bus error monitoring section. For details on nonmaskable interrupts, see section 10, Interrupt Control Unit (ICU). For details on bus errors, see section 11, Buses.

DTCCR is used to specify the control of the DTC.



Downloaded from Arrow.com.

### ERR Flag (Transfer Stop Flag)

The ERR flag indicates that a DTC transfer stop request is present due to a bus error or nonmaskable interrupt.

When the DTC accepts a transfer stop request, it stops data transfer.

[Setting condition]

- When a nonmaskable interrupt is generated, or when a bus error is generated

[Clearing condition]

- When the corresponding source flag is cleared on completion of processing for the nonmaskable interrupt, or when the bus error source is cleared on completion of processing for the bus error interrupt

### RCHNE Bit (Chain Transfer Enable after DTC Repeat Transfer)

The RCHNE bit enables or disables chain transfer when data transfer of the specified count ends (transfer counter = 0) in repeat transfer mode.

When the transfer counter CRAL becomes 00h in repeat transfer mode, chain transfer is not performed because the CRAH value is written to CRAL. Writing 1 to the RCHNE bit enables chain transfer when the transfer counter is rewritten.

### RRS Bit (DTC Transfer Data Read Skip Enable)

The DTC vector number is always compared with the vector number in the previous startup process.

When these vector numbers match and the RRS bit is set to 1, DTC data transfer is performed without reading the transferred data. However, when the previous transfer was chain transfer, the transferred data is always read regardless of the value of RRS bit.

Furthermore, when the transfer counter (CRA register) became 0 during the previous normal transfer and when the transfer counter (CRB register) became 0 during the previous block transfer, the transferred data is always read regardless of the value of RRS bit.

## 13.2.8 DTC Vector Base Register (DTCVBR)

Address: 0008 7404h

| | b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DTCVBR is used to set the base address for calculating the DTC vector table address.

The lower 12 bits (b11 to b0) are always 0 and cannot be modified.

The upper 4 bits (b31 to b28) are ignored, and the address of this register is extended by the value specified by b27.



Downloaded from Arrow.com.

### 13.2.9 DTC Address Mode Register (DTCADMOD)

Address: 0008 7408h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | SHORT |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SHORT | Short-Address Mode | 0: Full-address mode<br>1: Short-address mode | R/W |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |

DTCADMOD is used to select an address mode that specifies DTC's accessible area.

**SHORT Bit (Short-Address Mode)**

Full-address mode allows the DTC to access to a 4-Gbyte space (00000000h to FFFFFFFFh).

Short-address mode allows the DTC to access to a 16-Mbyte space (00000000h to 007FFFFFh and FF800000h to FFFFFFFFh).

### 13.2.10 DTC Module Start Register (DTCST)

Address: 0008 740Ch

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | DTCST |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DTCST | DTC Module Start | 0: DTC module stop<br>1: DTC module start | R/W |
| b7 to b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |

DTCST is used to start or stop the DTC module.

**DTCST Bit (DTC Module Start)**

Set the DTCST bit to 1 to enable the DTC to accept start requests. When this bit is cleared to 0, start requests are no longer accepted.

If this bit is cleared to 0 during data transfer, the accepted start request is active until the processing is completed.

To enable transitions to the module-stop state, all-module clock-stop state, software-standby mode, or deep software-standby mode, the DTCST bit must be set to "0".

For details on the facilities for transitions to the module-stop state, all-module clock-stop state, software-standby mode, and deep software-standby mode, refer to section 13.8.1, Setting the DTC Module Start Register, and section 8, Low Power Consumption.



Downloaded from Arrow.com.

## 13.3 Sources of Activation

The DTC is activated by an interrupt request. Setting ISEL[1:0] bits in an ISELRi register (where i is the interrupt vector number of the given interrupt) of the ICU to 01b selects the corresponding interrupt as an activation source for the DTC, and clearing the same bits to 00b selects the interrupt as a source of interrupts for the CPU.

On completion of a single round of data transfer (or the last of the consecutive transfers in the case of a chained transfer), the flag for the interrupt that was the source of activation and the ISEL[1:0] bits in the corresponding ISELRi register are cleared to 00b.

### 13.3.1 Allocating Transfer Data and DTC Vector Table

Transfer data is allocated in the data area. Be sure to set multiples of 4 for the transfer data start addresses in the vector table. Because, such addresses are accessed with their lowest 2 bits fixed at 00b.

Transfer data can be allocated with 3 longwords (short-address mode) or 4 longwords (full-address mode). Use the SHORT bit in DTCADMOD to select short-address mode (SHORT bit = 1) or full-address mode (SHORT bit = 0).

Figure 13.2 shows the allocation of transfer data in the data area. The DTC reads the transfer data start address for each startup source, and then reads the transfer data from this start address. Figure 13.3 shows the DTC vector table and transfer data.

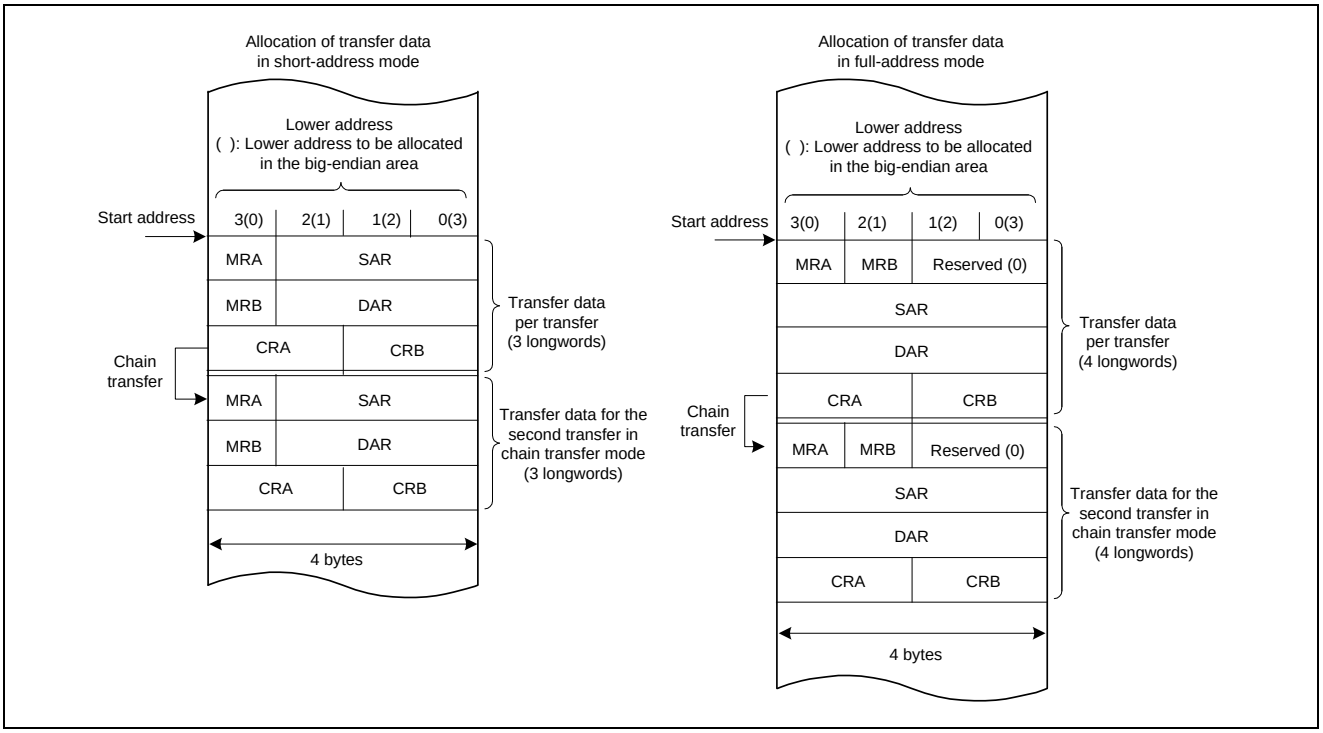



**Figure 13.2 Allocation of Transfer Data in the Data Area**



Downloaded from Arrow.com.

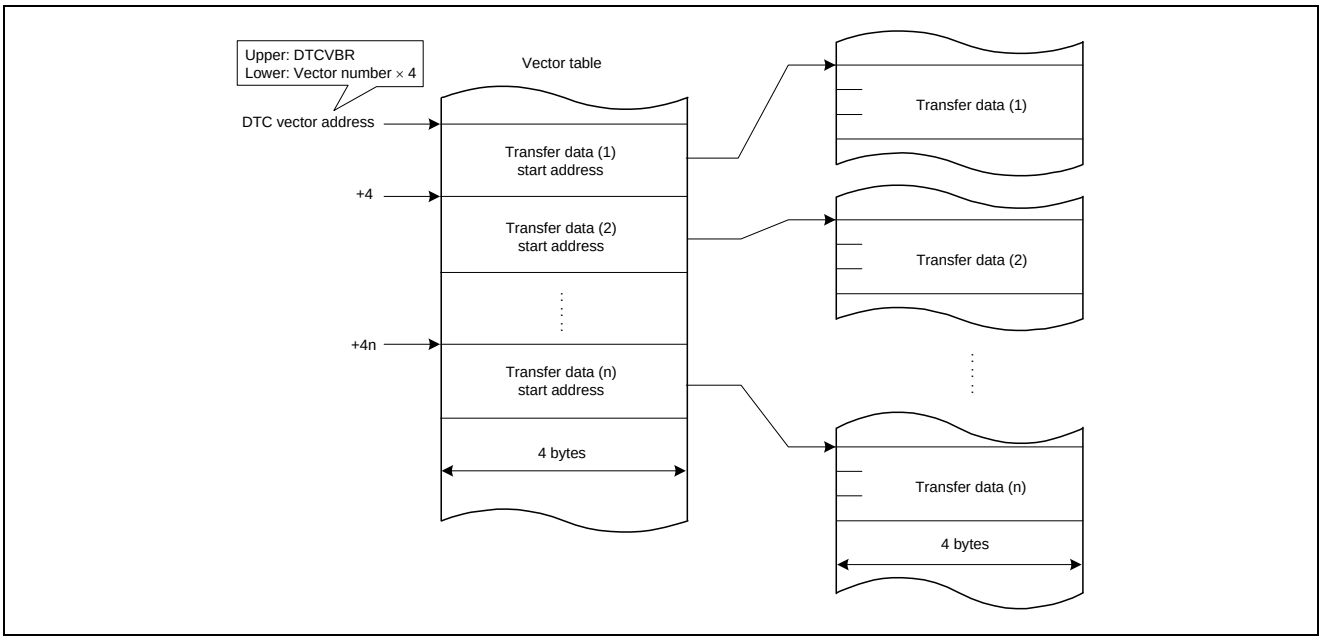



**Figure 13.3 DTC Vector Table and Transfer Data**



Downloaded from Arrow.com.

### 13.3.2 Startup source and Vector Address

Table 13.3 shows the correspondence between DTC startup sources and vector addresses.

**Table 13.3 Correspondence between Interrupt Sources, DTC Vector Addresses, and the ISELRi Register of the ICU**

| Activation Request Generation Source | Activation Source | Vector Number | DTC Vector Address Offset | ISELRi Register of the ICU | Priority |
|---|---|---|---|---|---|
| CMT Unit 0 | CMT0 | 28 | 0070h | ISELR028 | High |
| | CMT1 | 29 | 0074h | ISELR029 | |
| CMT Unit 1 | CMT2 | 30 | 0078h | ISELR030 | |
| | CMT3 | 31 | 007Ch | ISELR031 | |
| External pins | IRQ0 | 64 | 0100h | ISELR064 | |
| | IRQ1 | 65 | 0104h | ISELR065 | |
| | IRQ2 | 66 | 0108h | ISELR066 | |
| | IRQ3 | 67 | 010Ch | ISELR067 | |
| | IRQ4 | 68 | 0110h | ISELR068 | |
| | IRQ5 | 69 | 0114h | ISELR069 | |
| | IRQ6 | 70 | 0118h | ISELR070 | |
| | IRQ7 | 71 | 011Ch | ISELR071 | |
| | IRQ8 | 72 | 0120h | ISELR072 | |
| | IRQ9 | 73 | 0124h | ISELR073 | |
| | IRQ10 | 74 | 0128h | ISELR074 | |
| | IRQ11 | 75 | 012Ch | ISELR075 | |
| | IRQ12 | 76 | 0130h | ISELR076 | |
| | IRQ13 | 77 | 0134h | ISELR077 | |
| | IRQ14 | 78 | 0138h | ISELR078 | |
| | IRQ15 | 79 | 013Ch | ISELR079 | |
| AD0 | ADI0 | 98 | 0188h | ISELR098 | |
| AD1 | ADI1 | 99 | 018Ch | ISELR099 | |
| AD2 | ADI2 | 100 | 0190h | ISELR100 | |
| AD3 | ADI3 | 101 | 0194h | ISELR101 | |
| TPU0 | TGI0A | 104 | 01A0h | ISELR104 | |
| | TGI0B | 105 | 01A4h | ISELR105 | |
| | TGI0C | 106 | 01A8h | ISELR106 | |
| | TGI0D | 107 | 01ACh | ISELR107 | |
| TPU1 | TGI1A | 111 | 01BCh | ISELR111 | |
| | TGI1B | 112 | 01C0h | ISELR112 | |
| TPU2 | TGI2A | 117 | 01D4h | ISELR117 | |
| | TGI2B | 118 | 01D8h | ISELR118 | |
| TPU3 | TGI3A | 122 | 01E8h | ISELR122 | |
| | TGI3B | 123 | 01ECh | ISELR123 | |
| | TGI3C | 124 | 01F0h | ISELR124 | |
| | TGI3D | 125 | 01F4h | ISELR125 | |
| TPU4 | TGI4A | 127 | 01FCh | ISELR127 | |
| | TGI4B | 128 | 0200h | ISELR128 | Low |



Downloaded from Arrow.com.

| Activation Request Generation Source | Activation Source | Vector Number | DTC Vector Address Offset | ISELRi Register of the ICU | Priority |
|---|---|---|---|---|---|
| TPU5 | TGI5A | 133 | 0214h | ISELR133 | High |
| | TGI5B | 134 | 0218h | ISELR134 | |
| TPU6 | TGI6A | 138 | 0228h | ISELR138 | |
| | TGI6B | 139 | 022Ch | ISELR139 | |
| | TGI6C | 140 | 0230h | ISELR140 | |
| | TGI6D | 141 | 0234h | ISELR141 | |
| TPU7 | TGI7A | 145 | 0244h | ISELR145 | |
| | TGI7B | 146 | 0248h | ISELR146 | |
| TPU8 | TGI8A | 151 | 025Ch | ISELR151 | |
| | TGI8B | 152 | 0260h | ISELR152 | |
| TPU9 | TGI9A | 156 | 0270h | ISELR156 | |
| | TGI9B | 157 | 0274h | ISELR157 | |
| | TGI9C | 158 | 0278h | ISELR158 | |
| | TGI9D | 159 | 027Ch | ISELR159 | |
| TPU10 | TGI10A | 161 | 0284h | ISELR161 | |
| | TGI10B | 162 | 0288h | ISELR162 | |
| TPU11 | TGI11A | 167 | 029Ch | ISELR167 | |
| | TGI11B | 168 | 02A0h | ISELR168 | |
| TMR0 | CMIA0 | 174 | 02B8h | ISELR174 | |
| | CMIB0 | 175 | 02BCh | ISELR175 | |
| TMR1 | CMIA1 | 177 | 02C4h | ISELR177 | |
| | CMIB1 | 178 | 02C8h | ISELR178 | |
| TMR2 | CMIA2 | 180 | 02D0h | ISELR180 | |
| | CMIB2 | 181 | 02D4h | ISELR181 | |
| TMR3 | CMIA3 | 183 | 02DCh | ISELR183 | |
| | CMIB3 | 184 | 02E0h | ISELR184 | |
| DMAC | DMTEND0 | 198 | 0318h | ISELR198 | |
| | DMTEND1 | 199 | 031Ch | ISELR199 | |
| | DMTEND2 | 200 | 0320h | ISELR200 | |
| | DMTEND3 | 201 | 0324h | ISELR201 | |
| SCI0 | RXI0 | 215 | 035Ch | ISELR215 | |
| | TXI0 | 216 | 0360h | ISELR216 | |
| SCI1 | RXI1 | 219 | 036Ch | ISELR219 | |
| | TXI1 | 220 | 0370h | ISELR220 | |
| SCI2 | RXI2 | 223 | 037Ch | ISELR223 | |
| | TXI2 | 224 | 0380h | ISELR224 | |
| SCI3 | RXI3 | 227 | 038Ch | ISELR227 | |
| | TXI3 | 228 | 0390h | ISELR228 | |
| SCI4 | RXI4 | 231 | 039Ch | ISELR231 | |
| | TXI4 | 232 | 03A0h | ISELR232 | |
| SCI5 | RXI5 | 235 | 03ACh | ISELR235 | |
| | TXI5 | 236 | 03B0h | ISELR236 | Low |



Downloaded from Arrow.com.

| Activation Request Generation Source | Activation Source | Vector Number | DTC Vector Address Offset | ISELRi Register of the ICU | Priority |
|---|---|---|---|---|---|
| SCI6 | RXI6 | 239 | 03BCh | ISELR239 | High |
| | TXI6 | 240 | 03C0h | ISELR240 | |
| RIIC0 | RXI0 | 247 | 03DCh | ISELR247 | |
| | TXI0 | 248 | 03E0h | ISELR248 | |
| RIIC1 | RXI1 | 251 | 03ECh | ISELR251 | |
| | TXI1 | 252 | 03F0h | ISELR252 | Low |



Downloaded from Arrow.com.

## 13.4 Operation

The DTC stores transfer data in the data area.

When the DTC is activated, it reads the DTC vector corresponding to the vector number. Then the DTC reads transfer data from the transfer data store address pointed by the DTC vector, transfers data, and then writes back the transfer data after the data transfer. Storing transfer data in the data area allows data transfer of arbitrary number of channels.

There are three transfer modes: normal transfer mode, repeat transfer mode, and block transfer mode.

The DTC specifies a transfer source address in SAR and a transfer destination address in DAR. The values of these registers are incremented, decremented, or remain unchanged independently after data transfer.

Table 13.4 lists transfer modes of the DTC.

**Table 13.4 Transfer Modes of the DTC**

| Transfer Mode | Data Size Transferred on a Single Transfer Request | Increment/Decrement of Memory Address | Settable Transfer Count |
|---|---|---|---|
| Normal transfer mode | One byte/word/longword | Incremented/decremented by 1, 2, or 4 or address fixed | 1 to 65536 |
| Repeat transfer mode*1 | One byte/word/longword | Incremented/decremented by 1, 2 or, 4 or address fixed | 1 to 256*3 |
| Block transfer mode*2 | Block size specified in CRAH (1 to 256 bytes/words/longwords) | Incremented/decremented by 1, 2 or, 4 or address fixed | 1 to 65536 |

Notes: 1. Set transfer source or transfer destination in the repeat area.
2. Set transfer source or transfer destination in the block area.
3. After data transfer of the specified count, the initial state is restored and the operation is continued (repeated).

Setting the CHNE bit in MRB to 1 allows multiple transfers (chain transfer) on a single startup source. Chain transfer is enabled when transfer counter = 0 by setting the CHNS bit in MRB to 1.

Figure 13.4 shows the operation flowchart of the DTC. Table 13.5 shows chain transfer conditions (excluding combinations of the second transfer and the third transfer, and combinations of subsequent transfers).



Downloaded from Arrow.com.

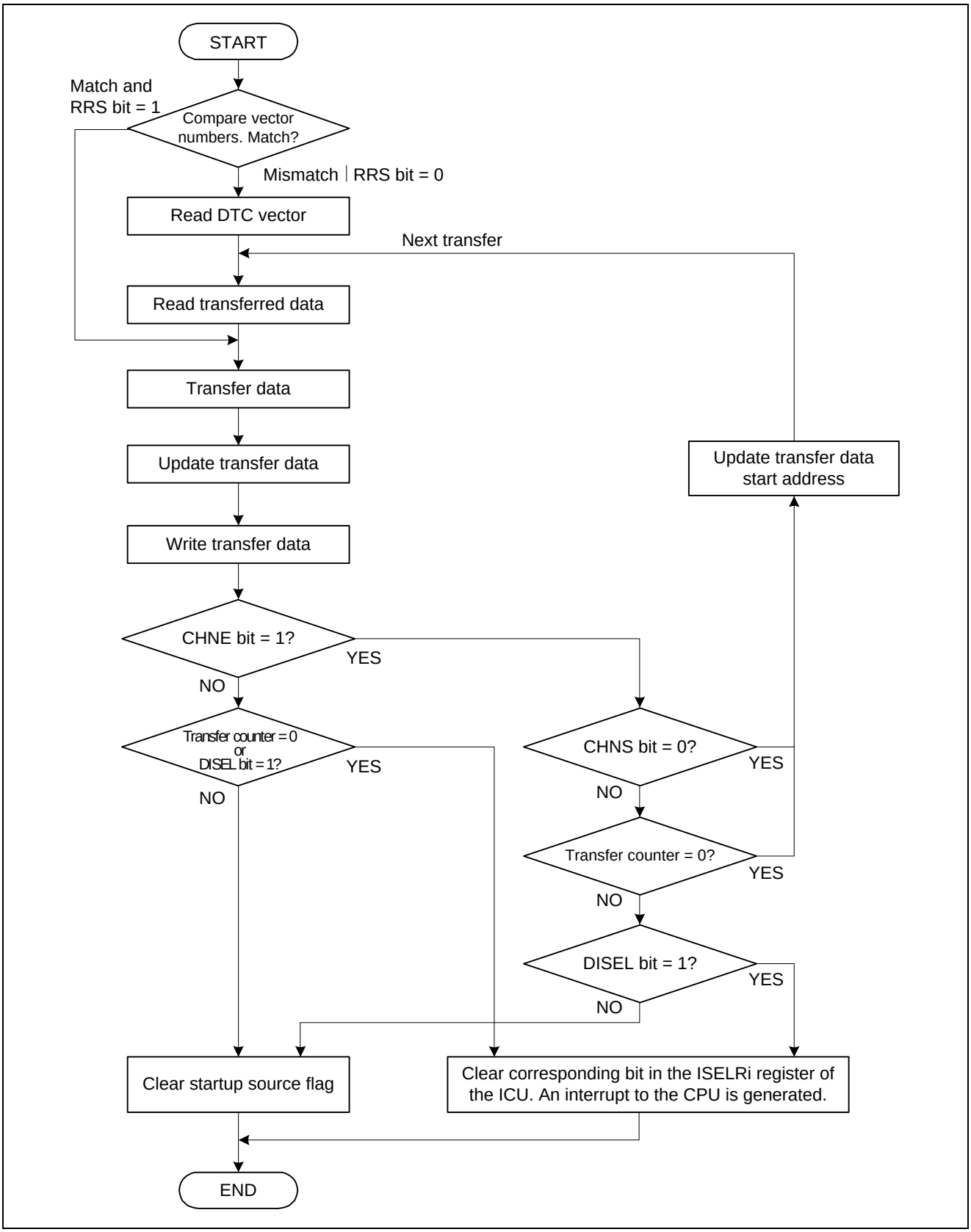



**Figure 13.4 Operation Flowchart of the DTC**



Downloaded from Arrow.com.

**Table 13.5 Chain Transfer Conditions**

| First Transfer | | | | Second Transfer | | | | |
|---|---|---|---|---|---|---|---|---|
| CHNE Bit | CHNS Bit | DISEL Bit | Transfer Counter*1 | CHNE Bit | CHNS Bit | DISEL Bit | Transfer Counter*1 | DTC Transfer |
| 0 | — | 0 | Other than 0 | — | — | — | — | Ends after the first transfer |
| 0 | — | 0 | 0*2 | — | — | — | — | Ends after the first transfer with an interrupt request to the CPU |
| 0 | — | 1 | — | — | — | — | — | |
| 1 | 0 | — | — | 0 | — | 0 | Other than 0 | Ends after the second transfer |
| | | | | 0 | — | 0 | 0*2 | Ends after the second transfer with an interrupt request to the CPU |
| | | | | 0 | — | 1 | — | |
| 1 | 1 | 0 | Other than 0 | — | — | — | — | Ends after the first transfer |
| 1 | 1 | — | 0*2 | 0 | — | 0 | Other than 0 | Ends after the second transfer |
| | | | | 0 | — | 0 | 0*2 | Ends after the second transfer with an interrupt request to the CPU |
| | | | | 0 | — | 1 | — | |
| 1 | 1 | 1 | Other than 0 | — | — | — | — | Ends after the first transfer with an interrupt request to the CPU |

Notes: 1. Normal transfer mode: CRA, Repeat transfer mode: CRAL, Block transfer mode: CRB

2. When the CRAL value is replaced with the CRAH value in repeat transfer mode



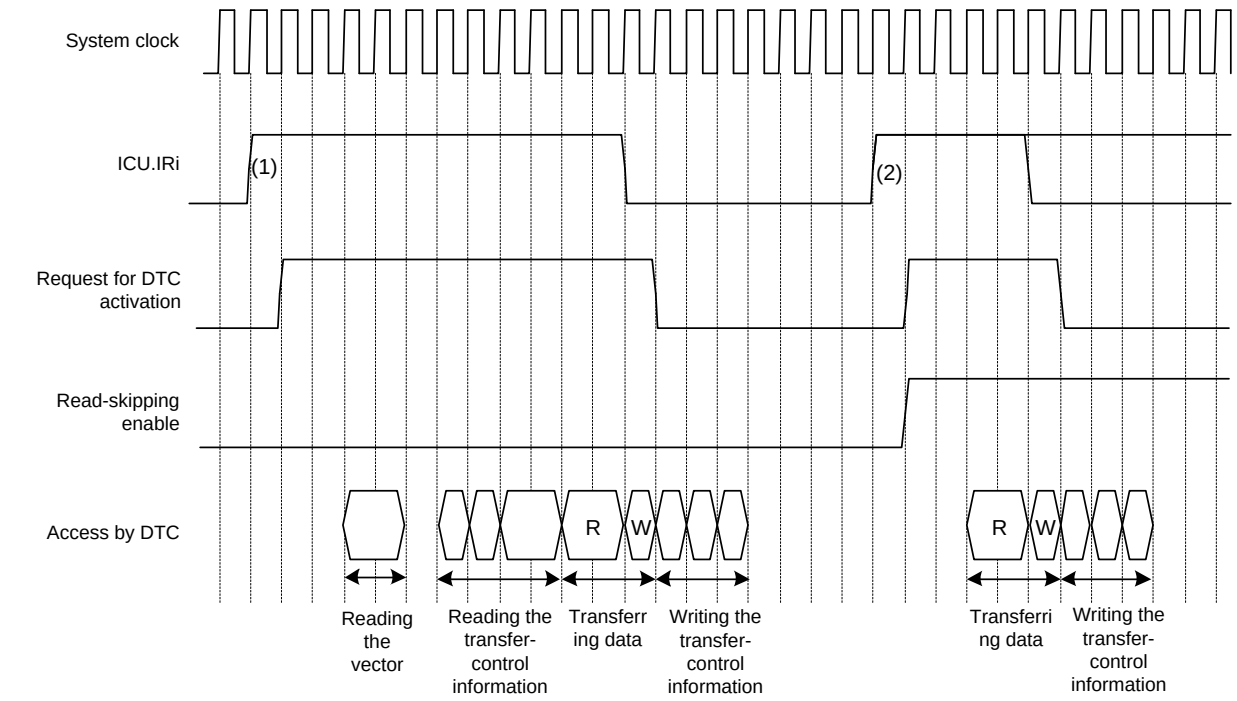
Downloaded from Arrow.com.

### 13.4.1 Transfer Data Read Skip Function

Vector address read and transfer data read can be skipped by the setting of the RRS bit in DTCCR.

When a DTC startup request is generated, the current DTC vector number is always compared with the DTC vector number in the previous startup process. When these vector numbers match and the RRS bit is set to 1, DTC data transfer is performed without reading the vector address and transfer data. However, when the previous transfer was chain transfer, the vector address and transfer data are always read. Furthermore, when the transfer counter (CRA register) became 0 during the previous normal transfer and when the transfer counter (CRB register) became 0 during the previous block transfer, transfer data is always read regardless of the value of RRS bit. Figure 13.5 shows an example of transfer data read skip.

To update the vector table and transfer data, set the RRS bit to 0, update the vector table and transfer data, and then set the RRS bit to 1. When the RRS bit is set to 0, the retained vector number is discarded and the vector table and transfer data that are updated in the following startup process are read.




i = DTC vector number (interrupt vector number)

Note: Since the activation request (vector number) is the same at points (1) and (2), if bit RRS = 1, reading of the transfer-control information is skipped for request (2).

**Figure 13.5 Operation with Skipping of the Transfer-Control Information**



Downloaded from Arrow.com.

### 13.4.2 Transfer Data Write-Back Skip Function

When the SM[1:0] bits in MRA or the DM[1:0] bits in MRB are set to "address fixed", a part of transfer data is not written back. This function is performed independently of the setting of short-address mode or full-address mode. Table 13.6 lists transfer data write-back skip conditions and applicable registers.

The CRA and CRB registers are always written back independently of the setting of short-address mode or full-address mode. Furthermore, in full-address mode, write-back of the MRA and MRB registers are always skipped.

**Table 13.6 Transfer Data Write-Back Skip Conditions and Applicable Registers**

| SM[1:0] Bits in MRA | | DM[1:0] Bits in MRB | | | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b3** | **b2** | **SAR Register** | **DAR Register** |
| 0 | 0 | 0 | 0 | Skip | Skip |
| 0 | 0 | 0 | 1 | | |
| 0 | 1 | 0 | 0 | | |
| 0 | 1 | 0 | 1 | | |
| 0 | 0 | 1 | 0 | Skip | Write-back |
| 0 | 0 | 1 | 1 | | |
| 0 | 1 | 1 | 0 | | |
| 0 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | Write-back | Skip |
| 1 | 0 | 0 | 1 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | |
| 1 | 0 | 1 | 0 | Write-back | Write-back |
| 1 | 0 | 1 | 1 | | |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | |



Downloaded from Arrow.com.

### 13.4.3 Normal Transfer Mode

This mode allows 1-byte, 1-word, or 1-longword data transfer on a single startup source. The transfer count can be set to 1 to 65536.

Transfer source addresses and transfer destination addresses can be set to increment, decrement, or fixed independently. This mode enables an interrupt request to the CPU to be generated at the end of specified-count transfer.

Table 13.7 lists register functions in normal transfer mode, and figure 13.6 shows the memory map of normal transfer mode.

**Table 13.7 Register Functions in Normal Transfer Mode**

| Register | Description | Value Written Back by Writing Transfer Data |
|---|---|---|
| SAR | Transfer source address | Increment/decrement/fixed* |
| DAR | Transfer destination address | Increment/decrement/fixed* |
| CRA | Transfer count A | CRA - 1 |
| CRB | Transfer count B | Not updated |

Note: * Write-back operation is skipped in address-fixed mode.

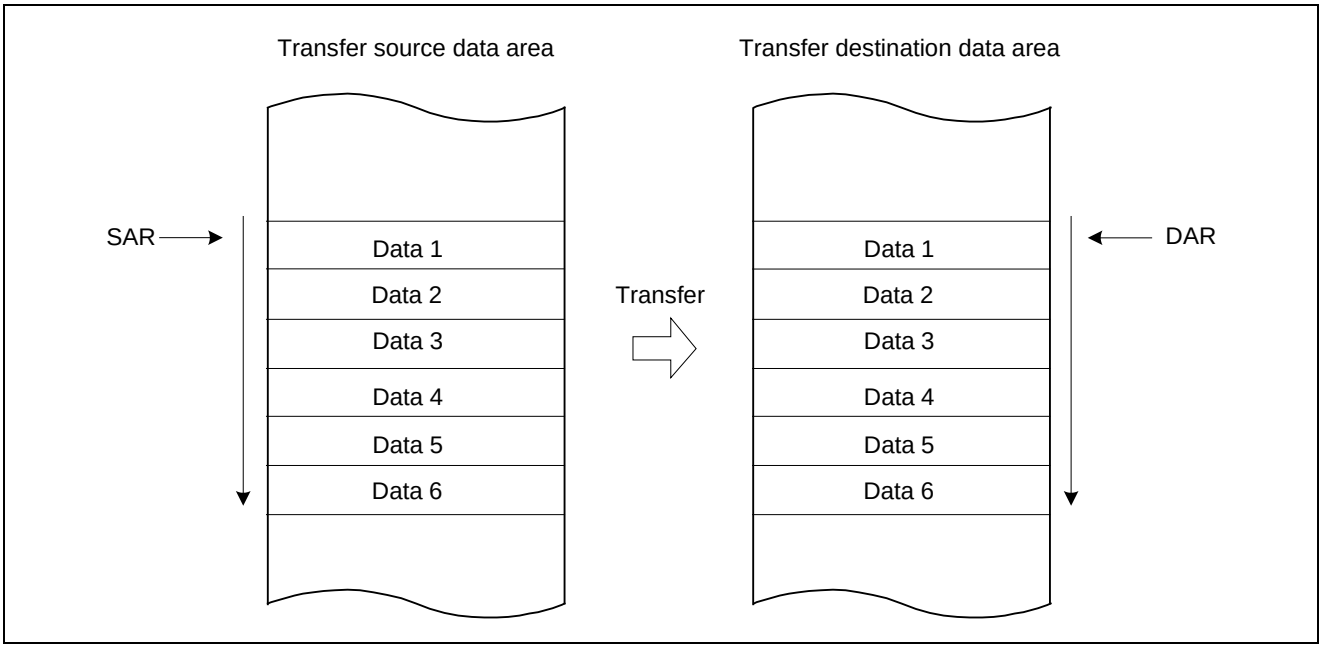



**Figure 13.6 Memory Map of Normal Transfer Mode**



Downloaded from Arrow.com.

### 13.4.4 Repeat Transfer Mode

This mode allows 1-byte, 1-word, or 1-longword data transfer on a single startup source.

Specify either transfer source or transfer destination for the repeat area by the DTS bit in MRB. The transfer count can be set to 1 to 256. When the specified-count transfer is completed, the initial value of the address register specified in the transfer counter and the repeat area is restored and transfer is repeated. The other address register is incremented or decremented continuously or remains unchanged.

When the transfer counter CRAL is decreased to 00h in repeat transfer mode, the CRAL value is updated to the value set in CRAH. Thus the transfer counter does not become 00h, which inhibits generation of interrupt request to the CPU when the DISEL bit in MRB is set to 0 (an interrupt request to the CPU is generated when specified data transfer is completed).

Table 13.8 lists the register functions in repeat transfer mode, and figure 13.7 shows the memory map of repeat transfer mode.

**Table 13.8 Register Functions in Repeat Transfer Mode**

| | | Value Written Back by Writing Transfer Data | |
|---|---|---|---|
| **Register** | **Description** | **When CRAL is not 1** | **When CRAL is 1** |
| SAR | Transfer source address | Increment/decrement/fixed* | (When the DTS bit in MRB is 0)<br>Increment/decrement/fixed*<br>(When the DTS bit in MRB is 1)<br>SAR register initial value |
| DAR | Transfer destination address | Increment/decrement/fixed* | (When the DTS bit in MRB is 0)<br>DAR register initial value<br>(When the DTS bit in MRB is 1)<br>Increment/decrement/fixed* |
| CRAH | Retains transfer count | CRAH | CRAH |
| CRAL | Transfer count A | CRAL - 1 | CRAH |
| CRB | Transfer count B | Not updated | Not updated |

Note: * Write-back is skipped in address-fixed mode.



Downloaded from Arrow.com.

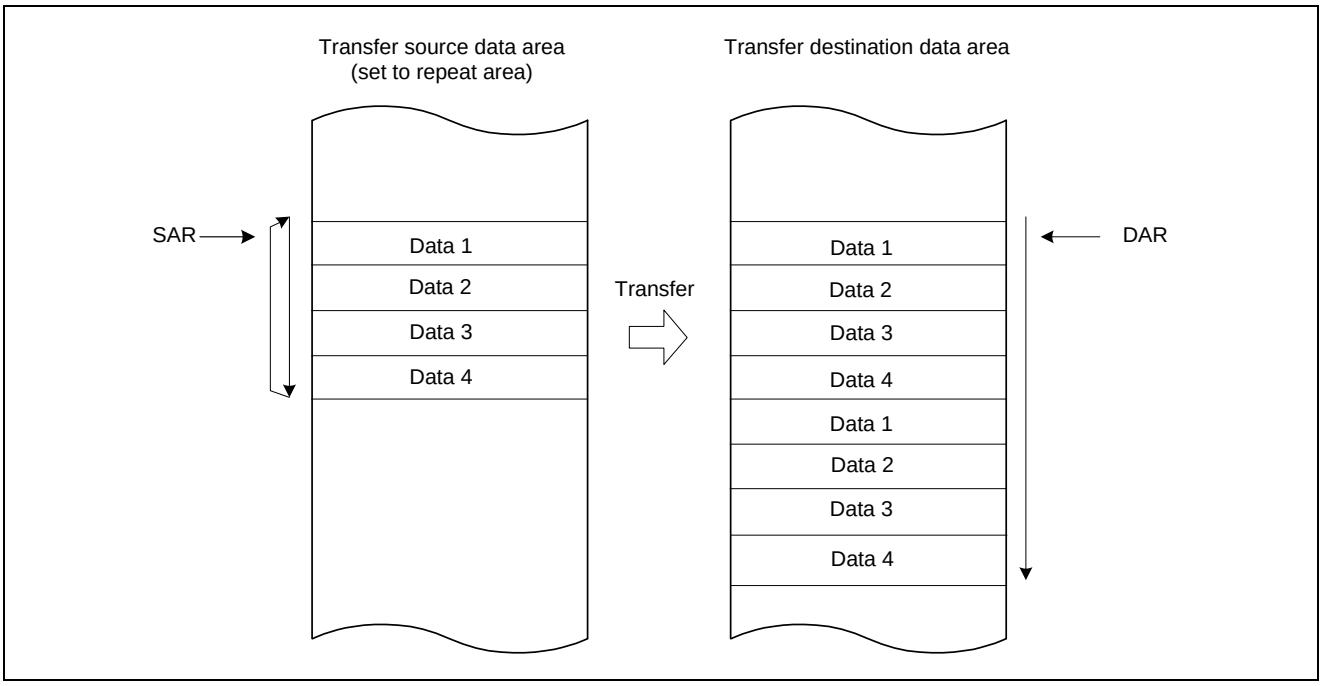



**Figure 13.7 Memory Map of Repeat Transfer Mode (Transfer Source: Repeat Area)**



Downloaded from Arrow.com.

### 13.4.5 Block Transfer Mode

This mode allows single-block data transfer on a single startup source.

Specify either transfer source or transfer destination for the block area by the DTS bit in MRB. The block size can be set to 1 to 256 bytes (or 1 to 256 words or 1 to 256 longwords).

When transfer of the specified one block is completed, the initial values of the block size counter CRAL and the address register (SAR when the DTS bit = 1 or DAR when the DTS bit = 0) specified in the block area are restored. The other address register is incremented or decremented continuously or remains unchanged.

The transfer count (block count) can be set to 1 to 65536. This mode enables an interrupt request to the CPU to be generated at the end of specified-count block transfer.

Table 13.9 lists register functions in block transfer mode, and figure 13.8 shows the memory map of block transfer mode.

**Table 13.9 Register Functions in Block Transfer Mode**

| Register | Description | Value Written Back by Writing Transfer Data |
|---|---|---|
| SAR | Transfer source address | (When DTS bit in MRB is 0)<br>Increment/decrement/fixed*<br>(When DTS bit in MRB is 1)<br>SAR register initial value |
| DAR | Transfer destination address | (When DTS bit in MRB is 0)<br>DAR register initial value<br>(When DTS bit in MRB is 1)<br>Increment/decrement/fixed* |
| CRAH | Retains block size | CRAH |
| CRAL | Block size counter | CRAH |
| CRB | Block transfer counter | CRB - 1 |

Note: * Write-back is skipped in address-fixed mode.

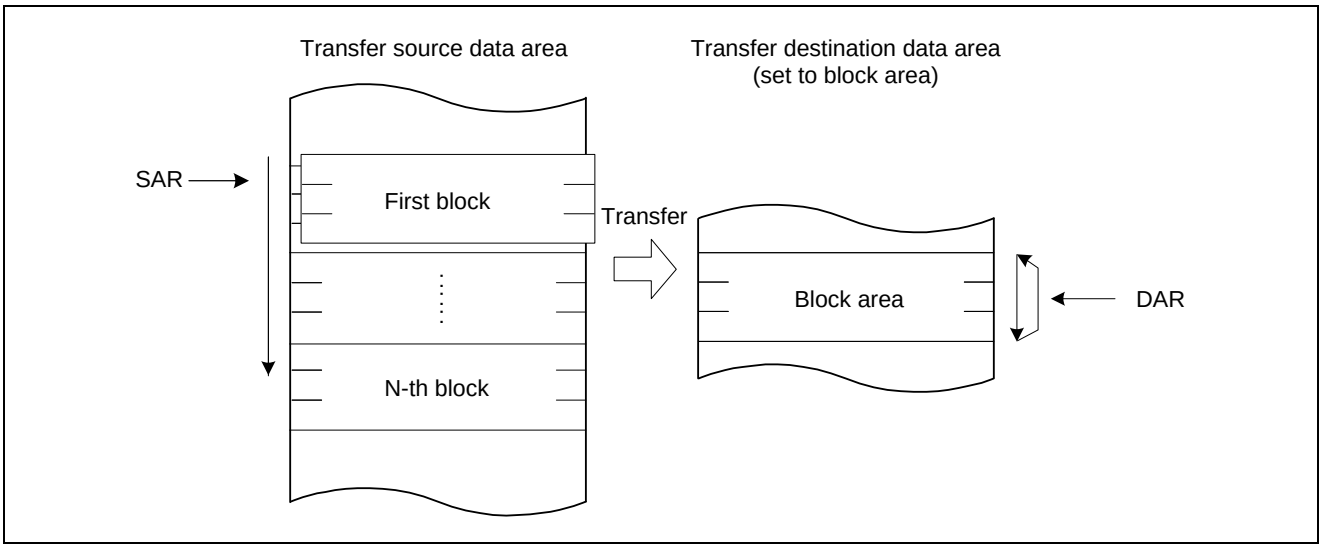



**Figure 13.8 Memory Map of Block Transfer Mode (Transfer Destination: Block Area)**



Downloaded from Arrow.com.

### 13.4.6 Chain Transfer

Setting the CHNE bit in MRB to 1 allows multiple data transfers continuously on a single startup source. Chain transfer can be set independently for registers SAR, DAR, CRA, and CRB that define data transfer, as well as for registers MRA and MRB. Figure 13.9 shows chain transfer operation.

When the CHNE and CHNS bits in MRB are set to 1 and 0, respectively, no interrupt is issued to the CPU and no interrupt source flag as the startup source is cleared, but chain transfer is performed instead based on the next transfer information.

When the CHNE and CHNS bits in MRB are both set to 1, no interrupt is issued to the CPU and no interrupt source flag as the startup source is cleared when the transfer counter value is 0, but chain transfer is performed instead based on the next transfer information. When the transfer counter value is not 0, the interrupt is issued to the CPU and the interrupt source flag as the startup source is cleared by setting the DISEL bit in MRB to 1.

In chain transfer, the interrupt is issued to the CPU or the interrupt source flag as the startup source is cleared upon transfer completion caused by clearing the CHNE bit in MRB to 0.

For detailed chain transfer flow, refer to figure 13.4, Operation Flowchart of the DTC, and table 13.5, Chain Transfer Conditions.

In repeat transfer mode, writing 1 to the RCHNE bit in DTCCR and the CHNE and CHNS bits in MRB enables chain transfer after data transfer of transfer counter = 1.

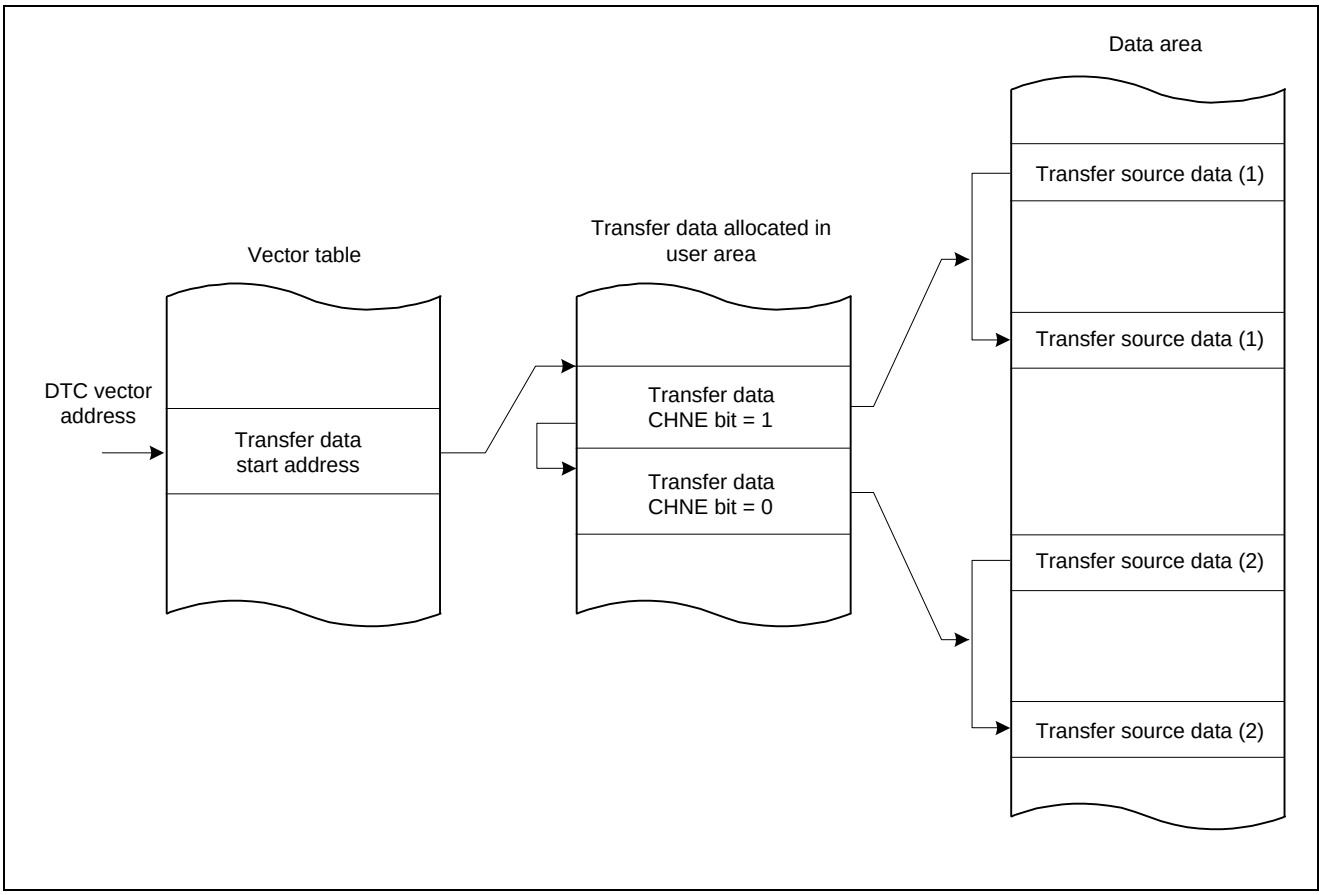



**Figure 13.9 Chain Transfer Operation**



Downloaded from Arrow.com.

### 13.4.7 Operation Timing

Figures 13.10 to 13.13 show examples of DTC operation timing.

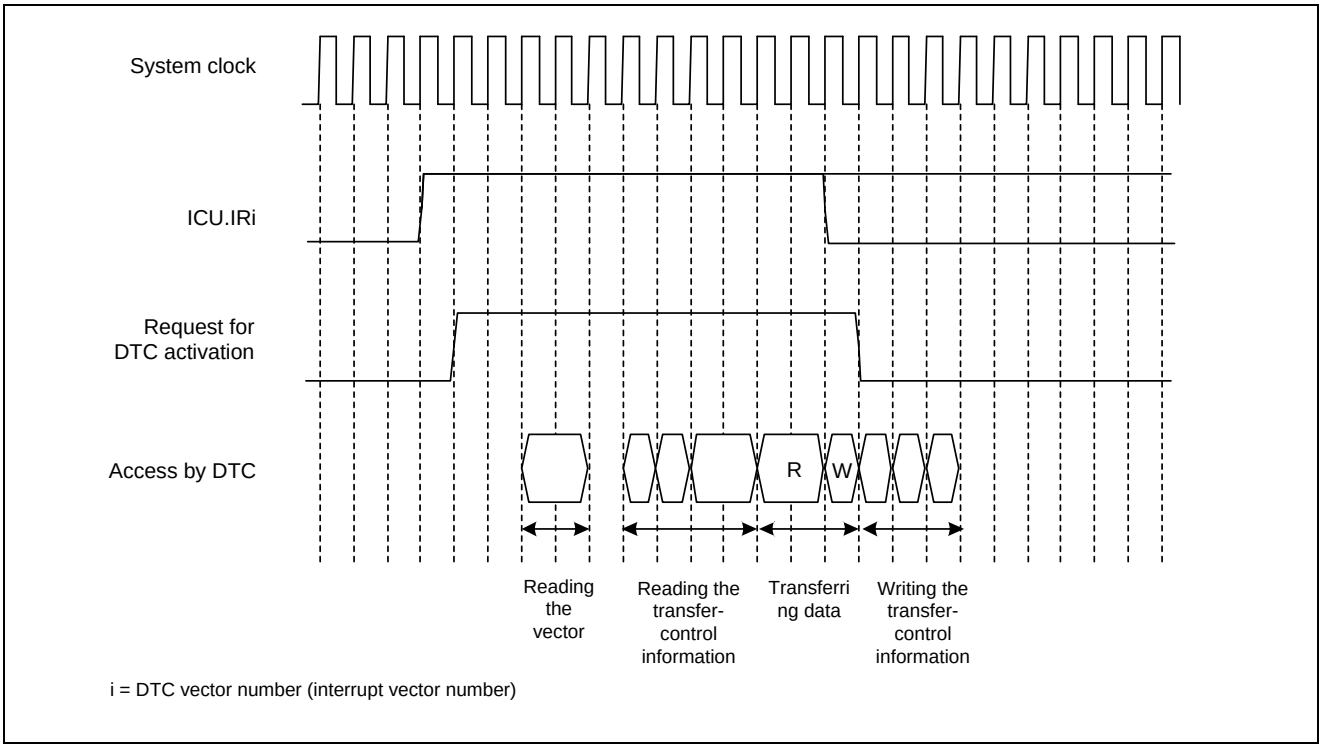



**Figure 13.10 Example of DTC Operation Timing 1**
**(Short-Address Mode, Normal Transfer Mode, Repeat Transfer Mode)**

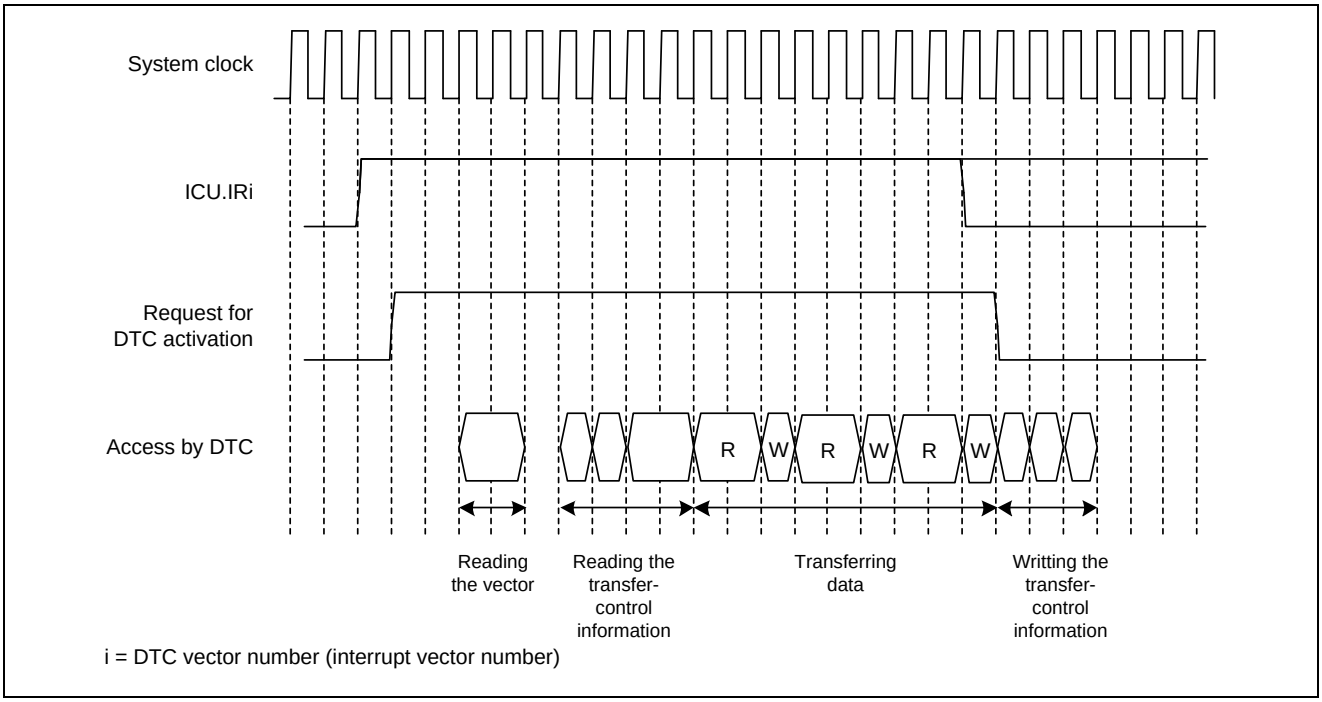



**Figure 13.11 Example of DTC Operation Timing 2**
**(Short-Address Mode, Block Transfer Mode, Block Size = 3)**



Downloaded from Arrow.com.

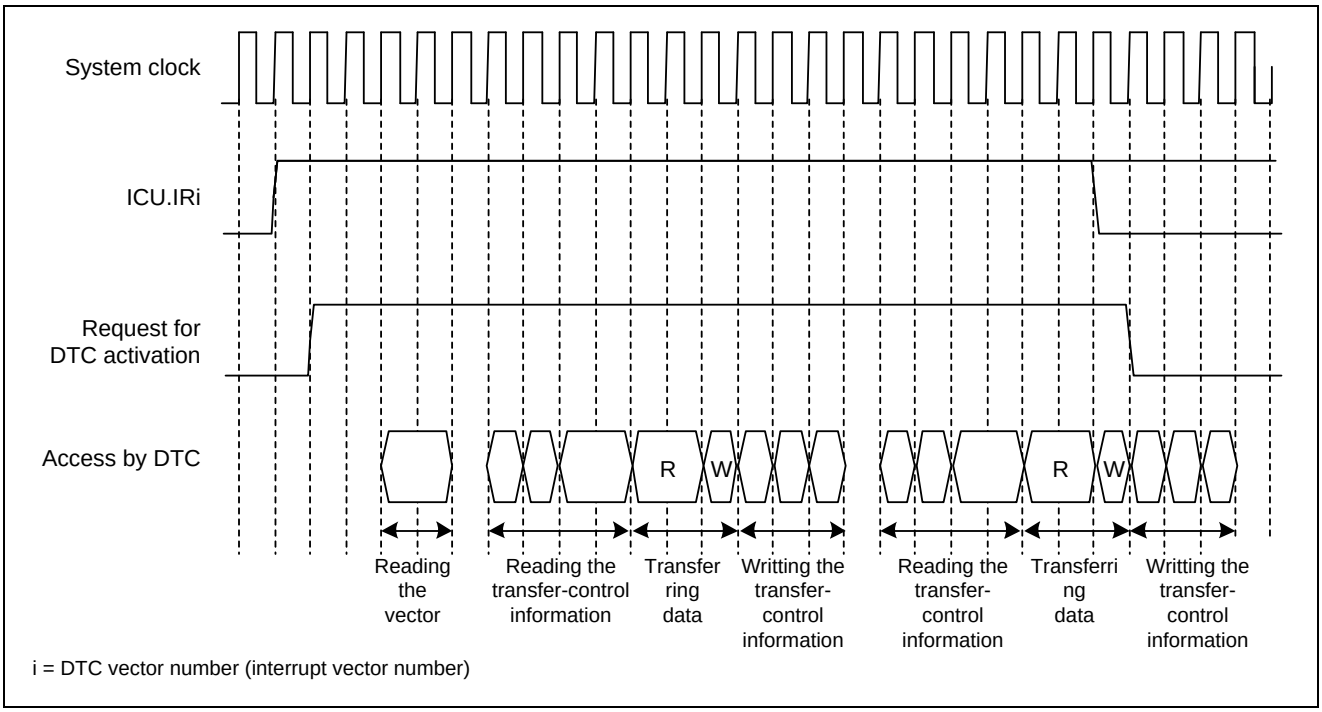



**Figure 13.12 Example of DTC Operation Timing 3 (Short-Address Mode, Chain Transfer)**

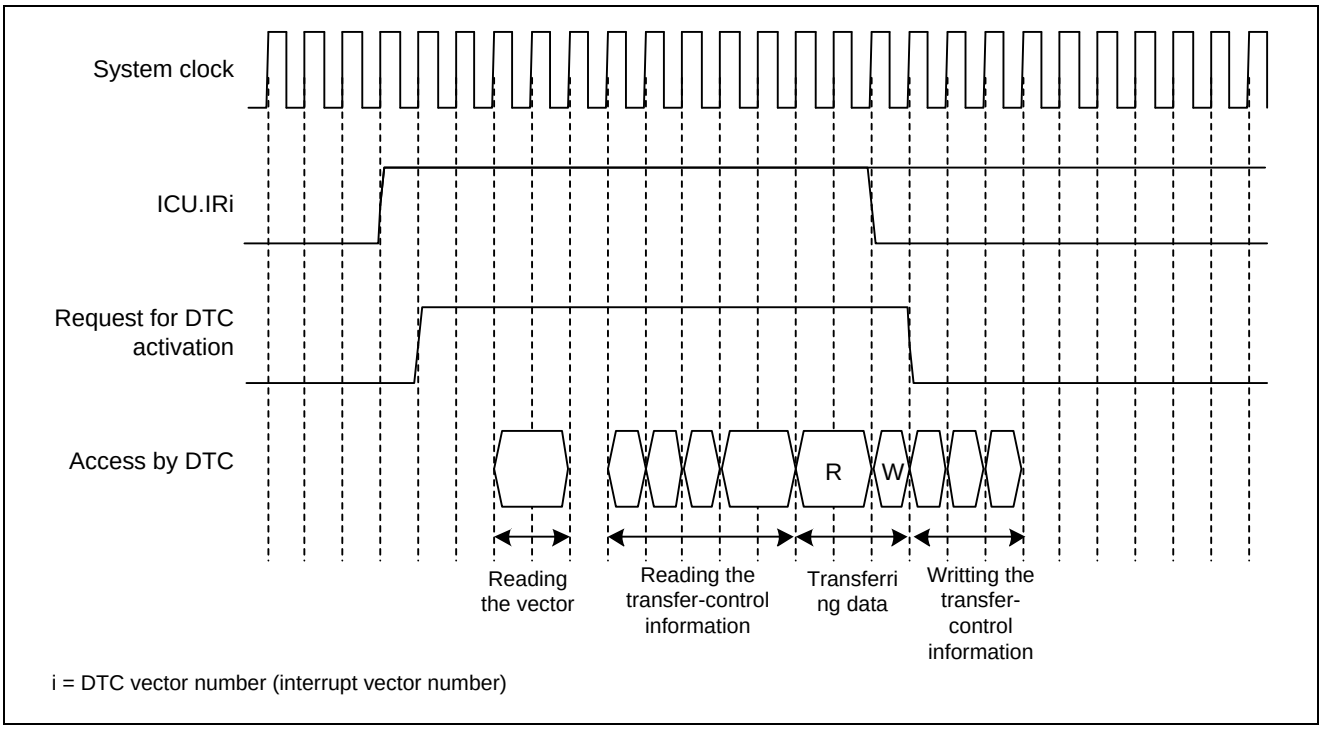



**Figure 13.13 Example of DTC Operation Timing 4 (Full-Address Mode, Normal Transfer Mode, Repeat Transfer Mode)**



Downloaded from Arrow.com.

## 13.4.8 Execution Cycle of the DTC

Table 13.10 lists the execution cycle of single data transfer of the DTC.

**Table 13.10 Table 13.10 Execution Cycle of the DTC**

| Transfer Mode | Vector Read | | Transfer Data Read | | | Transfer Data Write | | | Data Read | Data Write | Internal Operation | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Normal | V+1 | 0*1 | 4 × C + 1*2 | 3 × C + 1*3 | 0*1 | 3 × C*2, *3 | 2 × C*4 | C*5 | R+1 | W | 1 | 0*1 |
| Repeat | V+1 | 0*1 | 4 × C + 1*2 | 3 × C + 1*3 | 0*1 | 3 × C*2, *3 | 2 × C *4 | C *5 | R+1 | W | 1 | 0*1 |
| Block | V+1 | 0*1 | 4 × C + 1*2 | 3 × C + 1*3 | 0*1 | 3 × C*2, *3 | 2 × C *4 | C *5 | (R+1) × P | W × P | 1 | 0*1 |

Notes: 1. Transfer data skip
2. Full-address mode
3. Short-address mode
4. When SAR or DAR is set to address-fixed mode
5. When SAR and DAR are set to address-fixed mode

[Legend] P: Block size (set by CRAH and CRAL)
V: Access cycles for the vector information data storing destination
C: Access cycles for the transfer information data storing destination
R: Access cycles for the data read destination
W: Access cycles for the data write destination
(V, C, R, and W values depend on the access destination. For the number of cycles for the access destination, see section 25, RAM, section 26, ROM (Flash Memory for Code Storage), section 5, I/O Registers, and section 11, Buses.)

## 13.4.9 DTC Bus Mastership Release Timing

The DTC does not release the bus mastership during transfer data read and transfer data write. While transfer data is not read or written, bus arbitration is made according to the priority determined by the bus master arbitrator.



Downloaded from Arrow.com.

## 13.5 DTC Setting Procedure

Figure 13.14 shows the procedure to set the DTC.

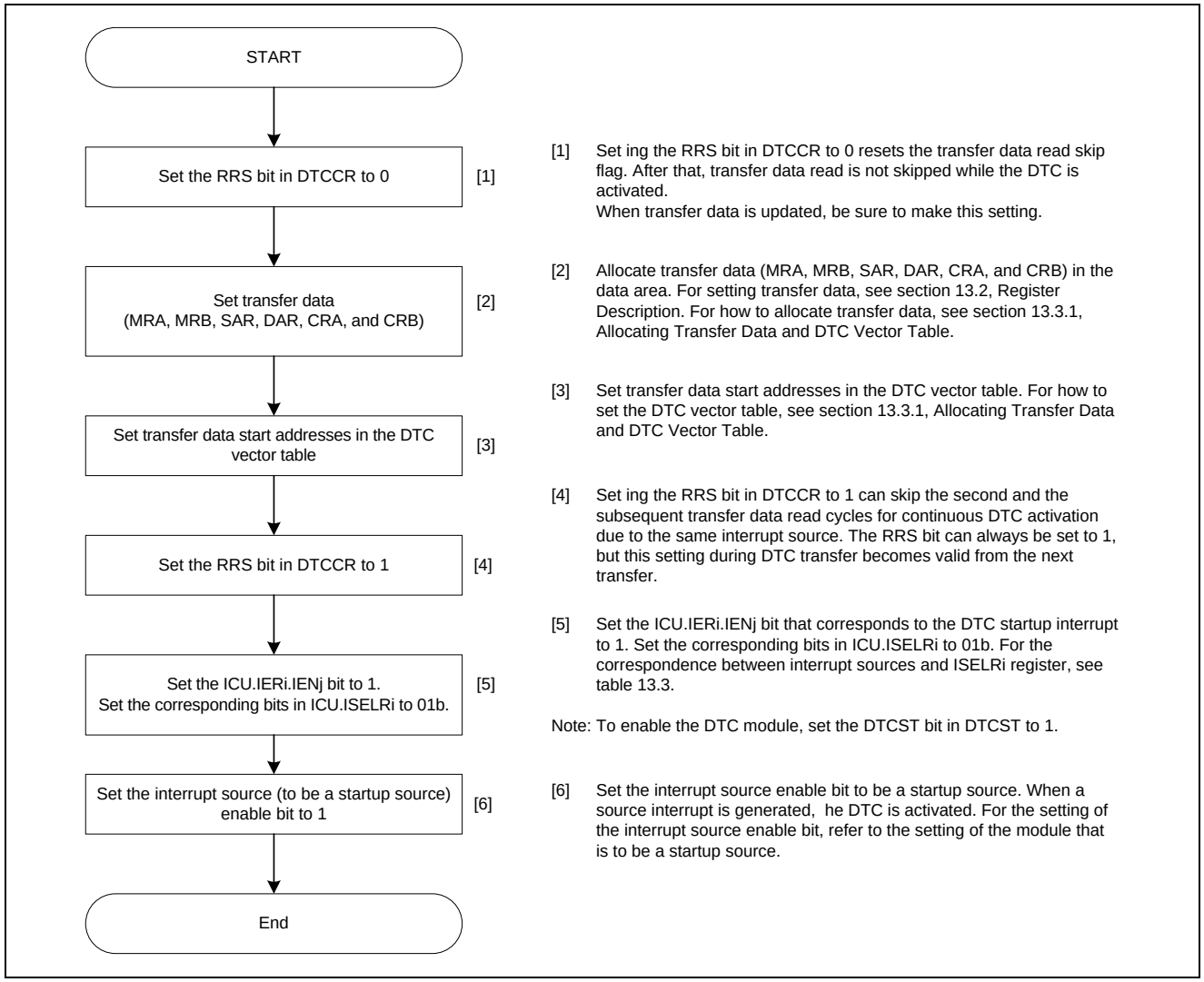



**Figure 13.14 Procedure to Set the DTC**



Downloaded from Arrow.com.

## 13.6 Examples of DTC Usage

### 13.6.1 Normal Transfer

As an example of DTC usage, its employment in the transfer of 128 bytes of data by an SCI is described below.

1. In the MRA register, make the settings to select a fixed source address (MRA.SM[1:0] = 00b), incrementation of the destination address (MRA.DM[1:0] = 10b), transfer in normal mode (MRA.MD[1:0] = 00b), and byte-sized transfer (MRA.SZ[1:0] = 00b). The MRB.DTS bit can be set to any value. For other bits of the MRB register, make the setting for one interrupt to initiate one round of transfer (MRB.CHNE = "0" and MRB.DISEL = "0"). Set the SAR to the address of the RDR for the given SCIm (m = 0 to 6), the DAR to the first address of the area in RAM where data are to be stored, and the CRA register to 128 (0080h). The CRB register can be set to any value.
2. The address where the transfer-control information for use with the RXI starts is set in the vector table for the DTC.
3. Set "01b" in the corresponding ICU.ISELRi register and "1" to the ICU.IERi.IENj bit. Set the DTCST.DTCST bit to "1".
4. Set the SCI for the prescribed reception mode. Enable reception-completed interrupts by setting the SCR.RIE bit in the given SCIm to "1". Also, so that further reception does not proceed if a reception error occurs while reception by the SCI is in progress, make the CPU able to accept reception-error interrupts.
5. Every time the reception of one byte by the SCI is completed, an RXI interrupt is generated to activate the DTC. The DTC transfers the received byte from the RDR of the SCI to RAM, after which the DAR register is incremented and the CRA register is decremented.
6. After 128 rounds of data transfer have been completed and the value in the CRA register becomes "0", an RXI interrupt request is generated for the CPU. Processing for completion is performed in the processing routine for this interrupt.



Downloaded from Arrow.com.

### 13.6.2 Chain Transfer

As an example of chained transfer by the DTC, its employment in the output of pulses by a PPG is described below.

Chained transfer is used to transfer pulse output data and vary the period of the output trigger for the PPG. The first half of the chained transfer is in repeated-transfer mode and the destinations for transfer are the PPGm.NDRH and PPGm.NDRL (where m = 0 or 1) registers. The second half of the chained transfer is in normal-transfer mode and the destinations for transfer are the TPUm.TGRA to TPUm.TGRD registers (where m = 0 to 11). This is because clearing of the activation source and generation of an interrupt on completion of the specified number of rounds of transfer are restricted to the second half of the chained transfer (transfer while MRB.CHNE = "0").

1. Settings are made for transfer to the PPGm.NDRH and PPGm.NDRL registers. In the MRA register, make the settings to select incrementation of the source address (MRA.SM[1:0] = 10b), a fixed destination address (MRA.DM[1:0] = 00b), transfer in repeated-transfer mode (MRA.MD[1:0] = 01b), and word-sized transfer (MRA.SZ[1:0] = 01b). Make the setting for a repeated area on the source side (MRB.DTS = "1"). For other bits of the MRB register, make the settings for chained transfer (MRB.CHNE = "1", MRB.CHNS = "0", and MRB.DISEL = "0"). Set the SAR to the first address of the data table, the DAR register to the address of the PPGm.NDRH register, and the CRAH and CRAL registers to the size of the data table. The CRB register can be set to any value.
2. Settings are made for transfer to the TPUm.TGRA register. In the MRA register, make the settings to select incrementation of the source address (MRA.SM[1:0] = 10b), a fixed destination address (MRA.DM[1:0] = 00b), transfer in normal mode (MRA.MD[1:0] = 00b), and word-sized transfer (MRA.SZ[1:0] = 01b). Set the SAR register to the first address of the data table, the DAR register to the address of the TPUm.TGRA register, and the CRA register to the size of the data table. The CRB register can be set to any value.
3. Place the transfer-control information for use in transfer to the TPU immediately after the transfer-control information for use in transfer to the PPGm.NDRH and PPGm.NDRL registers.
4. In the DTC vector table, set the address where the transfer-control information for use in transfer to the PPGm.NDRH and PPGm.NDRL registers starts.
5. Set "01b" in the corresponding ICU.ISELRi register and "1" to the ICU.IERi.IENj bit. Set the DTCST.DTCST bit to "1".
6. In the given TPUm, set the TIORH and TIORL registers so that the TGRA register operates as an output-compare register (with output disabled) and make the TIER setting to enable TGIA interrupt requests.
7. Set the default output values in the PPGm.NDRH and PPGm.NDRL registers and the next output values in the PPGm.NDRH and PPGm.NDRL registers. In the DDR of the corresponding port m (where m = 0 to 9 or A to E), set the appropriate bit to "1" so that output from the PPGm.NDRH and PPGm.NDRL registers proceeds. Also, select compare-match of the TPU as the output trigger in the PCR of the corresponding port m (where m = 0 to 9 or A to E).
8. Set the TSTRi.CST[5:0] (i = A, B) bits to "1" to start counting operation of the TPU.TCNT counter.
9. Every time a compare-match with the TPUm.TGRA register is generated, next output values are transferred to the PPGm.NDRH and PPGm.NDRL registers and the setting for the next output-trigger period is transferred to the TPUm.TGRA register.
10. After the specified number of rounds of data transfer has been completed (i.e. when the value in the CRA register has become "0"), a TGIA interrupt request is issued for the CPU. Processing for completion is performed in the processing routine for this interrupt.



Downloaded from Arrow.com.

### 13.6.3 Chain Transfer when Counter = 0

The second data transfer is performed only when the counter = 0. Repeat transfer of a transfer count of 256 or more is enabled by the re-setting for the first data transfer.

The following shows an example of configuring a 128-kbyte input buffer, where the input buffer is set so that its lower address starts with 0000h. Figure 13.15 shows a chain transfer when the counter = 0.

1. Set normal transfer mode for input data for the first data transfer. Set the following:
   Transfer source address: Fixed, CRA = 0000h (65,536 times), CHNE bit = 1 (chain transfer enabled) in MRB, CHNS bit = 1 (chain transfer is performed only when the transfer counter is 0) in MRB, and DISEL bit = 0 (an interrupt request to the CPU is generated when specified data transfer is completed) in MRB.
2. Prepare the upper 8-bit address of the start address at every 65,536 times of the transfer destination address for the first data transfer in another area (such as ROM). For example, when setting the input buffer to 200000h to 21FFFFh, prepare 21h and 20h.
3. For the second data transfer, set repeat transfer mode (transfer source: repeat area) for re-setting the transfer destination address of the first data transfer. Specify the upper 8 bits of DAR in the first transfer data area for the transfer destination. At this time, set CHNE bit = 0 (chain transfer disabled) in MRB and DISEL bit = 0 (an interrupt request to the CPU is generated when specified data transfer is completed) in MRB. When setting the input buffer mentioned above to 200000h to 21FFFFh, set the transfer counter to 2.
4. The first data transfer is performed by an interrupt 65,536 times. When the transfer counter of the first data transfer becomes 0, the second data transfer starts. Set the upper 8 bits of the transfer source address of the first data transfer to 21h. The transfer counter (lower 16 bits) of the transfer destination address of the first data transfer is 0000h.
5. In succession, the first data transfer is performed by an interrupt 65,536 times specified for the first data transfer. When the transfer counter of the first data transfer becomes 0, the second data transfer starts. Set the upper 8 bits of the transfer source address of the first data transfer to 20h. The transfer counter (lower 16 bits) of the transfer destination address of the first data transfer is 0000h.
6. Steps 4 and 5 above are repeated infinitely. Since the second data transfer is in repeat transfer mode, no interrupt request to the CPU is generated



Downloaded from Arrow.com.

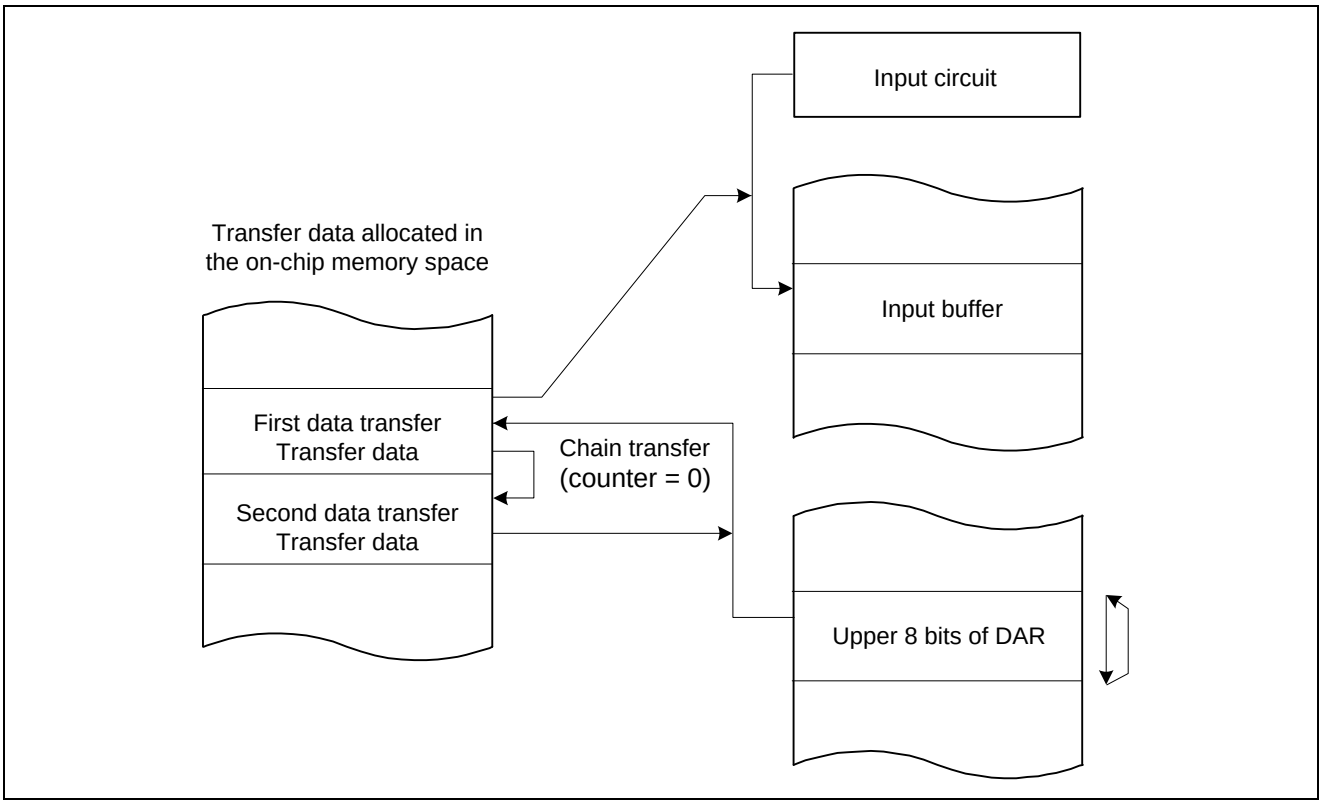



**Figure 13.15 Chain Transfer when Counter = 0**



Downloaded from Arrow.com.

## 13.7 Interrupt Source

When the DTC has finished data transfer of specified count or when data transfer with the DISEL bit in MRB set to 1 (an interrupt request to the CPU is generated each time DTC data transfer is performed) has been completed, an interrupt to the CPU is generated by the DTC startup source. Such interrupts to the CPU are controlled according to the CPU mask level and the priority level of the interrupt control unit.

## 13.8 Low-Power Consumption

### 13.8.1 Setting the DTC Module Start Register

To enable the DTC, the DTCST (DTC module operation) bit in DTCST must be set to 1.

Setting the DTCST (module-stop for the DTC) bit to "0" enables state transitions to module-stop mode, all-module clock-stop mode, software-standby mode, or deep software-standby mode.

#### (1) Module Stop

Writing 1 to the MSTPA27 bit (transition of the DTC to the module-stop state) in MSTPCRA enables the module-stop function for the DTC. If DTC operations are in progress when 1 is written to the bit, the DTC makes a transition to the module-stop state after completing DTC operations.

#### (2) Transition to All-Module Clock-Stop Mode

Writing 1 to the ACSE bit (permitting all-module clock-stop mode) in MSTPCRA, writing 1 to all the bits in MSTPCRA and MSTPCRB, including the MSTPA27 bit (transition of the DTC to the module-stop state), and then executing a WAIT instruction enables transition to the all-module clock-stop mode. However, if DTC operations are in progress when a WAIT instruction is executed, the DTC can make a transition to the all-module clock-stop state after completing DTC operations.

#### (3) Transition to Software Standby and Deep Software Standby Modes

Writing 1 to the SSBY bit (selecting a transition to software standby mode following the execution of a WAIT instruction) in SBYCR and then executing a WAIT instruction places the chip in software standby mode. If transfer by the DTC is in progress at the point of execution of the WAIT instruction, transfer continues until it is completed, after which the transition to software standby mode proceeds. Furthermore, when the setting of the DPSBY bit is "1" (selecting a transition to deep software standby mode after execution of a WAIT instruction) at the time of a transition to software standby mode, the transition will actually be to deep software standby mode.

## 13.9 Usage Notes

### 13.9.1 Transfer Data Start Address/Transfer Source Address/Transfer Destination Address

Be sure to set multiples of 4 for the transfer data start addresses in the vector table. Because, such addresses are accessed with their lowest 2 bits fixed at 00b.



Downloaded from Arrow.com.

### 13.9.2 Allocating Transfer Data

Allocate transfer data in the memory area according to the endian of the area as shown in figure 13.16.

For example, when writing CRA and CRB setting data with 16 bits in big endian, write the CRA setting data to lower address 0 and the CRB setting data to lower address 2. In little endian, write the CRB setting data to lower address 0 and the CRA setting data to lower address 2. When writing CRA and CRB setting data with 32 bits, place the CRA setting data at the MSB side and the CRB setting data at the LSB side regardless of endian, and then write the data to lower address 0.

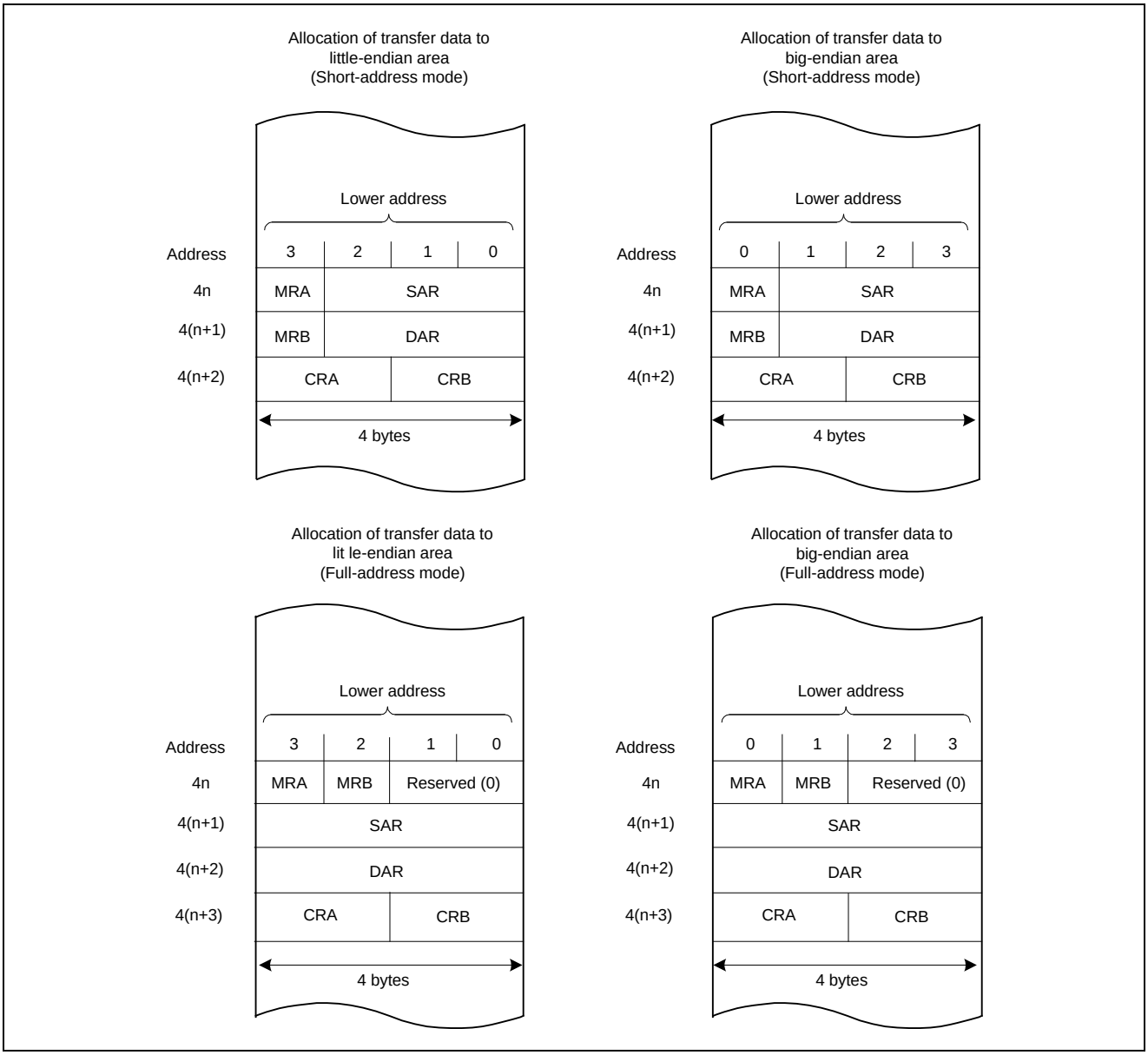



**Figure 13.16 Allocation of Transfer Data**



Downloaded from Arrow.com.

# 14. I/O Ports

The I/O ports of the RX610 Group function as a programmable I/O port, an I/O pin of a peripheral module, an input pin for an interrupt, or a bus control pin.

Each pin is also configurable as an I/O pin of a peripheral module or an input pin for an interrupt. All pins function as input pins immediately after a reset, and pin functions are switched by register settings. The setting of each pin is specified by the registers for the corresponding I/O port and peripheral modules.

Each port has data direction registers (DDR) that control input and output, data registers (DR) that store data for output, port registers (PORT) for reading the pin states, and input buffer control registers (ICR) that enable or disable the input buffer.

The configuration of the I/O ports differs with the package. The 144-pin LQFP version has 15 I/O ports (ports 0 to 9 and A to E), which handle 117 I/O pins. The 176-pin LFBGA version has 18 I/O ports (ports 0 to 9 and A to H), which handle 140 I/O pins.

## 14.1 Overview

Table 14.1 gives the specifications of the I/O ports and table 14.2 lists I/O ports and pin functions.

**Table 14.1 Specifications of I/O Ports**

| Item | | Description |
|---|---|---|
| I/O pins | 144-pin LQFP | 117 |
| | 176-pin LFBGA | 140 |
| Number of ports | 144-pin LQFP | 15 (0 to 9 and A to E) |
| | 176-pin LFBGA | 18 (0 to 9 and A to H) |
| Built-in input pull-up resistor | | Ports A to E |
| Open drain output capability | | Ports 2 and C |
| 5-V tolerance pins | | Ports 0 and 1 (P14, P15, P16, P17) |
| Schmitt trigger input pins | | All port inputs, IRQ inputs, TPU inputs, TMR inputs, RIIC inputs, and SCI inputs |
| Others | | Each pin is capable of driving a capacitive load of 30 pF in the case of a TTL load.<br>When configured as an output, a pin is capable of driving a Darlington transistor |



Downloaded from Arrow.com.

**Table 14.2 Port Functions**

| Port | Description | Bit | Function | | | CMOS Input | Schmitt Trigger Input | Input Pull-up Resistor Function | Open Drain Output Capability |
|---|---|---|---|---|---|---|---|---|---|
| | | | I/O | Input | Output | | | | |
| Port 0 | General I/O port pins, on-chip emulator inputs, interrupt inputs, TMR I/O signals, and SCI I/O signals | 0 | P00 | TMRI2/IRQ8-A | TxD6 | — | All input functions | — | — |
| | | 1 | P01 | TMCI2/RxD6/ IRQ9-A | | | All input functions | | |
| | | 2 | P02/SCK6 | IRQ10-A/TRST# | TMO2 | | All input functions | | |
| | | 3 | P03/SCK4 | TMRI3/IRQ11-A/ TMS | | | All input functions | | |
| | | 4 | P04 | TMCI3/IRQ12-A/ TDI | TxD4 | | All input functions | | |
| | | 5 | P05 | RxD4/IRQ13-A/ TCK | TMO3 | | All input functions | | |
| Port 1 | General I/O port pins, interrupt inputs, TPU inputs, SCI I/O signals, RIIC I/O signals, and A/D converter inputs | 0 | P10 | IRQ0-B | | — | All input functions | — | — |
| | | 1 | P11/SCK2 | IRQ1-B | | | All input functions | | |
| | | 2 | P12 | RxD2/IRQ2-B | | | All input functions | | |
| | | 3 | P13 | ADTRG0#/IRQ3-B | TxD2 | | All input functions | | |
| | | 4 | P14/SDA1 | TCLKA-B/IRQ4-B | | | All input functions | | |
| | | 5 | P15/SCK3/ SCL1 | TCLKB-B/IRQ5-B | | | All input functions | | |
| | | 6 | P16/SDA0 | TCLKC-B/RxD3/ IRQ6-B | | | All input functions | | |
| | | 7 | P17/SCL0 | TCLKD-B/ ADTRG1#/IRQ7-B | TxD3 | | All input functions | | |
| Port 2 | General I/O port pins, TPU I/O signals, PPG outputs, TMR I/O signals, and SCI I/O signals | 0 | P20/TIOCB3 | TIOCA3/TMRI0 | PO0/TxD0 | — | All input functions | — | √ |
| | | 1 | P21/TIOCA3 | TMCI0/RxD0 | PO1 | | All input functions | | |
| | | 2 | P22/TIOCC3/SCK0 | | PO2/TMO0 | | All input functions | | |
| | | 3 | P23/TIOCD3 | TIOCC3 | PO3 | | All input functions | | |
| | | 4 | P24/TIOCB4 | TIOCA4/TMRI1 | PO4 | | All input functions | | |
| | | 5 | P25/TIOCA4 | TMCI1/RxD1 | PO5 | | All input functions | | |
| | | 6 | P26/TIOCA5 | | PO6/TMO1/ TxD1 | | All input functions | | |
| | | 7 | P27/TIOCB5/SCK1 | TIOCA5 | PO7 | | All input functions | | |



Downloaded from Arrow.com.

| Port | Description | Bit | Function | | | CMOS Input | Schmitt Trigger Input | Input Pull-up Resistor Function | Open Drain Output Capability |
|---|---|---|---|---|---|---|---|---|---|
| | | | I/O | Input | Output | | | | |
| Port 3 | General I/O port pins, interrupt inputs, TPU I/O signals, and PPG outputs | 0 | P30/TIOCA0 | IRQ0-A | PO8 | — | All input functions | — | — |
| | | 1 | P31/TIOCB0 | TIOCA0/IRQ1-A | PO9 | | All input functions | | |
| | | 2 | P32/TIOCC0 | TCLKA-A/IRQ2-A | PO10 | | All input functions | | |
| | | 3 | P33/TIOCD0 | TIOCC0/TCLKB-A/ IRQ3-A | PO11 | | All input functions | | |
| | | 4 | P34/TIOCA1 | IRQ4-A | PO12 | | All input functions | | |
| | | 5 | P35/TIOCB1 | TIOCA1/TCLKC-A | PO13 | | All input functions | | |
| | | 6 | P36/TIOCA2 | | PO14 | | All input functions | | |
| | | 7 | P37/TIOCB2 | TIOCA2/TCLKD-A | PO15 | | All input functions | | |
| Port 4 | General I/O port pin, interrupt inputs, and A/D converter inputs | 0 | P40 | AN0/IRQ8-B | | — | P40, IRQ8-B | — | — |
| | | 1 | P41 | AN1/IRQ9-B | | | P41, IRQ9-B | | |
| | | 2 | P42 | AN2/IRQ10-B | | | P42, IRQ10-B | | |
| | | 3 | P43 | AN3/IRQ11-B | | | P43, IRQ11-B | | |
| | | 4 | P44 | AN4/IRQ12-B | | | P44, IRQ12-B | | |
| | | 5 | P45 | AN5/IRQ13-B | | | P45, IRQ13-B | | |
| | | 6 | P46 | AN6/IRQ14-B | | | P46, IRQ14-B | | |
| | | 7 | P47 | AN7/IRQ15-B | | | P47, IRQ15-B | | |
| Port 5 | General I/O port pins, system clock outputs, bus control I/O signals, and tracing I/O signals | 0 | P50 | | WR0#/WR# | — | All input functions | — | — |
| | | 1 | P51 | | WR1#/BC1# | | All input functions | | |
| | | 2 | P52 | | RD# | | All input functions | | |
| | | 3 | | P53 | BCLK | | All input functions | | |
| | | 4 | P54 | | TRDATA0 | | All input functions | | |
| | | 5 | P55 | | TRDATA1 | | All input functions | | |
| | | 6 | P56 | | TRDATA2 | | All input functions | | |
| | | 7 | P57 | WAIT# | TRDATA3 | | All input functions | | |
| Port 6 | General I/O port pins, interrupt inputs, bus control outputs, and D/A converter outputs | 0 | P60 | | CS0#/ CS4#-A/ CS5#-B | — | All input functions | — | — |
| | | 1 | P61 | | CS1#/ CS2#-B/ CS5#-A/ CS6#-B/ CS7#-B | | All input functions | | |
| | | 2 | P62 | | CS2#-A/ CS6#-A | | All input functions | | |
| | | 3 | P63 | | CS3#-A/ CS7#-A | | All input functions | | |
| | | 4 | P64 | | CS4#-B | | All input functions | | |
| | | 5 | P65 | IRQ15-A | | | All input functions | | |
| | | 6 | P66 | | DA0 | | All input functions | | |
| | | 7 | P67 | | DA1 | | All input functions | | |



Downloaded from Arrow.com.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Port 7 | General I/O port pins, interrupt inputs, bus control outputs, and A/D converter inputs | 0 | P70 | ADTRG2# | CS3#-B | — | All input functions | — | — |
| | | 1 | P71 | | CS4#-C/ CS5#-C/ CS6#-C/ CS7#-C | | All input functions | | |
| | | 2 | P72 | | | | All input functions | | |
| | | 3 | P73 | | | | All input functions | | |
| | | 4 | P74 | ADTRG3# | | | All input functions | | |
| | | 5 | P75 | | | | All input functions | | |
| | | 6 | P76 | IRQ14-A | | | All input functions | | |
| | | 7 | P77 | | | | All input functions | | |
| Port 8 | General I/O port pins and tracing outputs | 0 | P80 | | | — | All input functions | — | — |
| | | 1 | P81 | | TRSYNC | | All input functions | | |
| | | 2 | P82 | | TRCLK | | All input functions | | |
| | | 3 | P83 | | | | All input functions | | |
| | | 4 | P84 | | | | All input functions | | |
| | | 5 | P85 | | | | All input functions | | |
| | | 6 | P86 | | | | All input functions | | |
| Port 9 | General I/O port pins and A/D converter inputs | 0 | P90 | AN8 | | — | P90 | — | — |
| | | 1 | P91 | AN9 | | | P91 | | |
| | | 2 | P92 | AN10 | | | P92 | | |
| | | 3 | P93 | AN11 | | | P93 | | |
| | | 4 | P94 | AN12 | | | P94 | | |
| | | 5 | P95 | AN13 | | | P95 | | |
| | | 6 | P96 | AN14 | | | P96 | | |
| | | 7 | P97 | AN15 | | | P97 | | |
| Port A | General I/O port pins, address outputs, TPU I/O signals, and PPG outputs | 0 | PA0/TIOCA6 | | A0/PO16/ BC0# | — | All input functions | √ | — |
| | | 1 | PA1/TIOCB6 | TIOCA6 | A1/PO17 | | All input functions | | |
| | | 2 | PA2/TIOCC6 | TCLKE | A2/PO18 | | All input functions | | |
| | | 3 | PA3/TIOCD6 | TIOCC6/TCLKF | A3/PO19 | | All input functions | | |
| | | 4 | PA4/TIOCA7 | | A4/PO20 | | All input functions | | |
| | | 5 | PA5/TIOCB7 | TIOCA7/TCLKG | A5/PO21 | | All input functions | | |
| | | 6 | PA6/TIOCA8 | | A6/PO22 | | All input functions | | |
| | | 7 | PA7/TIOCB8 | TIOCA8/TCLKH | A7/PO23 | | All input functions | | |



Downloaded from Arrow.com.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Port B | General I/O port pins, address outputs, TPU I/O signals, and PPG outputs | 0 | PB0/TIOCA9 | | A8/PO24 | — | All input functions | √ | — |
| | | 1 | PB1/TIOCB9 | TIOCA9 | A9/PO25 | | All input functions | | |
| | | 2 | PB2/TIOCC9 | | A10/PO26 | | All input functions | | |
| | | 3 | PB3/TIOCD9 | TIOCC9 | A11/PO27 | | All input functions | | |
| | | 4 | PB4/TIOCA10 | | A12/PO28 | | All input functions | | |
| | | 5 | PB5/TIOCB10 | TIOCA10 | A13/PO29 | | All input functions | | |
| | | 6 | PB6/TIOCA11 | | A14/PO30 | | All input functions | | |
| | | 7 | PB7/TIOCB11 | TIOCA11 | A15/PO31 | | All input functions | | |
| Port C | General I/O port pins, address outputs, bus control outputs, and SCI I/O signals | 0 | PC0 | | A16 | — | All input functions | √ | √ |
| | | 1 | PC1 | | A17 | | All input functions | | |
| | | 2 | PC2 | | A18 | | All input functions | | |
| | | 3 | PC3 | | A19 | | All input functions | | |
| | | 4 | PC4 | | A20 | | All input functions | | |
| | | 5 | PC5/SCK5 | | A21/CS5#-D | | All input functions | | |
| | | 6 | PC6 | RxD5 | A22/CS6#-D | | All input functions | | |
| | | 7 | PC7 | | A23/CS4#-D/ CS7#-D/TxD5 | | All input functions | | |
| Port D | General I/O port pins and bidirectional data-bus lines | 0 | PD0/D0 | | | D0 | PD0 | √ | — |
| | | 1 | PD1/D1 | | | D1 | PD1 | | |
| | | 2 | PD2/D2 | | | D2 | PD2 | | |
| | | 3 | PD3/D3 | | | D3 | PD3 | | |
| | | 4 | PD4/D4 | | | D4 | PD4 | | |
| | | 5 | PD5/D5 | | | D5 | PD5 | | |
| | | 6 | PD6/D6 | | | D6 | PD6 | | |
| | | 7 | PD7/D7 | | | D7 | PD7 | | |
| Port E | General I/O port pins, bidirectional data-bus lines, and interrupt inputs | 0 | PE0/D8 | | | D8 | PE0 | √ | — |
| | | 1 | PE1/D9 | | | D9 | PE1 | | |
| | | 2 | PE2/D10 | | | D10 | PE2 | | |
| | | 3 | PE3/D11 | | | D11 | PE3 | | |
| | | 4 | PE4/D12 | | | D12 | PE4 | | |
| | | 5 | PE5/D13 | IRQ5-A | | D13 | PE5, IRQ5-A | | |
| | | 6 | PE6/D14 | IRQ6-A | | D14 | PE6, IRQ6-A | | |
| | | 7 | PE7/D15 | IRQ7-A | | D15 | PE7, IRQ7-A | | |



Downloaded from Arrow.com.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Port F | General I/O port pins | 0 | PF0 | — | All input functions | — | — |
| | | 1 | PF1 | | All input functions | | |
| | | 2 | PF2 | | All input functions | | |
| | | 3 | PF3 | | All input functions | | |
| | | 4 | PF4 | | All input functions | | |
| | | 5 | PF5 | | All input functions | | |
| | | 6 | PF6 | | All input functions | | |
| Port G | General I/O port pins | 0 | PF0 | — | All input functions | — | — |
| | | 1 | PF1 | | All input functions | | |
| | | 2 | PF2 | | All input functions | | |
| | | 3 | PF3 | | All input functions | | |
| | | 4 | PF4 | | All input functions | | |
| | | 5 | PF5 | | All input functions | | |
| | | 6 | PF6 | | All input functions | | |
| | | 7 | PF7 | | All input functions | | |
| Port H | General I/O port pins | 0 | PH0 | — | All input functions | — | — |
| | | 1 | PH1 | | All input functions | | |
| | | 2 | PH2 | | All input functions | | |
| | | 3 | PH3 | | All input functions | | |
| | | 4 | PH4 | | All input functions | | |
| | | 5 | PH5 | | All input functions | | |
| | | 6 | PH6 | | All input functions | | |
| | | 7 | PH7 | | All input functions | | |



Downloaded from Arrow.com.

## 14.2 Register Descriptions

Table 14.3 lists registers of each port.

**Table 14.3 Registers for Each Port**

| Port Symbol | Register Name | Register Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| P0 | Data direction register | DDR | 00h | 0008 C000h | 8 |
| | Data register | DR | 00h | 0008 C020h | 8 |
| | Port register | PORT | Undefined | 0008 C040h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C060h | 8 |
| P1 | Data direction register | DDR | 00h | 0008 C001h | 8 |
| | Data register | DR | 00h | 0008 C021h | 8 |
| | Port register | PORT | Undefined | 0008 C041h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C061h | 8 |
| P2 | Data direction register | DDR | 00h | 0008 C002h | 8 |
| | Data register | DR | 00h | 0008 C022h | 8 |
| | Port register | PORT | Undefined | 0008 C042h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C062h | 8 |
| | Open drain control register | ODR | 00h | 0008 C082h | 8 |
| P3 | Data direction register | DDR | 00h | 0008 C003h | 8 |
| | Data register | DR | 00h | 0008 C023h | 8 |
| | Port register | PORT | Undefined | 0008 C043h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C063h | 8 |
| P4 | Data direction register | DDR | 00h | 0008 C004h | 8 |
| | Data register | DR | 00h | 0008 C024h | 8 |
| | Port register | PORT | Undefined | 0008 C044h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C064h | 8 |
| P5 | Data direction register | DDR | 00h | 0008 C005h | 8 |
| | Data register | DR | 00h | 0008 C025h | 8 |
| | Port register | PORT | Undefined | 0008 C045h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C065h | 8 |
| P6 | Data direction register | DDR | 00h | 0008 C006h | 8 |
| | Data register | DR | 00h | 0008 C026h | 8 |
| | Port register | PORT | Undefined | 0008 C046h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C066h | 8 |
| P7 | Data direction register | DDR | 00h | 0008 C007h | 8 |
| | Data register | DR | 00h | 0008 C027h | 8 |
| | Port register | PORT | Undefined | 0008 C047h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C067h | 8 |
| P8 | Data direction register | DDR | 00h | 0008 C008h | 8 |
| | Data register | DR | 00h | 0008 C028h | 8 |
| | Port register | PORT | Undefined | 0008 C048h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C068h | 8 |



Downloaded from Arrow.com.

| Port Symbol | Register Name | Register Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| P9 | Data direction register | DDR | 00h | 0008 C009h | 8 |
| | Data register | DR | 00h | 0008 C029h | 8 |
| | Port register | PORT | Undefined | 0008 C049h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C069h | 8 |
| PA | Data direction register | DDR | 00h | 0008 C00Ah | 8 |
| | Data register | DR | 00h | 0008 C02Ah | 8 |
| | Port register | PORT | Undefined | 0008 C04Ah | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Ah | 8 |
| | Pull-up resistor control register | PCR | 00h | 0008 C0CAh | 8 |
| PB | Data direction register | DDR | 00h | 0008 C00Bh | 8 |
| | Data register | DR | 00h | 0008 C02Bh | 8 |
| | Port register | PORT | Undefined | 0008 C04Bh | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Bh | 8 |
| | Pull-up resistor control register | PCR | 00h | 0008 C0CBh | 8 |
| PC | Data direction register | DDR | 00h | 0008 C00Ch | 8 |
| | Data register | DR | 00h | 0008 C02Ch | 8 |
| | Port register | PORT | Undefined | 0008 C04Ch | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Ch | 8 |
| | Open drain control register | ODR | 00h | 0008 C08Ch | 8 |
| | Pull-up resistor control register | PCR | 00h | 0008 C0CCh | 8 |
| PD | Data direction register | DDR | 00h | 0008 C00Dh | 8 |
| | Data register | DR | 00h | 0008 C02Dh | 8 |
| | Port register | PORT | Undefined | 0008 C04Dh | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Dh | 8 |
| | Pull-up resistor control register | PCR | 00h | 0008 C0CDh | 8 |
| PE | Data direction register | DDR | 00h | 0008 C00Eh | 8 |
| | Data register | DR | 00h | 0008 C02Eh | 8 |
| | Port register | PORT | Undefined | 0008 C04Eh | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Eh | 8 |
| | Pull-up resistor control register | PCR | 00h | 0008 C0CEh | 8 |
| PF | Data direction register | DDR | 00h | 0008 C00Fh | 8 |
| | Data register | DR | 00h | 0008 C02Fh | 8 |
| | Port register | PORT | Undefined | 0008 C04Fh | 8 |
| | Input buffer control register | ICR | 00h | 0008 C06Fh | 8 |
| PG | Data direction register | DDR | 00h | 0008 C010h | 8 |
| | Data register | DR | 00h | 0008 C030h | 8 |
| | Port register | PORT | Undefined | 0008 C050h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C070h | 8 |
| PH | Data direction register | DDR | 00h | 0008 C011h | 8 |
| | Data register | DR | 00h | 0008 C031h | 8 |
| | Port register | PORT | Undefined | 0008 C051h | 8 |
| | Input buffer control register | ICR | 00h | 0008 C071h | 8 |



Downloaded from Arrow.com.

| Port Symbol | Register Name | Register Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| Common to two or more ports | Port function control register 0 | PFCR0 | 00h | 0008 C100h | 8 |
| | Port function control register 1 | PFCR1 | 00h | 0008 C101h | 8 |
| | Port function control register 2 | PFCR2 | 00h | 0008 C102h | 8 |
| | Port function control register 3 | PFCR3 | 00h | 0008 C103h | 8 |
| | Port function control register 4 | PFCR4 | 00h | 0008 C104h | 8 |
| | Port function control register 5 | PFCR5 | 00h | 0008 C105h | 8 |
| | Port function control register 6 | PFCR6 | 00h | 0008 C106h | 8 |
| | Port function control register 7 | PFCR7 | 00h | 0008 C107h | 8 |
| | Port function control register 8 | PFCR8 | 00h | 0008 C108h | 8 |
| | Port function control register 9 | PFCR9 | 00h | 0008 C109h | 8 |



Downloaded from Arrow.com.

### 14.2.1 Data Direction Register (DDR)

Addresses: P0.DDR 0008 C000h, P1.DDR 0008 C001h, P2.DDR 0008 C002h, P3.DDR 0008 C003h, P4.DDR 0008 C004h, P5.DDR0008 C005h, P6.DDR 0008 C006h, P7.DDR 0008 C007h, P8.DDR 0008 C008h, P9.DDR 0008 C009h, PA.DDR 0008 C00Ah, PB.DDR 0008 C00Bh, PC.DDR 0008 C00Ch, PD.DDR 0008 C00Dh, PE.DDR 0008 C00Eh, PF.DDR 0008 C00Fh, PG.DDR 0008 C010h, PH.DDR 0008 C011h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: The lower six bits are valid and the upper two bits are reserved in P0.DDR.
The lower seven bits are valid and the upper one bit is reserved in P8.DDR.
The lower seven bits are valid and the upper one bit is reserved in PF.DDR.
The reserved bits are read as 0. The write value should be 0.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0 | Pm0 I/O Select (m = 0 to 9 and A to H) | 0: An input pin | R/W |
| b1 | B1 | Pm1 I/O Select | 1: An output pin | R/W |
| b2 | B2 | Pm2 I/O Select | | R/W |
| b3 | B3 | Pm3 I/O Select | | R/W |
| b4 | B4 | Pm4 I/O Select | | R/W |
| b5 | B5 | Pm5 I/O Select | | R/W |
| b6 | B6 | Pm6 I/O Select | | R/W |
| b7 | B7 | Pm7 I/O Select | | R/W |

Each DDR is used to select the input or output direction for individual pins of the corresponding port that have been configured to function as general I/O pins.

Each bit of a Pm.DDR (m = 0 to 9 or A to H) corresponds to a pin of Pm, and the settings can change from bit to bit.



Downloaded from Arrow.com.

### 14.2.2 Data Register (DR)

Addresses: P0.DR 0008 C020h, P1.DR 0008 C021h, P2.DR 0008 C022h, P3.DR 0008 C023h, P4.DR 0008 C024h, P5.DR 0008 C025h, P6.DR 0008 C026h, P7.DR 0008 C027h, P8.DR 0008 C028h, P9.DR 0008 C029h, PA.DR 0008 C02Ah, PB.DR 0008 C02Bh, PC.DR 0008 C02Ch, PD.DR 0008 C02Dh, PE.DR 0008 C02Eh, PF.DR 0008 C02Fh, PG.DR 0008 C030h, PH.DR 0008 C031h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: The lower six bits are valid and the upper two bits are reserved in P0.DR.
The B3 bit in P5.DR is reserved.
The lower seven bits are valid and the upper one bit is reserved in P8.DR.
The lower seven bits are valid and the upper one bit is reserved in PF.DR.
The reserved bits other than the B3 bit in P5.DR are read as 0. The write value should be 0.
The B3 bit in P5.DR is readable and writable.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0 | Pm0 Output Data Store (m = 0 to 9 and A to H) | Output data are stored. | R/W |
| b1 | B1 | Pm1 Output Data Store | | R/W |
| b2 | B2 | Pm2 Output Data Store | | R/W |
| b3 | B3 | Pm3 Output Data Store | | R/W |
| b4 | B4 | Pm4 Output Data Store | | R/W |
| b5 | B5 | Pm5 Output Data Store | | R/W |
| b6 | B6 | Pm6 Output Data Store | | R/W |
| b7 | B7 | Pm7 Output Data Store | | R/W |

Each DR stores the output data from the individual pins of the corresponding port used as a general I/O port.

In addition, the output of the P53 pin is the BCLK signal and the value of the B3 bit in P5.DR does not affect the pin.



Downloaded from Arrow.com.

### 14.2.3 Port Register (PORT)

Addresses: P0.PORT 0008 C040h, P1.PORT 0008 C041h, P2.PORT 0008 C042h, P3.PORT 0008 C043h, P4.PORT 0008 C044h, P5.PORT 0008 C045h, P6.PORT 0008 C046h, P7.PORT 0008 C047h, P8.PORT0008 C048h, P9.PORT 0008 C049h, PA.PORT 0008 C04Ah, PB.PORT 0008 C04Bh, PC.PORT 0008 C04Ch, PD.PORT 0008 C04Dh, PE.PORT 0008 C04Eh, PF.PORT 0008 C04Fh, PG.PORT 0008 C050h, PH.PORT 0008 C051h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | x | x | x | x | x | x | x | x |

Note: The lower six bits are valid and the upper two bits are reserved in P0.PORT.
The lower seven bits are valid and the upper one bit is reserved in P8.PORT.
The lower seven bits are valid and the upper one bit is reserved in PF.PORT.
The reserved bits are read as 0 and cannot be modified.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0 | Pm0 (m = 0 to 9 and A to H) | Individual pin states of the corresponding port are reflected. | R |
| b1 | B1 | Pm1 | | R |
| b2 | B2 | Pm2 | | R |
| b3 | B3 | Pm3 | | R |
| b4 | B4 | Pm4 | | R |
| b5 | B5 | Pm5 | | R |
| b6 | B6 | Pm6 | | R |
| b7 | B7 | Pm7 | | R |

Each PORT reflects individual pin states of the corresponding port.

When a Pm.PORT (m = 0 to 9 and A to H) is read while the value in Pm.DDR is 1, the values of the corresponding bits in Pm.DR are read out. On the other hand, while the value in Pm.DDR is 0, the corresponding pin states are read out to here, regardless of the values in Pm.ICR.



Downloaded from Arrow.com.

### 14.2.4 Input Buffer Control Register (ICR)

Addresses: P0.ICR 0008 C060h, P1.ICR 0008 C061h, P2.ICR 0008 C062h, P3.ICR 0008 C063h, P4.ICR 0008 C064h, P5.ICR 0008 C065h, P6.ICR 0008 C066h, P7.ICR 0008 C067h, P8.ICR 0008 C068h, P9.ICR 0008 C069h, PA.ICR 0008 C06Ah, PB.ICR 0008 C06Bh, PC.ICR 0008 C06Ch, PD.ICR 0008 C06Dh, PE.ICR 0008 C06Eh, PF.ICR 0008 C06Fh, PG.ICR 0008 C070h, PH.ICR 0008 C071h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: The lower six bits are valid and the upper two bits are reserved in P0.ICR.
The lower seven bits are valid and the upper one bit is reserved in P8.ICR.
The lower seven bits are valid and the upper one bit is reserved in PF.ICR.
The reserved bits are read as 0. The write value should be 0.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0* | Pm0 Input Buffer Control (m = 0 to 9 and A to E) | 0: The input buffer for the corresponding pin is disabled, and the input signal is fixed to high level.<br>1: The input buffer for the corresponding pin is enabled. | R/W |
| b1 | B1* | Pm1 Input Buffer Control | | R/W |
| b2 | B2* | Pm2 Input Buffer Control | | R/W |
| b3 | B3* | Pm3 Input Buffer Control | | R/W |
| b4 | B4* | Pm4 Input Buffer Control | | R/W |
| b5 | B5* | Pm5 Input Buffer Control | | R/W |
| b6 | B6* | Pm6 Input Buffer Control | | R/W |
| b7 | B7* | Pm7 Input Buffer Control | | R/W |

Note: * For pins being used as input pins for peripheral modules, set the corresponding bits to 1. Set the bits corresponding to pins that are not being used for their input functions or are being used as analog I/O pins to 0.

Each ICR controls the input buffers for the individual pins of the corresponding port.

Each bit of a Pm.ICR (m = 0 to 9 and A to E) corresponds to a pin of Pm, and the settings can change from bit to bit.

When a Pm.PORT register is read, the pin states of the corresponding port are read out regardless of the values in Pm.ICR. For bits where the value in Pm.ICR is 0, however, the value may not reflect the pin state on the corresponding peripheral module side.

Changes in the settings of a Pm.ICR may generate edges internally, depending on the pin state. For this reason, change the settings of Pm.ICR while the corresponding input pins are not in use. For example, in the case of IRQn (n = 0 to 15) inputs, change settings of the corresponding Pm.ICR with interrupts disabled by clearing the IR flag in IRi (i = 64 to79 ("i" shows an interrupt vector number)) of the interrupt control unit (ICU) to 0, and then enable the corresponding interrupts. If a change to a Pm.ICR setting does generate an edge, negate the edge.



Downloaded from Arrow.com.

### 14.2.5 Pull-Up Resistor Control Register (PCR)

Addresses: PA.PCR 0008 C0CAh, PB.PCR 0008 C0CBh, PC.PCR 0008 C0CCh,
PD.PCR 0008 C0CDh, PE.PCR 0008 C0CEh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0 | Pm0 Input Pull-Up Resistor Control (m = A to E) | 0: Input pull-up resistor is disabled. | R/W |
| b1 | B1 | Pm1 Input Pull-Up Resistor Control | 1: Input pull-up resistor is enabled. | R/W |
| b2 | B2 | Pm2 Input Pull-Up Resistor Control | | R/W |
| b3 | B3 | Pm3 Input Pull-Up Resistor Control | | R/W |
| b4 | B4 | Pm4 Input Pull-Up Resistor Control | | R/W |
| b5 | B5 | Pm5 Input Pull-Up Resistor Control | | R/W |
| b6 | B6 | Pm6 Input Pull-Up Resistor Control | | R/W |
| b7 | B7 | Pm7 Input Pull-Up Resistor Control | | R/W |

Each PCR controls enabling/disabling the input pull-up resistor for individual pins of the corresponding port.

When in input pin state, for the pins corresponding to bits where the value in Pm.PCR is 1, input pull-up resistor is turned on. Table 14.4 summarizes the input pull-up resistor states.

**Table 14.4 Input Pull-Up Resistor States**

| Port | Pin State | Reset | In Other Operations |
|---|---|---|---|
| Port A | Address output | Disabled | |
| | Peripheral module output | Disabled | |
| | Port output | Disabled | |
| | Port input, peripheral module input | Disabled | Enabled/Disabled |
| Port B | Address output | Disabled | |
| | Peripheral module output | Disabled | |
| | Port output | Disabled | |
| | Port input, peripheral module input | Disabled | Enabled/Disabled |
| Port C | Address output | Disabled | |
| | Peripheral module output | Disabled | |
| | Port output | Disabled | |
| | Port input, peripheral module input | Disabled | Enabled/Disabled |
| Port D | Data I/O | Disabled | |
| | Port output | Disabled | |
| | Port input | Disabled | Enabled/Disabled |
| Port E | Data I/O | Disabled | |
| | Port output | Disabled | |
| | Port input, peripheral module input | Disabled | Enabled/Disabled |

[Legend] Disabled: Input pull-up resistor is always disabled.

Enabled/Disabled: Input pull-up resistor is enabled when the Bj bit in Pm.PCR (m = A to E, j = 0 to 7) is set to 1, and disabled when the bit is cleared to 0.



Downloaded from Arrow.com.

### 14.2.6 Open Drain Control Register (ODR)

Addresses: P2.ODR 0008 C082h, PC.ODR 0008 C08Ch

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | B0 | Pm0 Output Type Select (m = 2 and C) | 0: CMOS output pin | R/W |
| b1 | B1 | Pm1 Output Type Select | 1: NMOS open-drain output pin | R/W |
| b2 | B2 | Pm2 Output Type Select | | R/W |
| b3 | B3 | Pm3 Output Type Select | | R/W |
| b4 | B4 | Pm4 Output Type Select | | R/W |
| b5 | B5 | Pm5 Output Type Select | | R/W |
| b6 | B6 | Pm6 Output Type Select | | R/W |
| b7 | B7 | Pm7 Output Type Select | | R/W |

Each ODR is used to select an output type for the individual pins.

### 14.2.7 Port Function Control Register0 (PFCR0)

Address: 0008 C100h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CS7E | CS6E | CS5E | CS4E | CS3E | CS2E | CS1E | CS0E |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | CS0E | CS0 Enable | 0: Designated as an I/O port pin. | R/W |
| b1 | CS1E | CS1 Enable | 1: Designated as the CSn# output pin | R/W |
| b2 | CS2E | CS2 Enable | (n = 0 to 7) | R/W |
| b3 | CS3E | CS3 Enable | | R/W |
| b4 | CS4E | CS4 Enable | | R/W |
| b5 | CS5E | CS5 Enable | | R/W |
| b6 | CS6E | CS6 Enable | | R/W |
| b7 | CS7E | CS7 Enable | | R/W |

PFCR0 enables or disable CSn# output.

**CSnE Bits (CSn Enable) (n = 0 to 7)**

Each bit enables or disables the corresponding CSn# output.

To output a CSn signal, set the corresponding CSnE bit in PFCR0 to 1.



Downloaded from Arrow.com.

## 14.2.8 Port Function Control Register 1 (PFCR1)

Address: 0008 C101h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CS7S[1:0] | | CS6S[1:0] | | CS5S[1:0] | | CS4S[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | CS4S[1:0] | CS4# Output Pin Select | b1 b0 | R/W |
| | | | 0 0: CS4#-A is output from P60. | |
| | | | 0 1: CS4#-B is output from P64. | |
| | | | 1 0: CS4#-C is output from P71. | |
| | | | 1 1: CS4#-D is output from PC7. | |
| b3, b2 | CS5S[1:0] | CS5# Output Pin Select | b3 b2 | R/W |
| | | | 0 0: CS5#-A is output from P61. | |
| | | | 0 1: CS5#-B is output from P60. | |
| | | | 1 0: CS5#-C is output form P71. | |
| | | | 1 1: CS5#-D is output from PC5. | |
| b5, b4 | CS6S[1:0] | CS6# Output Pin Select | b5 b4 | R/W |
| | | | 0 0: CS6#-A is output from P62. | |
| | | | 0 1: CS6#-B is output from P61. | |
| | | | 1 0: CS6#-C is output from P71. | |
| | | | 1 1: CS6#-D is output from PC6. | |
| b7, b6 | CS7S[1:0] | CS7# Output Pin Select | b7 b6 | R/W |
| | | | 0 0: CS7#-A is output from P63. | |
| | | | 0 1: CS7#-B is output from P61. | |
| | | | 1 0: CS7#-C is output from P71. | |
| | | | 1 1: CS7#-D is output from PC7. | |

PFCR1 is used to select a pin for each CSn# output.

PFCR1 can designate output of the several CS signals to a single pin. When that setting is made, all of the designated CS signals are output from the single pin, where the several CS signals are designated. At the time, make the same setting to the external bus interface corresponding to each CS signal, which is output from the same single pin.

**CSnS[1:0] Bits (CSn# Output Pin Select) (n = 4 to 7)**

These bits select a pin for each CSn# output when a CSn# output is enabled (the corresponding CSnE bit in PFCR0 = 1).



Downloaded from Arrow.com.

Figure 14.1 describes the timing for output of CSn# signals for CS5 and CS6 areas to the same pin. Table 14.5 lists the relationship between CS# output pin select registers and output pins.

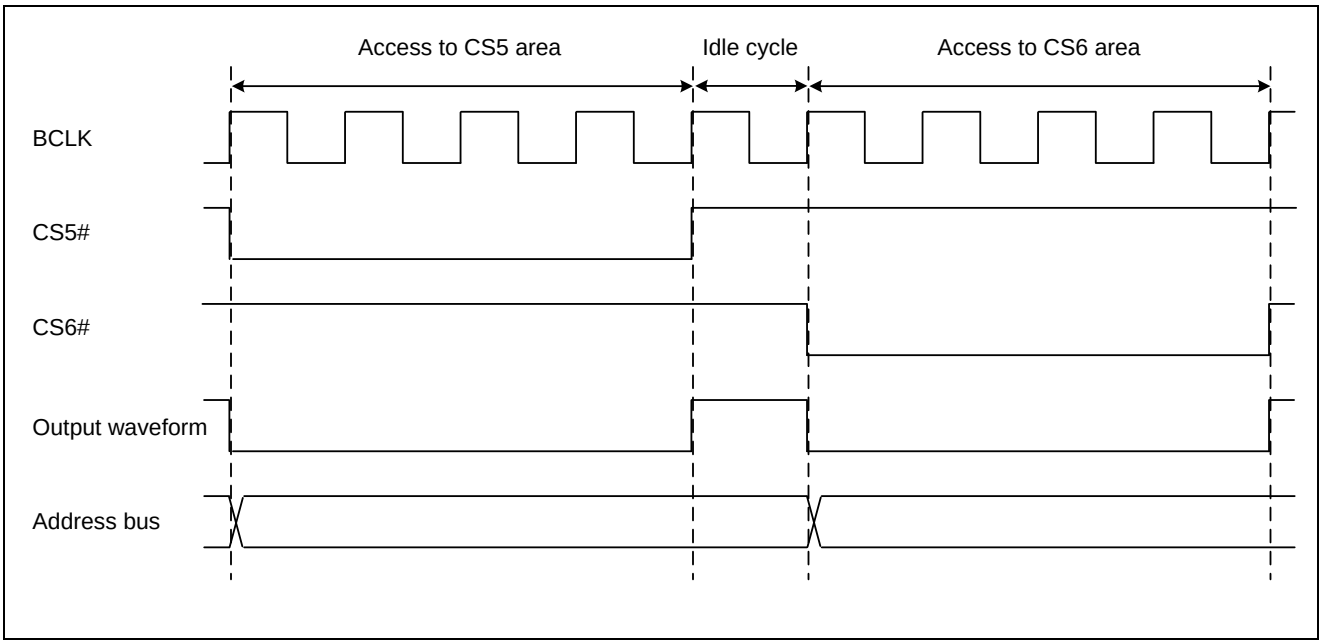



**Figure 14.1 Timing for Output of CSn# Signals to the Same Pin**

**Table 14.5 Relationship between CS# Output Pin Select Registers and Output Pins**

| Output Select | CS0# — | CS1# — | CS2# PFCR2.CS2S | CS3# PFCR2.CS3S | CS4# PFCR1.CS4S[1:0] | CS5# PFCR1.CS5S[1 0] | CS6# PFCR1.CS6S[1:0] | CS7# PFCR1.CS7S[1:0] |
|---|---|---|---|---|---|---|---|---|
| P60 | CS0# | | | | CS4#-A | CS5#-B | | |
| P61 | | CS1# | CS2#-B | | | CS5#-A | CS6#-B | CS7#-B |
| P62 | | | CS2#-A | | | | CS6#-A | |
| P63 | | | | CS3#-A | | | | CS7#-A |
| P64 | | | | | CS4#-B | | | |
| P70 | | | | CS3#-B | | | | |
| P71 | | | | | CS4#-C | CS5#-C | CS6#-C | CS7#-C |
| PC5 | | | | | | CS5#-D | | |
| PC6 | | | | | | | CS6#-D | |
| PC7 | | | | | CS4#-D | | | CS7#-D |



Downloaded from Arrow.com.

## 14.2.9 Port Function Control Register 2 (PFCR2)

Address: 0008 C102h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CS3S | CS2S | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b5 to b0 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b6 | CS2S | CS2# Output Pin Select | 0: CS2#-A is output from P62.<br>1: CS2#-B is output from P61. | R/W |
| b7 | CS3S | CS3# Output Pin Select | 0: CS3#-A is output from P63.<br>1: CS3#-B is output from P70. | R/W |

PFCR2 is used to select a pin for each CSn# output (n = 2 and 3).

**CSnS Bit (CSn# Output Pin Select) (n = 2 and 3)**

Each bit selects a pin for each CSn# output when a CSn# output is enabled (the corresponding CSiE bit in PFCR0 is 1).

Several CS# signals may be output from the same pin if the CS signals have been designated by CSn# output pin select bits (n = 2 and 3). For details, see section 14.2.8, Port Function Control Register 1 (PFCR1).



Downloaded from Arrow.com.

## 14.2.10 Port Function Control Register 3 (PFCR3)

Address: 0008 C103h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | A23E | A22E | A21E | A20E | A19E | A18E | A17E | A16E |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | A16E | Address A16 Enable | 0: A16 output is disabled.<br>1: A16 output is enabled. | R/W |
| b1 | A17E | Address A17 Enable | 0: A17 output is disabled.<br>1: A17 output is enabled. | R/W |
| b2 | A18E | Address A18 Enable | 0: A18 output is disabled.<br>1: A18 output is enabled. | R/W |
| b3 | A19E | Address A19 Enable | 0: A19 output is disabled.<br>1: A19 output is enabled. | R/W |
| b4 | A20E | Address A20 Enable | 0: A20 output is disabled.<br>1: A20 output is enabled. | R/W |
| b5 | A21E | Address A21 Enable | 0: A21 output is disabled.<br>1: A21 output is enabled. | R/W |
| b6 | A22E | Address A22 Enable | 0: A22 output is disabled.<br>1: A22 output is enabled. | R/W |
| b7 | A23E | Address A23 Enable | 0: A23 output is disabled.<br>1: A23 output is enabled. | R/W |

PFCR3 enables or disables address outputs.

**AnE Bit (Address An Enable) (n = 16 to 23)**

Each bit enables or disables an address output (An).



Downloaded from Arrow.com.

## 14.2.11 Port Function Control Register 4 (PFCR4)

Address: 0008 C104h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | A15E | A14E | A13E | A12E | A11E | A10E | A9E | A8E |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | A8E | Address A8 Enable | 0: A8 output is disabled.<br>1: A8 output is enabled. | R/W |
| b1 | A9E | Address A9 Enable | 0: A9 output is disabled.<br>1: A9 output is enabled. | R/W |
| b2 | A10E | Address A10 Enable | 0: A10 output is disabled.<br>1: A10 output is enabled. | R/W |
| b3 | A11E | Address A11 Enable | 0: A11 output is disabled.<br>1: A11 output is enabled. | R/W |
| b4 | A12E | Address A12 Enable | 0: A12 output is disabled.<br>1: A12 output is enabled. | R/W |
| b5 | A13E | Address A13 Enable | 0: A13 output is disabled.<br>1: A13 output is enabled. | R/W |
| b6 | A14E | Address A14 Enable | 0: A14 output is disabled.<br>1: A14 output is enabled. | R/W |
| b7 | A15E | Address A15 Enable | 0: A15 output is disabled.<br>1: A15 output is enabled. | R/W |

PFCR4 enables or disables address outputs.

**AnE Bit (Address An Enable) (n =8 to 15)**

Each bit enables or disables an address output (An).



Downloaded from Arrow.com.

## 14.2.12 Port Function Control Register 5 (PFCR5)

Address: 0008 C105h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | WR1BC1E | – | DHE | TCLKS | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b3 | TCLKS | TPU External Clock Input Pin Select | 0: P32, P33, P35, and P37 are designated as external clock input pins.<br>1: P14 to P17 are designated as external clock input pins. | R/W |
| b4 | DHE | D15-to-D8 Enable | 0: PE7 to PE0 are designated as I/O port pins.<br>1: PE7 to PE0 are designated as D15 to D8 pins (function as part of the external data bus) | R/W |
| b5 | — | Reserved | This bit is read as 0. The write value should be 0. | R/W |
| b6 | WR1BC1E | WR1#/BC1# Output Enable | 0: P51 is designated as an I/O port pin.<br>1: P51 is designated as the WR#1 or BC1# output pin. | R/W |
| b7 | — | Reserved | This bit is read as 0. The write value should be 0. | R/W |

PFCR5 is used to select external clock input pins for the TPU.

### TCLKS Bit (TPU External Clock Input Pin Select)

This bit selects the external clock input pins for the TPU.

### DHE Bit (D15-to-D8 Enable)

This bit enables or disables the input and output of data signals D15 to D8 on the PE7 to PE0 pins (valid in expansion mode with on-chip ROM disabled or enabled).

Note: This setting must be in accord with the setting of the external bus width selection bits in the CSn control register (CSnCNT.BSIZE[1:0]). If the value of the DHE bit is 0 but the width of the external bus is 16 bits, operation of the port E pin functions will be disrupted. For details on the BSIZE[1:0] bits, see 11.3.1, CSn Control Register (CSnCNT).

### WR1BC1E Bit (WR1#/BC1# Output Enable)

This bit enables or disables WR1#/BC1# output (valid in expansion mode with on-chip ROM disabled or enabled).



Downloaded from Arrow.com.

## 14.2.13 Port Function Control Register 6 (PFCR6)

Address: 0008 C106h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TPUMS5 | TPUMS4 | TPUMS3A | TPUMS3B | TPUMS2 | TPUMS1 | TPUMS0A | TPUMS0B |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TPUMS0B | Multifunction Select 0B for TPU I/O Pins | 0: Output compare and input capture are allocated to P32.<br>1: Input capture and output compare are allocated to P33 and P32, respectively. | R/W |
| b1 | TPUMS0A | Multifunction Select 0A for TPU I/O Pins | 0: Output compare and input capture are allocated to P30.<br>1: Input capture and output compare are allocated to P31 and P30, respectively. | R/W |
| b2 | TPUMS1 | Multifunction Select 1 for TPU I/O Pins | 0: Output compare and input capture are allocated to P34.<br>1: Input capture and output compare are allocated to P35 and P34, respectively. | R/W |
| b3 | TPUMS2 | Multifunction Select 2 for TPU I/O Pins | 0: Output compare and input capture are allocated to P36.<br>1: Input capture and output compare are allocated to P37and P36, respectively. | R/W |
| b4 | TPUMS3B | Multifunction Select 3B for TPU I/O Pins | 0: Output compare and input capture are allocated to P22.<br>1: Input capture and output compare are allocated to P23 and P22, respectively. | R/W |
| b5 | TPUMS3A | Multifunction Select 3A for TPU I/O Pins | 0: Output compare and input capture are allocated to P21.<br>1: Input capture and output compare are allocated to P20 and P21, respectively. | R/W |
| b6 | TPUMS4 | Multifunction Select 4 for TPU I/O Pins | 0: Output compare and input capture are allocated to P25.<br>1: Input capture and output compare are allocated to P24 and P25, respectively. | R/W |
| b7 | TPUMS5 | Multifunction Select 5 for TPU I/O Pins | 0: Output compare and input capture are allocated to P26.<br>1: Input capture and output compare are allocated to P27 and P26, respectively. | R/W |

PFCR6 is used to select the multiplexed function for the TPU (unit 0) I/O pins.

Setting the TPUMS0A, TPUMS1, TPUMS2, TPUMS3A, PUTMS4, and TPUMS5 bits to 1 enables the input capture inputs of TGRA and TGRB of the TPUn to be allocated to the same pin. Setting the TPUMS0B and TPUMS3B bits to 1 enables the input capture inputs of TGRC and TGRD of the TPUn to be allocated to the same pin.

By setting the timer mode register of the TPU (TPUn.TMDR), it is possible to set the input capture inputs of TRGA and TRGB or TRGC and TRGD to the same pin.

Table 14.6 shows the correspondences between the PFCR6 and TPUn.TMDR settings and the input capture inputs and external pins.

**TPUMS0B Bit (Multifunction Select 0B for TPU I/O Pins)**

This bit selects an input pin for TIOCC0.



Downloaded from Arrow.com.

### TPUMS0A Bit (Multifunction Select 0A for TPU I/O Pins)

This bit selects an input pin for TIOCA0.

### TPUMS1 Bit (Multifunction Select 1 for TPU I/O Pins)

This bit selects an input pin for TIOCA1.

### TPUMS2 Bit (Multifunction Select 2 for TPU I/O Pins)

This bit selects an input pin for TIOCA2.

### TPUMS3B Bit (Multifunction Select 3B for TPU I/O Pins)

This bit selects an input pin for TIOCC3.

### TPUMS3A Bit (Multifunction Select 3A for TPU I/O Pins)

This bit selects an input pin for TIOCA3.

### TPUMS4 Bit (Multifunction Select 4 for TPU I/O Pins)

This bit selects an input pin for TIOCA4.

### TPUMS5 Bit (Multifunction Select 5 for TPU I/O Pins)

This bit selects a multiplexed function for TIOCA5.

**Table 14.6 Correspondences between PFCR6 and TPUn.TMDR Settings, Input Capture Inputs, and External Pins**

| | | TPU0.TGRC | | TPU0.TGRD | |
|---|---|---|---|---|---|
| TPU0.TMDR. ICSELD | PFCR6. TPUMS0B | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCC0 | P32 | TIOCD0 | P33 |
| 0 | 1 | | P33 | TIOCD0 | P33 |
| 1 | 0 | | P32 | TIOCC0 | P32 |
| 1 | 1 | | P33 | TIOCC0 | P33 |

| | | TPU0.TGRA | | TPU0.TGRB | |
|---|---|---|---|---|---|
| TPU0.TMDR. ICSELB | PFCR6. TPUMS0A | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA0 | P30 | TIOCB0 | P31 |
| 0 | 1 | | P31 | TIOCB0 | P31 |
| 1 | 0 | | P30 | TIOCA0 | P30 |
| 1 | 1 | | P31 | TIOCA0 | P31 |

| | | TPU1.TGRA | | TPU1.TGRB | |
|---|---|---|---|---|---|
| TPU1.TMDR. ICSELB | PFCR6. TPUMS1 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA1 | P34 | TIOCB1 | P35 |
| 0 | 1 | | P35 | TIOCB1 | P35 |
| 1 | 0 | | P34 | TIOCA1 | P34 |
| 1 | 1 | | P35 | TIOCA1 | P35 |



Downloaded from Arrow.com.

| | | TPU2.TGRA | | TPU2.TGRB | |
|---|---|---|---|---|---|
| TPU2.TMDR. ICSELB | PFCR6. TPUMS2 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | | P36 | TIOCB2 | P37 |
| 0 | 1 | | P37 | TIOCB2 | P37 |
| 1 | 0 | | P36 | TIOCA2 | P36 |
| 1 | 1 | TIOCA2 | P37 | TIOCA2 | P37 |

| | | TPU3.TGRC | | TPU3.TGRD | |
|---|---|---|---|---|---|
| TPU3.TMDR. ICSELD | PFCR6. TPUMS3B | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCC3 | P22 | TIOCD3 | P23 |
| 0 | 1 | | P23 | TIOCD3 | P23 |
| 1 | 0 | | P22 | TIOCC3 | P22 |
| 1 | 1 | | P23 | TIOCC3 | P23 |

| | | TPU3.TGRA | | TPU3.TGRB | |
|---|---|---|---|---|---|
| TPU3.TMDR. ICSELB | PFCR6. TPUMS3A | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA3 | P21 | TIOCB3 | P20 |
| 0 | 1 | | P20 | TIOCB3 | P20 |
| 1 | 0 | | P21 | TIOCA3 | P21 |
| 1 | 1 | | P20 | TIOCA3 | P20 |

| | | TPU4.TGRA | | TPU4.TGRB | |
|---|---|---|---|---|---|
| TPU4.TMDR. ICSELB | PFCR6. TPUMS4 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA4 | P25 | TIOCB4 | P24 |
| 0 | 1 | | P24 | TIOCB4 | P24 |
| 1 | 0 | | P25 | TIOCA4 | P25 |
| 1 | 1 | | P24 | TIOCA4 | P24 |

| | | TPU5.TGRA | | TPU5.TGRB | |
|---|---|---|---|---|---|
| TPU5.TMDR. ICSELB | PFCR6. TPUMS5 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA5 | P26 | TIOCB5 | P27 |
| 0 | 1 | | P27 | TIOCB5 | P27 |
| 1 | 0 | | P26 | TIOCA5 | P26 |
| 1 | 1 | | P27 | TIOCA5 | P27 |



Downloaded from Arrow.com.

## 14.2.14 Port Function Control Register 7 (PFCR7)

Address: 0008 C107h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TPUMS11 | TPUMS10 | TPUMS9A | TPUMS9B | TPUMS8 | TPUMS7 | TPUMS6A | TPUMS6B |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TPUMS6B | Multifunction Select 6B for TPU I/O Pins | 0: Output compare and input capture are allocated to PA2.<br>1: Input capture and output compare are allocated to PA3 and PA2, respectively. | R/W |
| b1 | TPUMS6A | Multifunction Select 6A for TPU I/O Pins | 0: Output compare and input capture are allocated to PA0.<br>1: Input capture and output compare are allocated to PA1 and PA0, respectively. | R/W |
| b2 | TPUMS7 | Multifunction Select 7 for TPU I/O Pins | 0: Output compare and input capture are allocated to PA4.<br>1: Input capture and output compare are allocated to PA5 and PA4, respectively. | R/W |
| b3 | TPUMS8 | Multifunction Select 8 for TPU I/O Pins | 0: Output compare and input capture are allocated to PA6.<br>1: Input capture and output compare are allocated to PA7 and PA6, respectively. | R/W |
| b4 | TPUMS9B | Multifunction Select 9B for TPU I/O Pins | 0: Output compare and input capture are allocated to PB2.<br>1: Input capture and output compare are allocated to PB3 and PB2, respectively. | R/W |
| b5 | TPUMS9A | Multifunction Select 9A for TPU I/O Pins | 0: Output compare and input capture are allocated to PB0.<br>1: Input capture and output compare are allocated to PB1 and PB0, respectively. | R/W |
| b6 | TPUMS10 | Multifunction Select 10 for TPU I/O Pins | 0: Output compare and input capture are allocated to PB4.<br>1: Input capture and output compare are allocated to PB5 and PB4, respectively. | R/W |
| b7 | TPUMS11 | Multifunction Select 11 for TPU I/O Pins | 0: Output compare and input capture are allocated to PB6.<br>1: Input capture and output compare are allocated to PB7 and PB6, respectively. | R/W |

PFCR7 is used to select the multiplexed function for TPU (unit 1) I/O Pins.

Setting the TPUMS6A, TPUMS7, TPUMS8, TPUMS9A, PUTMS10, and TPUMS11 bits to 1 enables the input capture inputs of TGRA and TGRB of the TPUn to be allocated to the same pin. Setting the TPUMS6B and TPUMS9B bits to 1 enables the input capture inputs of TGRC and TGRD of the TPUn to be allocated to the same pin.

By setting the timer mode register of the TPU (TPUn.TMDR), it is possible to set the input capture inputs of TRGA and TRGB or TRGC and TRGD to the same pin.

Table 14.7 shows the correspondences between the PFCR7 and TPUn.TMDR settings and the input capture inputs and external pins.

**TPUMS6B Bit (Multifunction Select 6B for TPU I/O Pins)**

This bit selects an input pin for TIOCC6.



Downloaded from Arrow.com.

### TPUMS6A Bit (Multifunction Select 6A for TPU I/O Pins)

This bit selects an input pin for TIOCA6.

### TPUMS7 Bit (Multifunction Select 7 for TPU I/O Pins)

This bit selects an input pin for TIOCA7.

### TPUMS8 Bit (Multifunction Select 8 for TPU I/O Pins)

This bit selects an input pin for TIOCA8.

### TPUMS9B Bit (Multifunction Select 9B for TPU I/O Pins)

This bit selects an input pin for TIOCC9.

### TPUMS9A Bit (Multifunction Select 9A for TPU I/O Pins)

This bit selects an input pin for TIOCA9.

### TPUMS10 Bit (Multifunction Select 10 for TPU I/O Pins)

This bit selects an input pin for TIOCA10.

### TPUMS11 Bit (Multifunction Select 11 for TPU I/O Pins TPU)

This bit selects an input pin for TIOCA11.

**Table 14.7 Correspondences between PFCR7 and TPUn.TMDR Settings and Input Capture Inputs and External Pins**

| | | TPU6.TGRC | | TPU6.TGRD | |
|---|---|---|---|---|---|
| TPU6.TMDR. ICSELD | PFCR7. TPUMS6B | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCC6 | PA2 | TIOCD6 | PA3 |
| 0 | 1 | | PA3 | TIOCD6 | PA3 |
| 1 | 0 | | PA2 | TIOCC6 | PA2 |
| 1 | 1 | | PA3 | TIOCC6 | PA3 |

| | | TPU6.TGRA | | TPU6.TGRB | |
|---|---|---|---|---|---|
| TPU6.TMDR. ICSELB | PFCR7. TPUMS6A | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA6 | PA0 | TIOCB6 | PA1 |
| 0 | 1 | | PA1 | TIOCB6 | PA1 |
| 1 | 0 | | PA0 | TIOCA6 | PA0 |
| 1 | 1 | | PA1 | TIOCA6 | PA1 |

| | | TPU7.TGRA | | TPU7.TGRB | |
|---|---|---|---|---|---|
| TPU7.TMDR. ICSELB | PFCR7. TPUMS7 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA7 | PA4 | TIOCB7 | PA5 |
| 0 | 1 | | PA5 | TIOCB7 | PA5 |
| 1 | 0 | | PA4 | TIOCA7 | PA4 |
| 1 | 1 | | PA5 | TIOCA7 | PA5 |



Downloaded from Arrow.com.

| | | TPU8.TGRA | | TPU8.TGRB | |
|---|---|---|---|---|---|
| TPU8.TMDR. ICSELB | PFCR7. TPUMS8 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA8 | PA6 | TIOCB8 | PA7 |
| 0 | 1 | | PA7 | TIOCB8 | PA7 |
| 1 | 0 | | PA6 | TIOCA8 | PA6 |
| 1 | 1 | | PA7 | TIOCA8 | PA7 |

| | | TPU9.TGRC | | TPU9.TGRD | |
|---|---|---|---|---|---|
| TPU9.TMDR. ICSELD | PFCR7. TPUMS9B | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCC9 | PB2 | TIOCD9 | PB3 |
| 0 | 1 | | PB3 | TIOCD9 | PB3 |
| 1 | 0 | | PB2 | TIOCC9 | PB2 |
| 1 | 1 | | PB3 | TIOCC9 | PB3 |

| | | TPU9.TGRA | | TPU9.TGRB | |
|---|---|---|---|---|---|
| TPU9.TMDR. ICSELB | PFCR7. TPUMS9A | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA9 | PB0 | TIOCB9 | PB1 |
| 0 | 1 | | PB1 | TIOCB9 | PB1 |
| 1 | 0 | | PB0 | TIOCA9 | PB0 |
| 1 | 1 | | PB1 | TIOCA9 | PB1 |

| | | TPU10.TGRA | | TPU10.TGRB | |
|---|---|---|---|---|---|
| TPU10.TMDR. ICSELB | PFCR7. TPUMS10 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA10 | PB4 | TIOCB10 | PB5 |
| 0 | 1 | | PB5 | TIOCB10 | PB5 |
| 1 | 0 | | PB4 | TIOCA10 | PB4 |
| 1 | 1 | | PB5 | TIOCA10 | PB5 |

| | | TPU11.TGRA | | TPU11.TGRB | |
|---|---|---|---|---|---|
| TPU11.TMDR. ICSELB | PFCR7. TPUMS11 | Input Capture Input | External Pin | Input Capture Input | External Pin |
| 0 | 0 | TIOCA11 | PB6 | TIOCB11 | PB7 |
| 0 | 1 | | PB7 | TIOCB11 | PB7 |
| 1 | 0 | | PB6 | TIOCA11 | PB6 |
| 1 | 1 | | PB7 | TIOCA11 | PB7 |



Downloaded from Arrow.com.

## 14.2.15 Port Function Control Register 8 (PFCR8)

Address: 0008 C108h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ITS15 | ITS14 | ITS13 | ITS12 | ITS11 | ITS10 | ITS9 | ITS8 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ITS8 | IRQ8 Pin Select | 0: P00 is designated as the IRQ8-A input pin.<br>1: P40 is designated as the IRQ8-B input pin. | R/W |
| b1 | ITS9 | IRQ9 Pin Select | 0: P01 is designated as IRQ9-A input pin.<br>1: P41 is designated as IRQ9-B input pin. | R/W |
| b2 | ITS10 | IRQ10 Pin Select | 0: P02 is designated as the IRQ10-A input pin.<br>1: P42 is designated as the IRQ10-B input pin. | R/W |
| b3 | ITS11 | IRQ11 Pin Select | 0: P03 is designated as the IRQ11-A input pin.<br>1: P43 is designated as the IRQ11-B input pin. | R/W |
| b4 | ITS12 | IRQ12 Pin Select | 0: P04 is designated as the IRQ12-A input pin.<br>1: P44 is designated as the IRQ12-B input pin. | R/W |
| b5 | ITS13 | IRQ13 Pin Select | 0: P05 is designated as the IRQ13-A input pin.<br>1: P45 is designated as the IRQ13-B input pin. | R/W |
| b6 | ITS14 | IRQ14 Pin Select | 0: P76 is designated as the IRQ14-A input pin.<br>1: P46 is designated as the IRQ14-B input pin. | R/W |
| b7 | ITS15 | IRQ15 Pin Select | 0: P65 is designated as the IRQ15-A input pin.<br>1: P47 is designated as the IRQ15-B input pin. | R/W |

PFCR8 is used to select pins for IRQ8 to IRQ15 inputs.

**ITSn (IRQn Pin Select) (n = 8 to 15)**

Each bit selects a pin for an IRQn input.



Downloaded from Arrow.com.

## 14.2.16 Port Function Control Register 9 (PFCR9)

Address: 0008 C109h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ITS7 | ITS6 | ITS5 | ITS4 | ITS3 | ITS2 | ITS1 | ITS0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ITS0 | IRQ0 Pin Select | 0: P30 is designated as the IRQ0-A input pin.<br>1: P10 is designated as the IRQ0-B input pin. | R/W |
| b1 | ITS1 | IRQ1 Pin Select | 0: P31 is designated as the IRQ1-A input pin.<br>1: P11 is designated as the IRQ1-B input pin. | R/W |
| b2 | ITS2 | IRQ2 Pin Select | 0: P32 is designated as the IRQ2-A input pin.<br>1: P12 is designated as the IRQ2-B input pin. | R/W |
| b3 | ITS3 | IRQ3 Pin Select | 0: P33 is designated as the IRQ3-A input pin.<br>1: P13 is designated as the IRQ3-B input pin. | R/W |
| b4 | ITS4 | IRQ4 Pin Select | 0: P34 is designated as the IRQ4-A input pin.<br>1: P14 is designated as the IRQ4-B input pin. | R/W |
| b5 | ITS5 | IRQ5 Pin Select | 0: PE5 is designated as the IRQ5-A input pin.<br>1: P15 is designated as the IRQ5-B input pin. | R/W |
| b6 | ITS6 | IRQ6 Pin Select | 0: PE6 is designated as the IRQ6-A input pin.<br>1: P16 is designated as the IRQ6-B input pin. | R/W |
| b7 | ITS7 | IRQ7 Pin Select | 0: PE7 is designated as the IRQ7-A input pin.<br>1: P17 is designated as the IRQ7-B input pin. | R/W |

PFCR9 is used to select pins for IRQ0 to IRQ7 inputs.

**ITSn (IRQn Pin Select) (n= 0 to7)**

Each bit selects a pin for an IRQn input.



Downloaded from Arrow.com.

## 14.3 Settings of Ports

Individual pins for peripheral modules are indicated by "_OE" appended to the end of the pin name (for example, TIOCA4_OE). In addition, the settings to enable desired outputs and the other settings are indicated as 1 and 0 in the following tables, respectively.

Table 14.8 lists the settings to enable output of the signals from the individual pins of each port. For details on the applicable output signals, see descriptions of the registers for each peripheral module.

Setting the port function control register m (PFCRm) changes the functions of peripheral-module pins with names ending in A to D. For pins functioning as inputs, the pin names are placed in parentheses: " ( ) ". "—" in the following tables means "Don't care".

Input from a pin to a peripheral module is enabled by setting the corresponding bit in the input buffer control register (Pm.ICR) to 1*. It is necessary to set the peripheral module to use the enabled input function. For the peripheral module settings to use the input function, see the section for the peripheral module.

Note: * When the input function is allocated to two or more external pins, PFCRm as well as Pm.ICR should be set to select an input pin.

### 14.3.1 Port 0 (P0)

#### (1) P00/(TMRI2)/TxD6/(IRQ8-A)

The pin function is switched as shown below according to the combination of the register setting for the SCI and the B0 bit in P0.DDR.

| | | Setting | |
|---|---|---|---|
| | | SCI | I/O Port |
| Module Name | Pin Function | TxD6_OE | P0.DDR.B0 |
| SCI | TxD6 output | 1 | — |
| I/O port | P00 output | 0 | 1 |
| | P00 input (initial value) | 0 | 0 |

#### (2) P01/(TMCI2)/(RxD6)/(IRQ9-A)

The pin function is switched as shown below according to the value of the B1 bit in P0.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| Module Name | Pin Function | P0.DDR.B1 |
| I/O port | P01 output | 1 |
| | P01 input (initial value) | 0 |



Downloaded from Arrow.com.

### (3) P02/TMO2/SCK6/(IRQ10-A)/(TRST#)

The pin function is switched as shown below according to the combination of the register settings for the TMR and SCI and the B2 bit in P0.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TMR | SCI | I/O Port |
| Module Name | Pin Function | TMO2_OE | SCK6_OE | P0.DDR.B2 |
| TMR | TMO2 output | 1 | — | — |
| SCI | SCK6 output | 0 | 1 | — |
| I/O port | P02 output | 0 | 0 | 1 |
| | P02 input (initial value) | 0 | 0 | 0 |

### (4) P03/(TMRI3)/SCK4/(IRQ11-A)/(TMS)

The pin function is switched as shown below according to the combination of the register setting for the SCI and the B3 bit in P0.DDR.

| | | Setting | |
|---|---|---|---|
| | | SCI | I/O Port |
| Module Name | Pin Function | SCK4_OE | P0.DDR.B3 |
| SCI | SCK4 output | 1 | — |
| I/O port | P03 output | 0 | 1 |
| | P03 input (initial value) | 0 | 0 |

### (5) P04/(TMCI3)/TxD4/(IRQ12-A)/(TDI)

The pin function is switched as shown below according to the combination of the register setting for the SCI and the B4 bit in P0.DDR.

| | | Setting | |
|---|---|---|---|
| | | SCI | I/O Port |
| Module Name | Pin Function | TxD4_OE | P0.DDR.B4 |
| SCI | TxD4 output | 1 | — |
| I/O port | P04 output | 0 | 1 |
| | P04 input (initial value) | 0 | 0 |

### (6) P05/TMO3/(RxD4)/(IRQ13-A)/(TCK)

The pin function is switched as shown below according to the combination of the register setting for the TMR and the B5 bit in P0.DDR.

| | | Setting | |
|---|---|---|---|
| | | TMR | I/O Port |
| Module Name | Pin Function | TMO3_OE | P0.DDR.B5 |
| TMR | TMO3 output | 1 | — |
| I/O port | P05 output | 0 | 1 |
| | P05 input (initial value) | 0 | 0 |



Downloaded from Arrow.com.

## 14.3.2 Port 1 (P1)

### (1) P10/(IRQ0-B)

The pin function is switched as shown below according to the value of the B0 bit in P1.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| Module Name | Pin Function | P1.DDR.B0 |
| I/O port | P10 output | 1 |
| | P10 input (initial value) | 0 |

### (2) P11/SCK2/(IRQ1-B)

The pin function is switched as shown below according to the combination of the register setting for the SCI and the B1 bit in P1.DDR.

| | | Setting | |
|---|---|---|---|
| | | SCI | I/O Port |
| Module Name | Pin Function | SCK2_OE | P1.DDR.B1 |
| SCI | SCK2 output | 1 | — |
| I/O port | P11 output | 0 | 1 |
| | P11 input (initial value) | 0 | 0 |

### (3) P12/(RxD2)/(IRQ2-B)

The pin function is switched as shown below according to the value of the B2 bit in P1.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| Module Name | Pin Function | P1.DDR.B2 |
| I/O port | P12 output | 1 |
| | P12 input (initial value) | 0 |

### (4) P13/TxD2/(ADTRG0#)/(IRQ3-B)

The pin function is switched as shown below according to the combination of the register setting for the SCI and the B3 bit in P1.DDR.

| | | Setting | |
|---|---|---|---|
| | | SCI | I/O Port |
| Module Name | Pin Function | TxD2_OE | P1.DDR.B3 |
| SCI | TxD2 output | 1 | — |
| I/O port | P13 output | 0 | 1 |
| | P13 input (initial value) | 0 | 0 |



Downloaded from Arrow.com.

### (5) P14/(TCLKA-B)/SDA1/(IRQ4-B)

The pin function is switched as shown below according to the combination of the register setting for the RIIC and the B4 bit in P1.DDR.

| | | Setting | |
|---|---|---|---|
| | | RIIC | I/O Port |
| Module Name | Pin Function | SDA1_OE | P1.DDR.B4 |
| RIIC | SDA1 I/O | 1 | — |
| I/O port | P14 output | 0 | 1 |
| | P14 input (initial value) | 0 | 0 |

### (6) P15/(TCLKB-B)/SCK3/SCL1/(IRQ5-B)

The pin function is switched as shown below according to the combination of the register settings for the SCI and RIIC, and the B5 bit in P1.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | SCI | RIIC | I/O Port |
| Module Name | Pin Function | SCK3_OE | SCL1_OE | P1.DDR.B5 |
| SCI | SCK3 output | 1 | — | — |
| RIIC | SCL1 I/O | 0 | 1 | — |
| I/O port | P15 output | 0 | 0 | 1 |
| | P15 input (initial value) | 0 | 0 | 0 |

### (7) P16/(TCLKC-B)/(RxD3)/SDA0/(IRQ6-B)

The pin function is switched as shown below according to the combination of the register setting for the RIIC and the B6 bit in P1.DDR.

| | | Setting | |
|---|---|---|---|
| | | RIIC | I/O Port |
| Module Name | Pin Function | SDA0_OE | P1.DDR.B6 |
| RIIC | SDA0 I/O | 1 | — |
| I/O port | P16 output | 0 | 1 |
| | P16 input (initial value) | 0 | 0 |

### (8) P17/(TCLKD-B)/TxD3/SCL0/(ADTRG1#)/(IRQ7-B)

The pin function is switched as shown below according to the combination of the register settings for the SCI and RIIC, and the B7 bit in P1.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | SCI | RIIC | I/O Port |
| Module Name | Pin Function | TxD3_OE | SCL0_OE | P1.DDR.B7 |
| SCI | TxD3 output | 1 | — | — |
| RIIC | SCL0 I/O | 0 | 1 | — |
| I/O port | P17 output | 0 | 0 | 1 |
| | P17 input (initial value) | 0 | 0 | 0 |



Downloaded from Arrow.com.

## 14.3.3 Port 2 (P2)

### (1) P20/PO0/(TIOCA3)/TIOCB3/(TMRI0)/TxD0

The pin function is switched as shown below according to the combination of the register settings for the TPU, SCI, and PPG, and the B0 bit in P2.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | TPU | SCI | PPG | I/O Port |
| Module Name | Pin Function | TIOCB3_OE | TxD0_OE | PO0_OE | P2.DDR.B0 |
| TPU | TIOCB3 output | 1 | — | — | — |
| SCI | TxD0 output | 0 | 1 | — | — |
| PPG | PO0 output | 0 | 0 | 1 | — |
| I/O port | P20 output | 0 | 0 | 0 | 1 |
| | P20 input (initial value) | 0 | 0 | 0 | 0 |

### (2) P21/PO1/TIOCA3/(TMCI0)/(RxD0)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B1 bit in P2.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCA3_OE | PO1_OE | P2.DDR.B1 |
| TPU | TIOCA3 output | 1 | — | — |
| PPG | PO1 output | 0 | 1 | — |
| I/O port | P21 output | 0 | 0 | 1 |
| | P21 input (initial value) | 0 | 0 | 0 |

### (3) P22/PO2/TIOCC3/TMO0/SCK0

The pin function is switched as shown below according to the combination of the register settings for the TPU, TMR , SCI, and PPG, and the B2 bit in P2.DDR.

| | | Setting | | | | |
|---|---|---|---|---|---|---|
| | | TPU | TMR | SCI | PPG | I/O Port |
| Module Name | Pin Function | TIOCC3_OE | TMO0_OE | SCK0_OE | PO2_OE | P2.DDR.B2 |
| TPU | TIOCC3 output | 1 | — | — | — | — |
| TMR | TMO0 output | 0 | 1 | — | — | — |
| SCI | SCK0 output | 0 | 0 | 1 | — | — |
| PPG | PO2 output | 0 | 0 | 0 | 1 | — |
| I/O port | P22 output | 0 | 0 | 0 | 0 | 1 |
| | P22 input (initial value) | 0 | 0 | 0 | 0 | 0 |



Downloaded from Arrow.com.

### (4) P23/PO3/(TIOCC3)/TIOCD3

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B3 bit in P2.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCD3_OE | PO3_OE | P2.DDR.B3 |
| TPU | TIOCD3 output | 1 | — | — |
| PPG | PO3 output | 0 | 1 | — |
| I/O port | P23 output | 0 | 0 | 1 |
| | P23 input (initial value) | 0 | 0 | 0 |

### (5) P24/PO4/(TIOCA4)/TIOCB4/(TMRI1)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B4 bit in P2.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCB4_OE | PO4_OE | P2.DDR.B4 |
| TPU | TIOCB4 output | 1 | — | — |
| PPG | PO4 output | 0 | 1 | — |
| I/O port | P24 output | 0 | 0 | 1 |
| | P24 input (initial value) | 0 | 0 | 0 |

### (6) P25/PO5/TIOCA4/(TMCI1)/(RxD1)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B5 bit in P2.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCA4_OE | PO5_OE | P2.DDR.B5 |
| TPU | TIOCA4 output | 1 | — | — |
| PPG | PO5 output | 0 | 1 | — |
| I/O port | P25 output | 0 | 0 | 1 |
| | P25 input (initial value) | 0 | 0 | 0 |



Downloaded from Arrow.com.

## (7) P26/PO6/TIOCA5/TMO1/TxD1

The pin function is switched as shown below according to the combination of the register settings for the TPU, TMR , SCI, and PPG, and the B6 bit in P2.DDR.

| | | Setting | | | | |
|---|---|---|---|---|---|---|
| | | TPU | TMR | SCI | PPG | I/O Port |
| Module Name | Pin Function | TIOCA5_OE | TMO1_OE | TxD1_OE | PO6_OE | P2.DDR.B6 |
| TPU | TIOCA5 output | 1 | — | — | — | — |
| TMR | TMO1 output | 0 | 1 | — | — | — |
| SCI | TxD1 output | 0 | 0 | 1 | — | — |
| PPG | PO6 output | 0 | 0 | 0 | 1 | — |
| I/O port | P26 output | 0 | 0 | 0 | 0 | 1 |
| | P26 input (initial value) | 0 | 0 | 0 | 0 | 0 |

## (8) P27/PO7/(TIOCA5)/TIOCB5/SCK1

The pin function is switched as shown below according to the combination of the register settings for the TPU, SCI, and PPG, and the B7 bit in P2.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | TPU | SCI | PPG | I/O Port |
| Module Name | Pin Function | TIOCB5_OE | SCK1_OE | PO7_OE | P2.DDR.B7 |
| TPU | TIOCB5 output | 1 | — | — | — |
| SCI | SCK1 output | 0 | 1 | — | — |
| PPG | PO7 output | 0 | 0 | 1 | — |
| I/O port | P27 output | 0 | 0 | 0 | 1 |
| | P27 input (initial value) | 0 | 0 | 0 | 0 |



Downloaded from Arrow.com.

## 14.3.4 Port 3 (P3)

### (1) P30/PO8/TIOCA0/(IRQ0-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B0 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCA0_OE | PO8_OE | P3.DDR.B0 |
| TPU | TIOCA0 output | 1 | — | — |
| PPG | PO8 output | 0 | 1 | — |
| I/O port | P30 output | 0 | 0 | 1 |
| | P30 input (initial value) | 0 | 0 | 0 |

### (2) P31/PO9/(TIOCA0)/TIOCB0/(IRQ1-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B1 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCB0_OE | PO9_OE | P3.DDR.B1 |
| TPU | TIOCB0 output | 1 | — | — |
| PPG | PO9 output | 0 | 1 | — |
| I/O port | P31 output | 0 | 0 | 1 |
| | P31 input (initial value) | 0 | 0 | 0 |

### (3) P32/PO10/TIOCC0/(TCLKA-A)/(IRQ2-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B2 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCC0_OE | PO10_OE | P3.DDR.B2 |
| TPU | TIOCC0 output | 1 | — | — |
| PPG | PO10 output | 0 | 1 | — |
| I/O port | P32 output | 0 | 0 | 1 |
| | P32 input (initial value) | 0 | 0 | 0 |



Downloaded from Arrow.com.

### (4) P33/PO11/(TIOCC0)/TIOCD0/(TCLKB-A)/(IRQ3-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B3 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCD0_OE | PO11_OE | P3.DDR.B3 |
| TPU | TIOCD0 output | 1 | — | — |
| PPG | PO11 output | 0 | 1 | — |
| I/O port | P33 output | 0 | 0 | 1 |
| | P33 input (initial value) | 0 | 0 | 0 |

### (5) P34/PO12/TIOCA1/(IRQ4-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B4 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCA1_OE | PO12_OE | P3.DDR.B4 |
| TPU | TIOCA1 output | 1 | — | — |
| PPG | PO12 output | 0 | 1 | — |
| I/O port | P34 output | 0 | 0 | 1 |
| | P34 input (initial value) | 0 | 0 | 0 |

### (6) P35/PO13/(TIOCA1)/TIOCB1/(TCLKC-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B5 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCB1_OE | PO13_OE | P3.DDR.B5 |
| TPU | TIOCB1 output | 1 | — | — |
| PPG | PO13 output | 0 | 1 | — |
| I/O port | P35 output | 0 | 0 | 1 |
| | P35 input (initial value) | 0 | 0 | 0 |



Downloaded from Arrow.com.

### (7) P36/PO14/TIOCA2

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B6 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCA2_OE | PO14_OE | P3.DDR.B6 |
| TPU | TIOCA2 output | 1 | — | — |
| PPG | PO14 output | 0 | 1 | — |
| I/O port | P36 output | 0 | 0 | 1 |
| | P36 input (initial value) | 0 | 0 | 0 |

### (8) P37/PO15/(TIOCA2)/TIOCB2/(TCLKD-A)

The pin function is switched as shown below according to the combination of the register settings for the TPU and PPG, and the B7 bit in P3.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | TPU | PPG | I/O Port |
| Module Name | Pin Function | TIOCB2_OE | PO15_OE | P3.DDR.B7 |
| TPU | TIOCB2 output | 1 | — | — |
| PPG | PO15 output | 0 | 1 | — |
| I/O port | P37 output | 0 | 0 | 1 |
| | P37 input (initial value) | 0 | 0 | 0 |



Downloaded from Arrow.com.

### 14.3.5 Port 4 (P4)

#### (1) P40/(AN0)/(IRQ8-B)

The pin function is switched as shown below according to the value of the B0 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B0** |
| I/O port | P40 output | 1 |
| | P40 input (initial value) | 0 |

#### (2) P41/(AN1)/(IRQ9-B)

The pin function is switched as shown below according to the value of the B1 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B1** |
| I/O port | P41 output | 1 |
| | P41 input (initial value) | 0 |

#### (3) P42/(AN2)/(IRQ10-B)

The pin function is switched as shown below according to the value of the B2 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B2** |
| I/O port | P42 output | 1 |
| | P42 input (initial value) | 0 |

#### (4) P43/(AN3)/(IRQ11-B)

The pin function is switched as shown below according to the value of the B3 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B3** |
| I/O port | P43 output | 1 |
| | P43 input (initial value) | 0 |



Downloaded from Arrow.com.

### (5) P44/(AN4)/(IRQ12-B)

The pin function is switched as shown below according to the value of the B4 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B4** |
| I/O port | P44 output | 1 |
| | P44 input (initial value) | 0 |

### (6) P45/(AN5)/(IRQ13-B)

The pin function is switched as shown below according to the value of the B5 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B5** |
| I/O port | P45 output | 1 |
| | P45 input (initial value) | 0 |

### (7) P46/(AN6)/(IRQ14-B)

The pin function is switched as shown below according to the value of the B6 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B6** |
| I/O port | P46 output | 1 |
| | P46 input (initial value) | 0 |

### (8) P47/(AN7)/(IRQ15-B)

The pin function is switched as shown below according to the value of the B7 bit in P4.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P4.DDR.B7** |
| I/O port | P47 output | 1 |
| | P47 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.6 Port 5 (P5)

### (1) P50/WR0#/WR#

The pin function is switched as shown below according to the combination of the register setting for the bus controller and the B0 bit in P5.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| Module Name | Pin Function | WR0#_OE/WR#_OE | P5.DDR.B0 |
| Bus controller | WR0#/WR# output* | 1 | — |
| I/O port | P50 output | 0 | 1 |
| | P50 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (2) P51/WR1#/BC1#

The pin function is switched as shown below according to the combination of the port function control register (PFCRm) setting and the B1 bit in P5.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| Module Name | Pin Function | WR1#_OE/BC1#_OE | P5.DDR.B1 |
| Bus controller | WR1#/BC1# output* | 1 | — |
| I/O port | P51 output | 0 | 1 |
| | P51 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (3) P52/RD#

The pin function is switched as shown below according to the combination of the operating mode, the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), and the B2 bit in P5.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| Module Name | Pin Function | RD_OE | P5.DDR.B2 |
| Bus controller | RD# output* | 1 | — |
| I/O port | P52 output | 0 | 1 |
| | P52 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (4) P53/BCLK

The pin function is switched as shown below according to the value of the B3 bit in P5.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P5.DDR.B3 (BCLK_OE)** |
| Clock generation circuit | BCLK output | 1 |
| I/O port | P53 input (initial value) | 0 |

Note: If the BCLK signal is to be output, stop the BCLK clock by setting SCKCR.PSTOP1 to 1, set P5.DDR.B3 to select output for this pin, and then restore the value of SCKCR.PSTOP1 to 1 for output of the BCLK signal.

### (5) P54/TRDATA0

The pin function is switched as shown below according to the combination of the register setting for the B4 bit in P5.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P5.DDR.B4** |
| I/O port | P54 output | 1 |
| | P54 input (initial value) | 0 |

### (6) P55/TRDATA1

The pin function is switched as shown below according to the value of the B5 bit in P5.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P5.DDR.B5** |
| I/O port | P55 output | 1 |
| | P55 input (initial value) | 0 |

### (7) P56/TRDATA2

The pin function is switched as shown below according to the value of the B6 bit in P5.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P5.DDR.B6** |
| I/O port | P56 output | 1 |
| | P56 input (initial value) | 0 |

### (8) P57/(WAIT#)/TRDATA3

The pin function is switched as shown below according to the value of the B7 bit in P5.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P5.DDR.B7** |
| I/O port | P57 output | 1 |
| | P57 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.7 Port 6 (P6)

### (1) P60/CS0#/CS4#-A/CS5#-B

The pin function is switched as shown below according to the combination of the operating mode, the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B0 bit in P6.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | | | I/O Port |
| **Module Name** | **Pin Function** | **CS0#_OE** | **CS4#-A_OE** | **CS5#-B_OE** | **P6.DDR.B0** |
| Bus controller | CS0# output | 1 | — | — | — |
| | CS4#-A output | — | 1 | — | — |
| | CS5#-B output | — | — | 1 | — |
| I/O port | P60 output | 0 | 0 | 0 | 1 |
| | P60 input (initial value) | 0 | 0 | 0 | 0 |

### (2) P61/CS1#/CS2#-B/CS5#-A/CS6#-B/CS7#-B

The pin function is switched as shown below according to the combination of the operating mode, the EXBE bit in SYSCR0, the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B1 bit in P6.DDR.

| | | Setting | | | | | |
|---|---|---|---|---|---|---|---|
| | | Bus Controller | | | | | I/O Port |
| **Module Name** | **Pin Function** | **CS1#_OE** | **CS2#-B_OE** | **CS5#-A_OE** | **CS6#-B_OE** | **CS7#-B_OE** | **P6.DDR.B1** |
| Bus controller | CS1# output* | 1 | — | — | — | — | — |
| | CS2#-B output* | — | 1 | — | — | — | — |
| | CS5#-A output* | — | — | 1 | — | — | — |
| | CS6#-B output* | — | — | — | 1 | — | — |
| | CS7#-B output* | — | — | — | — | 1 | — |
| I/O port | P61 output | 0 | 0 | 0 | 0 | 0 | 1 |
| | P61 input (initial value) | 0 | 0 | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (3) P62/CS2#-A/CS6#-A

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B2 bit in P6.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | Bus Controller | | I/O Port |
| **Module Name** | **Pin Function** | **CS2#-A_OE** | **CS6#-A_OE** | **P6.DDR.B2** |
| Bus controller | CS2#-A output* | 1 | — | — |
| | CS6#-A output* | — | 1 | — |
| I/O port | P62 output | 0 | 0 | 1 |
| | P62 input (initial value) | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (4) P63/CS3#-A/CS7#-A

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B3 bit in P6.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | Bus Controller | | I/O Port |
| **Module Name** | **Pin Function** | **CS3#-A_OE** | **CS7#-A_OE** | **P6.DDR.B3** |
| Bus controller | CS3#-A output* | 1 | — | — |
| | CS7#-A output* | — | 1 | — |
| I/O port | P63 output | 0 | 0 | 1 |
| | P63 input (initial value) | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (5) P64/CS4#-B

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B4 bit in P6.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| **Module Name** | **Pin Function** | **CS4#-B_OE** | **P6.DDR.B4** |
| Bus controller | CS4#-B output* | 1 | — |
| I/O port | P64 output | 0 | 1 |
| | P64 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (6) P65/(IRQ15-A)

The pin function is switched as shown below according to the value of the B5 bit in P6.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P6.DDR.B5** |
| I/O port | P65 output | 1 |
| | P65 input (initial value) | 0 |

### (7) P66/DA0

The pin function is switched as shown below according to the register setting for the D/A converter and the B6 bit in P6.DDR.

| | | Setting | |
|---|---|---|---|
| | | D/A Converter | I/O Port |
| **Module Name** | **Pin Function** | **DA0_OE** | **P6.DDR.B6** |
| D/A converter | DA0 output | 1 | — |
| I/O port | P66 output | 0 | 1 |
| | P66 input (initial value) | 0 | 0 |

### (8) P67/DA1

The pin function is switched as shown below according to the register setting for the D/A converter and the B7 bit in P6.DDR.

| | | Setting | |
|---|---|---|---|
| | | D/A Converter | I/O Port |
| **Module Name** | **Pin Function** | **DA1_OE** | **P6.DDR.B7** |
| D/A converter | DA1 output | 1 | — |
| I/O port | P67 output | 0 | 1 |
| | P67 input (initial value) | 0 | 0 |



Downloaded from Arrow.com.

## 14.3.8 Port 7 (P7)

### (1) P70/CS3#-B/(ADTRG2#)

The pin function is switched as shown below according to the combination of the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B0 bit in P7.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| **Module Name** | **Pin Function** | **CS3#-B_OE** | **P7.DDR.B0** |
| Bus controller | CS3#-B output* | 1 | — |
| I/O port | P70 output | 0 | 1 |
| | P70 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (2) P71/CS4#-C/CS5#-C/CS6#-C/CS7#-C

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B1 bit in P7.DDR.

| | | Setting | | | | |
|---|---|---|---|---|---|---|
| | | Bus Controller | | | | I/O Port |
| **Module Name** | **Pin Function** | **CS4#-C_OE** | **CS5#-C_OE** | **CS6#-C_OE** | **CS7#-C_OE** | **P7.DDR.B1** |
| Bus controller | CS4#-C output* | 1 | — | — | — | — |
| | CS5#-C output* | — | 1 | — | — | — |
| | CS6#-C output* | — | — | 1 | — | — |
| | CS7#-C output* | — | — | — | 1 | — |
| I/O port | P71 output | 0 | 0 | 0 | 0 | 1 |
| | P71 input (initial value) | 0 | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (3) P72

The pin function is switched as shown below according to the value of the B2 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B2** |
| I/O port | P72 output | 1 |
| | P72 input (initial value) | 0 |



Downloaded from Arrow.com.

### (4) P73

The pin function is switched as shown below according to the value of the B3 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B3** |
| I/O port | P73 output | 1 |
| | P73 input (initial value) | 0 |

### (5) P74/(ADTRG3#)

The pin function is switched as shown below according to the value of the B4 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B4** |
| I/O port | P74 output | 1 |
| | P74 input (initial value) | 0 |

### (6) P75

The pin function is switched as shown below according to the value of the B5 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B5** |
| I/O port | P75 output | 1 |
| | P75 input (initial value) | 0 |

### (7) P76/(IRQ14-A)

The pin function is switched as shown below according to the value of the B6 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B6** |
| I/O port | P76 output | 1 |
| | P76 input (initial value) | 0 |

### (8) P77

The pin function is switched as shown below according to the value of the B7 bit in P7.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P7.DDR.B7** |
| I/O port | P77 output | 1 |
| | P77 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.9 Port 8 (P8)

### (1) P80

The pin function is switched as shown below according to the value of the B0 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B0** |
| I/O port | P80 output | 1 |
| | P80 input (initial value) | 0 |

### (2) P81/TRSYNC#

The pin function is switched as shown below according to the value of the B1 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B1** |
| I/O port | P81 output | 1 |
| | P81 input (initial value) | 0 |

### (3) P82/TRCLK

The pin function is switched as shown below according to the value of the B2 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B2** |
| I/O port | P82 output | 1 |
| | P82 input (initial value) | 0 |

### (4) P83

The pin function is switched as shown below according to the value of the B3 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B3** |
| I/O port | P83 output | 1 |
| | P83 input (initial value) | 0 |



Downloaded from Arrow.com.

### (5) P84

The pin function is switched as shown below according to the value of the B4 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B4** |
| I/O port | P84 output | 1 |
| | P84 input (initial value) | 0 |

### (6) P85

The pin function is switched as shown below according to the value of the B5 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B5** |
| I/O port | P85 output | 1 |
| | P85 input (initial value) | 0 |

### (7) P86

The pin function is switched as shown below according to the value of the B6 bit in P8.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P8.DDR.B6** |
| I/O port | P86 output | 1 |
| | P86 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.10 Port 9 (P9)

### (1) P90/(AN8)

The pin function is switched as shown below according to the value of the B0 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B0** |
| I/O port | P90 output | 1 |
| | P90 input (initial value) | 0 |

### (2) P91/(AN9)

The pin function is switched as shown below according to the value of the B1 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B1** |
| I/O port | P91 output | 1 |
| | P91 input (initial value) | 0 |

### (3) P92/(AN10)

The pin function is switched as shown below according to the value of the B2 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B2** |
| I/O port | P92 output | 1 |
| | P92 input (initial value) | 0 |

### (4) P93/(AN11)

The pin function is switched as shown below according to the value of the B3 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B3** |
| I/O port | P93 output | 1 |
| | P93 input (initial value) | 0 |



Downloaded from Arrow.com.

### (5) P94/(AN12)

The pin function is switched as shown below according to the value of the B4 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B4** |
| I/O port | P94 output | 1 |
| | P94 input (initial value) | 0 |

### (6) P95/(AN13)

The pin function is switched as shown below according to the value of the B5 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B5** |
| I/O port | P95 output | 1 |
| | P95 input (initial value) | 0 |

### (7) P96/(AN14)

The pin function is switched as shown below according to the value of the B6 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B6** |
| I/O port | P96 output | 1 |
| | P96 input (initial value) | 0 |

### (8) P97/(AN15)

The pin function is switched as shown below according to the value of the B7 bit in P9.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **P9.DDR.B7** |
| I/O port | P97 output | 1 |
| | P97 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.11 Port A (PA)

### (1) PA0/A0/BC0#/PO16/TIOCA6

The pin function is switched as shown below according to the combination of the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), the rgister settings for the PPG and TPU, and the B0 bit in PA.DDR.

| | | Setting | | | | |
|---|---|---|---|---|---|---|
| | | Bus Controller | | TPU | PPG | I/O Port |
| **Module Name** | **Pin Function** | **A0_OE** | **BC0#_OE** | **TIOCA6_OE** | **PO16_OE** | **PA.DDR.B0** |
| Bus controller | Address output* | 1 | 0 | — | — | 1 |
| | Byte control output* | 0 | 1 | — | — | 1 |
| TPU | TIOCA6 output | 0 | 0 | 1 | — | — |
| PPG | PO16 output | 0 | 0 | 0 | 1 | — |
| I/O port | PA0 output* | 0 | 0 | 0 | 0 | 1 |
| | PA0 input (initial value) | 0 | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B0 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (2) PA1/A1/PO17/(TIOCA6)/TIOCB6

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B1 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| **Module Name** | **Pin Function** | **A1_OE** | **TIOCB6_OE** | **PO17_OE** | **PA.DDR.B1** |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCB6 output | 0 | 1 | — | — |
| PPG | PO17 output | 0 | 0 | 1 | — |
| I/O port | PA1 output* | 0 | 0 | 0 | 1 |
| | PA1 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B1 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (3) PA2/A2/PO18/TIOCC6/(TCLKE)

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B2 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A2_OE | TIOCC6_OE | PO18_OE | PA.DDR.B2 |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCC6 output | 0 | 1 | — | — |
| PPG | PO18 output | 0 | 0 | 1 | — |
| I/O port | PA2 output* | 0 | 0 | 0 | 1 |
| | PA2 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B2 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (4) PA3/A3/PO19/(TIOCC6)/TIOCD6/(TCLKF)

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B3 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A3_OE | TIOCD6_OE | PO19_OE | PA.DDR.B3 |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCD6 output | 0 | 1 | — | — |
| PPG | PO19 output | 0 | 0 | 1 | — |
| I/O port | PA3 output* | 0 | 0 | 0 | 1 |
| | PA3 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B3 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (5) PA4/A4/PO20/TIOCA7

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B4 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A4_OE | TIOCA7_OE | PO20_OE | PA.DDR.B4 |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCA7 output | 0 | 1 | — | — |
| PPG | PO20 output | 0 | 0 | 1 | — |
| I/O port | PA4 output* | 0 | 0 | 0 | 1 |
| | PA4 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B4 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (6) PA5/A5/PO21/(TIOCA7)/TIOCB7/(TCLKG)

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B5 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A5_OE | TIOCB7_OE | PO21_OE | PA.DDR.B5 |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCB7 output | 0 | 1 | — | — |
| PPG | PO21 output | 0 | 0 | 1 | — |
| I/O port | PA5 output* | 0 | 0 | 0 | 1 |
| | PA5 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B5 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

## (7) PA6/A6/PO22/TIOCA8

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B6 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | **Bus Controller** | **TPU** | **PPG** | **I/O Port** |
| **Module Name** | **Pin Function** | **A6_OE** | **TIOCA8_OE** | **PO22_OE** | **PA.DDR.B6** |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCA8 output | 0 | 1 | — | — |
| PPG | PO22 output | 0 | 0 | 1 | — |
| I/O port | PA6 output* | 0 | 0 | 0 | 1 |
| | PA6 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B6 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

## (8) PA7/A7/PO23/(TIOCA8)/TIOCB8/(TCLKH)

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, and the B7 bit in PA.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | **Bus Controller** | **TPU** | **PPG** | **I/O Port** |
| **Module Name** | **Pin Function** | **A7_OE** | **TIOCB8_OE** | **PO23_OE** | **PA.DDR.B7** |
| Bus controller | Address output* | 1 | — | — | 1 |
| TPU | TIOCB8 output | 0 | 1 | — | — |
| PPG | PO23 output | 0 | 0 | 1 | — |
| I/O port | PA7 output* | 0 | 0 | 0 | 1 |
| | PA7 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Address output is enabled when PA.DDR.B7 = 1 in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

## 14.3.12 Port B (PB)

### (1) PB0/A8/PO24/TIOCA9

The pin function is switched as shown below according to the combination of the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B0 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | **Bus Controller** | **TPU** | **PPG** | **I/O Port** |
| **Module Name** | **Pin Function** | **A8_OE** | **TIOCA9_OE** | **PO24_OE** | **PB.DDR.B0** |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCA9 output | 0 | 1 | — | — |
| PPG | PO24 output | 0 | 0 | 1 | — |
| I/O port | PB0 output* | 0 | 0 | 0 | 1 |
| | PB0 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (2) PB1/A9/PO25/(TIOCA9)/TIOCB9

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B1 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | **Bus Controller** | **TPU** | **PPG** | **I/O Port** |
| **Module Name** | **Pin Function** | **A9_OE** | **TIOCB9_OE** | **PO25_OE** | **PB.DDR.B1** |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCB9 output | 0 | 1 | — | — |
| PPG | PO25 output | 0 | 0 | 1 | — |
| I/O port | PB1 output* | 0 | 0 | 0 | 1 |
| | PB1 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (3) PB2/A10/PO26/TIOCC9

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B2 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A10_OE | TIOCC9_OE | PO26_OE | PB.DDR.B2 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCC9 output | 0 | 1 | — | — |
| PPG | PO26 output | 0 | 0 | 1 | — |
| I/O port | PB2 output* | 0 | 0 | 0 | 1 |
| | PB2 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (4) PB3/A11/PO27/(TIOCC9)/TIOCD9

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B3 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A11_OE | TIOCD9_OE | PO27_OE | PB.DDR.B3 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCD9 output | 0 | 1 | — | — |
| PPG | PO27 output | 0 | 0 | 1 | — |
| I/O port | PB3 output* | 0 | 0 | 0 | 1 |
| | PB3 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (5) PB4/A12/PO28/TIOCA10

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B4 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A12_OE | TIOCA10_OE | PO28_OE | PB.DDR.B4 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCA10 output | 0 | 1 | — | — |
| PPG | PO28 output | 0 | 0 | 1 | — |
| I/O port | PB4 output* | 0 | 0 | 0 | 1 |
| | PB4 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (6) PB5/A13/PO29/(TIOCA10)/TIOCB10

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B5 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A13_OE | TIOCB10_OE | PO29_OE | PB.DDR.B5 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCB10 output | 0 | 1 | — | — |
| PPG | PO29 output | 0 | 0 | 1 | — |
| I/O port | PB5 output* | 0 | 0 | 0 | 1 |
| | PB5 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (7) PB6/A14/PO30/TIOCA11

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B6 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A14_OE | TIOCA11_OE | PO30_OE | PB.DDR.B6 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCA11 output | 0 | 1 | — | — |
| PPG | PO30 output | 0 | 0 | 1 | — |
| I/O port | PB6 output* | 0 | 0 | 0 | 1 |
| | PB6 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (8) PB7/A15/PO31/(TIOCA11)/TIOCB11

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the register settings for the PPG and TPU, the port function control register m (PFCRm) setting, and the B7 bit in PB.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | TPU | PPG | I/O Port |
| Module Name | Pin Function | A15_OE | TIOCB11_OE | PO31_OE | PB.DDR.B7 |
| Bus controller | Address output* | 1 | — | — | — |
| TPU | TIOCB11 output | 0 | 1 | — | — |
| PPG | PO31 output | 0 | 0 | 1 | — |
| I/O port | PB7 output* | 0 | 0 | 0 | 1 |
| | PB7 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

## 14.3.13 Port C (PC)

### (1) PC0/A16

The pin function is switched as shown below according to the combination of the external bus enable bit (EXBE) in the system control register 0 (SYSCR0), the register setting for the bus controller, the port function control register m (PFCRm) setting, and the B0 bit in PC.DDR

| | | Setting | |
|---|---|---|---|
| | | **Bus Controller** | **I/O Port** |
| **Module Name** | **Pin Function** | **A16_OE** | **PC.DDR.B0** |
| Bus controller | A16 output* | 1 | — |
| I/O port | PC0 output | 0 | 1 |
| | PC0 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (2) PC1/A17

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister settings for the bus controller, the port function control register m (PFCRm) setting, and the B1 bit in PC.DDR.

| | | Setting | |
|---|---|---|---|
| | | **Bus Controller** | **I/O Port** |
| **Module Name** | **Pin Function** | **A17_OE** | **PC.DDR.B1** |
| Bus controller | A17 output* | 1 | — |
| I/O port | PC1output | 0 | 1 |
| | PC1 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (3) PC2/A18

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B2 bit in PC.DDR.

| | | Setting | |
|---|---|---|---|
| | | **Bus Controller** | **I/O Port** |
| **Module Name** | **Pin Function** | **A18_OE** | **PC.DDR.B2** |
| Bus controller | A18 output* | 1 | — |
| I/O port | PC2 output | 0 | 1 |
| | PC2 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (4) PC3/A19

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B3 bit in PC.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| Module Name | Pin Function | A19_OE | PC.DDR.B3 |
| Bus controller | A19 output* | 1 | — |
| I/O port | PC3 output | 0 | 1 |
| | PC3 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (5) PC4/A20

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B4 bit in PC.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| Module Name | Pin Function | A20_OE | PC.DDR.B4 |
| Bus controller | A20 output* | 1 | — |
| I/O port | PC4 output | 0 | 1 |
| | PC4 input (initial value) | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (6) PC5/A21/SCK5/CS5#-D

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B5 bit in PC.DDR.

| | | Setting | | | |
|---|---|---|---|---|---|
| | | Bus Controller | | SCI | I/O Port |
| Module Name | Pin Function | A21_OE | CS5#-D_OE | SCK5_OE | PC.DDR.B5 |
| Bus controller | A21 output* | 1 | — | — | — |
| | CS5#-D output* | 0 | 1 | — | — |
| SCI | SCK5 output | 0 | 0 | 1 | — |
| I/O port | PC5 output | 0 | 0 | 0 | 1 |
| | PC5 input (initial value) | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

### (7) PC6/A22/(RxD5)/CS6#-D

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B6 bit in PC.DDR.

| | | Setting | | |
|---|---|---|---|---|
| | | Bus Controller | | I/O Port |
| **Module Name** | **Pin Function** | **A22_OE** | **CS6#-D_OE** | **PC.DDR.B6** |
| Bus controller | A22 output* | 1 | — | — |
| | CS6#-D output* | 0 | 1 | — |
| I/O port | PC6 output | 0 | 0 | 1 |
| | PC6 input (initial value) | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

### (8) PC7/A23/TxD5/CS4#-D/CS7#-D

The pin function is switched as shown below according to the combination of the EXBE bit in SYSCR0, the rgister setting for the bus controller, the port function control register m (PFCRm) setting, and the B7 bit in PC.DDR.

| | | Setting | | | | |
|---|---|---|---|---|---|---|
| | | Bus Controller | | | SCI | I/O Port |
| **Module Name** | **Pin Function** | **A23_OE** | **CS4#-D_OE** | **CS7#-D_OE** | **TxD5_OE** | **PC.DDR.B7** |
| Bus controller | A23 output* | 1 | — | — | — | — |
| | CS4#-D output* | 0 | 1 | — | — | — |
| | CS7#-D output* | 0 | 0 | 1 | — | — |
| SCI | TxD5 output | 0 | 0 | 0 | 1 | — |
| I/O port | PC7 output | 0 | 0 | 0 | 0 | 1 |
| | PC7 input (initial value) | 0 | 0 | 0 | 0 | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).



Downloaded from Arrow.com.

## 14.3.14 Port D (PD)

### (1) PD0/D0, PD1/D1, PD2/D2, PD3/D3, PD4/D4, PD5/D5, PD6/D6, PD7/D7

The pin function is switched as shown below according to the combination of the external bus enable (EXBE) bit in the system control register 0 (SYSCR0) and the Bj bit (j = 0 to 7) in PD.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PD.DDR.Bn** |
| Bus controller | Data I/O* | — |
| I/O port | PDn output | 1 |
| | PDn input (initial value) | 0 |

Note: * Enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).

## 14.3.15 Port E (PE)

### (1) PE0/D8, PE1/D9, PE2/D10, PE3/D11, PE4/D12, PE5/D13/(IRQ5#-A), PE6/D14/(IRQ6#-A), PE7/D15/(IRQ7#-A)

The pin function is switched as shown below according to the combination of bus mode setting, the EXBE bit in SYSCR0, the port function control register m (PFCRm) setting, and the Bj bit (j = 0 to 7) in PE.DDR.

| | | Setting | |
|---|---|---|---|
| | | Bus Controller | I/O Port |
| **Module Name** | **Pin Function** | **Dn_E (n = 8 to 15)** | **PE.DDR.Bn** |
| Bus controller | Data I/O* | 1 | — |
| I/O port | PEn output | 0 | 1 |
| | PEn input (initial value) | 0 | 0 |

Note: * The function is enabled in expansion mode with on-chip ROM disabled or disabled (SYSCR0.EXBE = 1).
These pins function as data I/O pins (D15 to D8) in 16-bit bus mode (indicated as 1 in the above table), and as general I/O pins in the other modes (indicated as 0 in the above table).



Downloaded from Arrow.com.

### 14.3.16 Port F (PF)

#### (1) PF0

The pin function is switched as shown below according to the value of the B0 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B0** |
| I/O port | PF0 output | 1 |
| | PF0 input (initial value) | 0 |

#### (2) PF1

The pin function is switched as shown below according to the value of the B1 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B1** |
| I/O port | PF1 output | 1 |
| | PF1 input (initial value) | 0 |

#### (3) PF2

The pin function is switched as shown below according to the value of the B2 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B2** |
| I/O port | PF2 output | 1 |
| | PF2 input (initial value) | 0 |

#### (4) PF3

The pin function is switched as shown below according to the value of the B3 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B3** |
| I/O port | PF3 output | 1 |
| | PF3 input (initial value) | 0 |



Downloaded from Arrow.com.

### (5) PF4

The pin function is switched as shown below according to the value of the B4 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B4** |
| I/O port | PF4 output | 1 |
| | PF4 input (initial value) | 0 |

### (6) PF5

The pin function is switched as shown below according to the value of the B5 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B5** |
| I/O port | PF5 output | 1 |
| | PF5 input (initial value) | 0 |

### (7) PF6

The pin function is switched as shown below according to the value of the B6 bit in PF.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PF.DDR.B6** |
| I/O port | PF6 output | 1 |
| | PF6 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.17 Port G (PG)

### (1) PG0

The pin function is switched as shown below according to the value of the B0 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B0** |
| I/O port | PG0 output | 1 |
| | PG0 input (initial value) | 0 |

### (2) PG1

The pin function is switched as shown below according to the value of the B1 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B1** |
| I/O port | PG1 output | 1 |
| | PG1 input (initial value) | 0 |

### (3) PG2

The pin function is switched as shown below according to the value of the B2 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B2** |
| I/O port | PG2 output | 1 |
| | PG2 input (initial value) | 0 |

### (4) PG3

The pin function is switched as shown below according to the value of the B3 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B3** |
| I/O port | PG3 output | 1 |
| | PG3 input (initial value) | 0 |

### (5) PG4

The pin function is switched as shown below according to the value of the B4 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B4** |
| I/O port | PG4 output | 1 |
| | PG4 input (initial value) | 0 |



Downloaded from Arrow.com.

### (6) PG5

The pin function is switched as shown below according to the value of the B5 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B5** |
| I/O port | PG5 output | 1 |
| | PG5 input (initial value) | 0 |

### (7) PG6

The pin function is switched as shown below according to the value of the B6 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B6** |
| I/O port | PG6 output | 1 |
| | PG6 input (initial value) | 0 |

### (8) PG7

The pin function is switched as shown below according to the value of the B7 bit in PG.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PG.DDR.B7** |
| I/O port | PG7 output | 1 |
| | PG7 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.3.18 Port H (PH)

### (1) PH0

The pin function is switched as shown below according to the value of the B0 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B0** |
| I/O port | PH0 output | 1 |
| | PH0 input (initial value) | 0 |

### (2) PH1

The pin function is switched as shown below according to the value of the B1 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B1** |
| I/O port | PH1 output | 1 |
| | PH1 input (initial value) | 0 |

### (3) PH2

The pin function is switched as shown below according to the value of the B2 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B2** |
| I/O port | PH2 output | 1 |
| | PH2 input (initial value) | 0 |

### (4) PH3

The pin function is switched as shown below according to the value of the B3 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B3** |
| I/O port | PH3 output | 1 |
| | PH3 input (initial value) | 0 |

### (5) PH4

The pin function is switched as shown below according to the value of the B4 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B4** |
| I/O port | PH4 output | 1 |
| | PH4 input (initial value) | 0 |



Downloaded from Arrow.com.

### (6) PH5

The pin function is switched as shown below according to the value of the B5 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B5** |
| I/O port | PH5 output | 1 |
| | PH5 input (initial value) | 0 |

### (7) PH6

The pin function is switched as shown below according to the value of the B6 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B6** |
| I/O port | PH6 output | 1 |
| | PH6 input (initial value) | 0 |

### (8) PH7

The pin function is switched as shown below according to the value of the B7 bit in PH.DDR.

| | | Setting |
|---|---|---|
| | | I/O Port |
| **Module Name** | **Pin Function** | **PH.DDR.B7** |
| I/O port | PH7 output | 1 |
| | PH7 input (initial value) | 0 |



Downloaded from Arrow.com.

## 14.4 Settings to Enable Output of the Signals

Table 14.8 lists the settings to enable output from the individual pins of each port.

**Table 14.8 Settings to Enable Output of the Signals from Each Port**

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| P0 | 0 | SCI6 | TxD6_OE | TxD6 | | SCR.TE = 1 |
| | 1 | | — | — | — | — |
| | 2 | TMR2 | TMO2_OE | TMO2 | | TCSR.OSA[1:0] = 01/10/11 or TCSR.OSB[1:0] = 01/10/11 |
| | | SCI6 | SCK6_OE | SCK6 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1,SCR.CKE[1] = 0 |
| | 3 | SCI4 | SCK4_OE | SCK4 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | 4 | SCI4 | TxD4_OE | TxD4 | | SCR.TE = 1 |
| | 5 | TMR3 | TMO3_OE | TMO3 | | TCSR.OSA[1:0] = 01/10/11 or TCSR.OSB[1:0] = 01/10/11 |
| P1 | 0 | | — | — | — | — |
| | 1 | SCI2 | SCK2_OE | SCK2 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | 2 | | — | — | — | — |
| | 3 | SCI2 | TxD2_OE | TxD2 | | SCR.TE = 1 |
| | 4 | RIIC1 | SDA1_OE | SDA1 | | ICCR1.ICE = 1 |
| | 5 | SCI3 | SCK3_OE | SCK3 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | | RIIC1 | SCL1_OE | SCL1 | | ICCR1.ICE = 1 |
| | 6 | RIIC0 | SDA0_OE | SDA0 | | ICCR1.ICE = 1 |
| | 7 | SCI3 | TxD3_OE | TxD3 | | SCR.TE = 1 |
| | | RIIC0 | SCL0_OE | SCL0 | | ICCR1.ICE = 1 |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| P2 | 0 | TPU3 | TIOCB3_OE | TIOCB3 | | TIORH.IOB[3] = 0, TIORH.IOB[1 0] = 01/10/11 |
| | | SCI0 | TxD0_OE | TxD0 | | SCR.TE = 1 |
| | | PPG0 | PO0_OE | PO0 | | NDERL.NDER0 = 1 |
| | 1 | TPU3 | TIOCA3_OE | TIOCA3 | | TIORH.IOA[3] = 0, TIORH.IOA[1 0] = 01/10/11 |
| | | PPG0 | PO1_OE | PO1 | | NDERL.NDER1 = 1 |
| | 2 | TPU3 | TIOCC3_OE | TIOCC3 | | TMDR.BFA = 0, TIORL.IOC[3] = 0, TIORL.IOC[1:0] = 01/10/11 |
| | | TMR0 | TMO0_OE | TMO0 | | TCSR.OSA[1:0]=01/10/11 or TCSR.OSB[1:0]=01/10/11 |
| | | SCI0 | SCK0_OE | SCK0 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | | PPG0 | PO2_OE | PO2 | | NDERL.NDER2 = 1 |
| | 3 | TPU3 | TIOCD3_OE | TIOCD3 | | TMDR.BFB = 0, TIORL.IOD[3] = 0, TIORL.IOD[1:0] = 01/10/11 |
| | | PPG0 | PO3_OE | PO3 | | NDERL.NDER3 = 1 |
| | 4 | TPU4 | TIOCB4_OE | TIOCB4 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG0 | PO4_OE | PO4 | | NDERL.NDER4 = 1 |
| | 5 | TPU4 | TIOCA4_OE | TIOCA4 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG0 | PO5_OE | PO5 | | NDERL.NDER5 = 1 |
| | 6 | TPU5 | TIOCA5_OE | TIOCA5 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | TMR1 | TMO1_OE | TMO1 | | TCSR.OSA[1:0] = 01/10/11 or TCSR.OSB[1:0] = 01/10/11 |
| | | SCI1 | TxD1_OE | TxD1 | | SCR.TE = 1 |
| | | PPG0 | PO6_OE | PO6 | | NDERL.NDER6 = 1 |
| | 7 | TPU5 | TIOCB5_OE | TIOCB5 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | SCI1 | SCK1_OE | SCK1 | | When SCMR.SMIF=1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | | PPG0 | PO7_OE | PO7 | | NDERL.NDER7 = 1 |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| P3 | 0 | TPU0 | TIOCA0_OE | TIOCA0 | | TIORH.IOA[3] = 0, TIOH.IOA[1:0] = 01/10/11 |
| | | PPG0 | PO8_OE | PO8 | | NDERH.NDER8 = 1 |
| | 1 | TPU0 | TIOCB0_OE | TIOCB0 | | TIORH.IOB[3] = 0, TIORH.IOB[1 0] = 01/10/11 |
| | | PPG0 | PO9_OE | PO9 | | NDERH.NDER9 = 1 |
| | 2 | TPU0 | TIOCC0_OE | TIOCC0 | | TMDR.BFA = 0, TIORL.IOC[3] = 0,TIORL.IOC[1:0] = 01/10/11 |
| | | PPG0 | PO10_OE | PO10 | | NDERH.NDER10 = 1 |
| | 3 | TPU0 | TIOCD0_OE | TIOCD0 | | TMDR.BFB = 0, TIORL.IOD[3] = 0, TIORL.IOD[1:0] = 01/10/11 |
| | | PPG0 | PO11_OE | PO11 | | NDERH.NDER11 = 1 |
| | 4 | TPU1 | TIOCA1_OE | TIOCA1 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG0 | PO12_OE | PO12 | | NDERH.NDER12 = 1 |
| | 5 | TPU1 | TIOCB1_OE | TIOCB1 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG0 | PO13_OE | PO13 | | NDERH.NDER13 = 1 |
| | 6 | TPU2 | TIOCA2_OE | TIOCA2 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG0 | PO14_OE | PO14 | | NDERH.NDER14 = 1 |
| | 7 | TPU2 | TIOCB2_OE | TIOCB2 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG0 | PO15_OE | PO15 | | NDERH.NDER15 = 1 |
| P5 | 0 | SYSC,BSC | WR0_OE | WR0 | | SYSCR0.EXBE = 1, CSnMOD.WRMOD = 0 |
| | | SYSC,BSC | WR_OE | WR | | SYSCR0.EXBE = 1, CSnMOD.WRMOD = 1 |
| | 1 | SYSC,BSC | WR1_OE | WR1 | | SYSCR0.EXBE = 1, PFCR5.WR1BC1E = 1, CSnMOD.WRMOD = 0<br>CSnCNT.BSIZE[1 0] = 00 |
| | | SYSC,BSC | BC1_OE | BC1 | | SYSCR0.EXBE=1,PFCR5.WR1BC1E =1,CSnMOD.WRMOD=1<br>CSnCNT.BSIZE[1 0] = 00 |
| | 2 | SYSC | RD_OE | RD | | SYSCR0.EXBE = 1 |
| | 3 | PORT | BCLK_OE | BCLK | | P5.DDR.B3 = 1 |
| | 4 | | — | — | — | — |
| | 5 | | — | — | — | — |
| | 6 | | — | — | — | — |
| | 7 | | — | — | — | — |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| P6 | 0 | SYSC | CS0_OE | CS0 | | SYSCR0.EXBE = 1, PFCR0.CS0E = 1, CS0CNT.EXENB = 1 |
| | | SYSC | CS4A_OE | CS4 | PFCR1.CS4S[1:0] = 00 | SYSCR0.EXBE = 1, PFCR0.CS4E = 1, CS4CNT.EXENB = 1 |
| | | SYSC | CS5B_OE | CS5 | PFCR1.CS5S[1:0] = 01 | SYSCR0.EXBE = 1, PFCR0.CS5E = 1, CS5CNT.EXENB = 1 |
| | 1 | SYSC | CS1_OE | CS1 | | SYSCR0.EXBE = 1, PFCR0.CS1E = 1, CS1CNT.EXENB = 1 |
| | | SYSC | CS2B_OE | CS2 | PFCR2.CS2S = 1 | SYSCR0.EXBE = 1, PFCR0.CS2E = 1, CS2CNT.EXENB = 1 |
| | | SYSC | CS5A_OE | CS5 | PFCR1.CS5S[1:0] = 00 | SYSCR0.EXBE = 1, PFCR0.CS5E = 1, CS5CNT.EXENB = 1 |
| | | SYSC | CS6B_OE | CS6 | PFCR1.CS6S[1:0] = 01 | SYSCR0.EXBE = 1, PFCR0.CS6E = 1, CS6CNT.EXENB = 1 |
| | | SYSC | CS7B_OE | CS7 | PFCR1.CS7S[1:0] = 01 | SYSCR0.EXBE = 1, PFCR0.CS7E = 1, CS7CNT.EXENB = 1 |
| | 2 | SYSC | CS2A_OE | CS2 | PFCR2.CS2S = 0 | SYSCR0.EXBE = 1, PFCR0.CS2E = 1, CS2CNT.EXENB = 1 |
| | | SYSC | CS6A_OE | CS6 | PFCR1.CS6S[1:0] = 00 | SYSCR0.EXBE = 1, PFCR0.CS6E = 1, CS6CNT.EXENB = 1 |
| | 3 | SYSC | CS3A_OE | CS3 | PFCR2.CS3S = 0 | SYSCR0.EXBE = 1, PFCR0.CS3E = 1, CS3CNT.EXENB = 1 |
| | | SYSC | CS7A_OE | CS7 | PFCR1.CS7S[1:0] = 00 | SYSCR0.EXBE = 1, PFCR0.CS7E = 1, CS7CNT.EXENB = 1 |
| | 4 | SYSC | CS4B_OE | CS4 | PFCR1.CS4S[1:0] = 01 | SYSCR0.EXBE = 1, PFCR0.CS4E = 1, CS4CNT.EXENB = 1 |
| | 5 | | — | — | — | — |
| | 6 | DAC | DA0_OE | DA0 | | DACR.DAOE0 = 1 |
| | 7 | DAC | DA1_OE | DA1 | | DACR.DAOE1 = 1 |
| P7 | 0 | SYSC | CS3B_OE | CS3 | PFCR2.CS3S = 1 | SYSCR0.EXBE = 1, PFCR0.CS3E = 1, CS3CNT.EXENB = 1 |
| | 1 | SYSC | CS4C_OE | CS4 | PFCR1.CS4S[1:0] = 10 | SYSCR0.EXBE = 1, PFCR0.CS4E = 1, CS4CNT.EXENB = 1 |
| | | SYSC | CS5C_OE | CS5 | PFCR1.CS5S[1:0] = 10 | SYSCR0.EXBE = 1, PFCR0.CS5E = 1, CS5CNT.EXENB = 1 |
| | | SYSC | CS6C_OE | CS6 | PFCR1.CS6S[1:0] = 10 | SYSCR0.EXBE = 1, PFCR0.CS6E = 1, CS6CNT.EXENB = 1 |
| | | SYSC | CS7C_OE | CS7 | PFCR1.CS7S[1:0] = 10 | SYSCR0.EXBE = 1, PFCR0.CS7E = 1, CS7CNT.EXENB = 1 |
| | 2 | | — | — | — | — |
| | 3 | | — | — | — | — |
| | 4 | | — | — | — | — |
| | 5 | | — | — | — | — |
| | 6 | | — | — | — | — |
| | 7 | | — | — | — | — |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| PA | 0 | TPU6 | TIOCA6_OE | TIOCA6 | | TIORH.IOA[3] = 0, TIORH.IOA[1:0] = 01/10/11 |
| | | PPG1 | PO 16_OE | PO16 | | NDERL.NDER16 = 1 |
| | | SYSC,BSC | BC0_OE | BC0 | | SYSCR0.EXBE = 1, CSnMOD.WRMOD = 1, PA.DDR.B0 = 1 |
| | | SYSC,BSC | A0_OE | A0 | | SYSCR0.EXBE = 1, CSnMOD.WRMOD = 0, PA.DDR.B0 = 1 |
| | 1 | TPU6 | TIOCB6_OE | TIOCB6 | | TIORH.IOB[3] = 0, TIORH.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO 17_OE | PO17 | | NDERL.NDER17 = 1 |
| | | SYSC | A1_OE | A1 | | SYSCR0.EXBE = 1, PA.DDR.B1 = 1 |
| | 2 | TPU6 | TIOCC6_OE | TIOCC6 | | TMDR.BFA = 0, TIORL.IOC[3] = 0, TIORL.IOC[1:0] = 01/10/11 |
| | | PPG1 | PO 18_OE | PO18 | | NDERL.NDER18 = 1 |
| | | SYSC | A2_OE | A2 | | SYSCR0.EXBE = 1, PA.DDR.B2 = 1 |
| | 3 | TPU6 | TIOCD6_OE | TIOCD6 | | TMDR.BFB = 0, TIORL.IOD[3] = 0, TIORL.IOD[1:0] = 01/10/11 |
| | | PPG1 | PO 19_OE | PO19 | | NDERL.NDER19 = 1 |
| | | SYSC | A3_OE | A3 | | SYSCR0.EXBE = 1, PA.DDR.B3 = 1 |
| | 4 | TPU7 | TIOCA7_OE | TIOCA7 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG1 | PO 20_OE | PO20 | | NDERL.NDER20 = 1 |
| | | SYSC | A4_OE | A4 | | SYSCR0.EXBE = 1, PA.DDR.B4 = 1 |
| | 5 | TPU7 | TIOCB7_OE | TIOCB7 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO 21_OE | PO21 | | NDERL.NDER21 = 1 |
| | | SYSC | A5_OE | A5 | | SYSCR0.EXBE = 1, PA.DDR.B5 = 1 |
| | 6 | TPU8 | TIOCA8_OE | TIOCA8 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG1 | PO 22_OE | PO22 | | NDERL.NDER22 = 1 |
| | | SYSC | A6_OE | A6 | | SYSCR0.EXBE = 1, PA.DDR.B6 = 1 |
| | 7 | TPU8 | TIOCB8_OE | TIOCB8 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO 23_OE | PO23 | | NDERL.NDER23 = 1 |
| | | SYSC | A7_OE | A7 | | SYSCR0.EXBE = 1, PA.DDR.B7 = 1 |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| PB | 0 | TPU9 | TIOCA9_OE | TIOCA9 | | TIORH.IOA[3] = 0, TIORH.IOA[1:0] = 01/10/11 |
| | | PPG1 | PO24_OE | PO24 | | NDERH.NDER24 = 1 |
| | | SYSC | A8_OE | A8 | | SYSCR0.EXBE = 1, PFCR4.A08E = 1 |
| | 1 | TPU9 | TIOCB9_OE | TIOCB9 | | TIORH.IOB[3] = 0, TIORH.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO25_OE | PO25 | | NDERH.NDER25 = 1 |
| | | SYSC | A9_OE | A9 | | SYSCR0.EXBE = 1, PFCR4.A09E = 1 |
| | 2 | TPU9 | TIOCC9_OE | TIOCC9 | | TMDR.BFA = 0, TIORL.IOC[3] = 0, TIORL.IOC[1:0] = 01/10/11 |
| | | PPG1 | PO26_OE | PO26 | | NDERH.NDER26 = 1 |
| | | SYSC | A10_OE | A10 | | SYSCR0.EXBE = 1, PFCR4.A10E = 1 |
| | 3 | TPU9 | TIOCD9_OE | TIOCD9 | | TMDR.BFB = 0, TIORL.IOD[3] = 0, TIORL.IOD[1:0] = 01/10/11 |
| | | PPG1 | PO27_OE | PO27 | | NDERH.NDER27 = 1 |
| | | SYSC | A11_OE | A11 | | SYSCR0.EXBE = 1, PFCR4.A11E = 1 |
| | 4 | TPU10 | TIOCA10_OE | TIOCA10 | | TIOR.IOA[3] = 0, TIOR.IOA[1:0] = 01/10/11 |
| | | PPG1 | PO28_OE | PO28 | | NDERH.NDER28 = 1 |
| | | SYSC | A12_OE | A12 | | SYSCR0.EXBE = 1, PFCR4.A12E = 1 |
| | 5 | TPU10 | TIOCB10_OE | TIOCB10 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO29_OE | PO29 | | NDERH.NDER29 = 1 |
| | | SYSC | A13_OE | A13 | | SYSCR0.EXBE = 1, PFCR4.A13E = 1 |
| | 6 | TPU11 | TIOCA11_OE | TIOCA11 | | TIOR.IOA[3] = 0, TIORIOA[1:0] = 01/10/11 |
| | | PPG1 | PO30_OE | PO30 | | NDERH.NDER30 = 1 |
| | | SYSC | A14_OE | A14 | | SYSCR0.EXBE = 1, PFCR4.A14E = 1 |
| | 7 | TPU11 | TIOCB11_OE | TIOCB11 | | TIOR.IOB[3] = 0, TIOR.IOB[1:0] = 01/10/11 |
| | | PPG1 | PO31_OE | PO31 | | NDERH.NDER31 = 1 |
| | | SYSC | A15_OE | A15 | | SYSCR0.EXBE = 1, PFCR4.A15E = 1 |



Downloaded from Arrow.com.

| Port | | Corresponding Peripheral Module | Signal Name for Setting Output | Output Signal Name | Register Setting for Selection of Signals | Setting for Each Internal Module |
|---|---|---|---|---|---|---|
| PC | 0 | SYSC | A16_OE | A16 | | SYSCR0.EXBE = 1, PFCR3 A16E = 1 |
| | 1 | SYSC | A17_OE | A17 | | SYSCR0.EXBE = 1, PFCR3.A17E = 1 |
| | 2 | SYSC | A18_OE | A18 | | SYSCR0.EXBE = 1, PFCR3.A18E = 1 |
| | 3 | SYSC | A19_OE | A19 | | SYSCR0.EXBE = 1, PFCR3.A19E = 1 |
| | 4 | SYSC | A20_OE | A20 | | SYSCR0.EXBE = 1, PFCR3.A20E = 1 |
| | 5 | SYSC | A21_OE | A21 | | SYSCR0.EXBE = 1, PFCR3.A21E = 1 |
| | | SCI5 | SCK5_OE | SCK5 | | When SCMR.SMIF = 1:<br>SMR.GM = 0, SCR.CKE[1:0] = 01 or SMR.GM = 1<br>When SCMR.SMIF = 0:<br>SMR.CM = 0, SCR.CKE[1:0] = 01 or SMR.CM = 1, SCR.CKE[1] = 0 |
| | | SYSC | CS5D_OE | CS5 | PFCR1.CS5S[1:0] = 11 | SYSCR0.EXBE = 1, PFCR0.CS5E = 1, CS5CNT.EXENB = 1 |
| | 6 | SYSC | A22_OE | A22 | | SYSCR0.EXBE = 1, PFCR3.A22E = 1, PC.DDR.B6 = 1 |
| | | SYSC | CS6D_OE | CS6 | PFCR1.CS6S[1:0] = 11 | SYSCR0.EXBE = 1, PFCR0.CS6E = 1, CS6CNT.EXENB = 1 |
| | 7 | SYSC | A23_OE | A23 | | SYSCR0.EXBE = 1, PFCR3.A23E = 1, PC.DDR.B7 = 1 |
| | | SCI5 | TxD5_OE | TxD5 | | SCR.TE = 1 |
| | | SYSC | CS4D_OE | CS4 | PFCR1.CS4S[1:0] = 11 | SYSCR0.EXBE = 1, PFCR0.CS4E = 1, CS4CNT.EXENB = 1 |
| | | SYSC | CS7D_OE | CS7 | PFCR1.CS7S[1:0] = 11 | SYSCR0.EXBE = 1, PFCR0.CS7E = 1, CS7CNT.EXENB = 1 |
| PD | 0 | SYSC | D0_E | D0 | | SYSCR0.EXBE = 1 |
| | 1 | SYSC | D1_E | D1 | | SYSCR0.EXBE = 1 |
| | 2 | SYSC | D2_E | D2 | | SYSCR0.EXBE = 1 |
| | 3 | SYSC | D3_E | D3 | | SYSCR0.EXBE = 1 |
| | 4 | SYSC | D4_E | D4 | | SYSCR0.EXBE = 1 |
| | 5 | SYSC | D5_E | D5 | | SYSCR0.EXBE = 1 |
| | 6 | SYSC | D6_E | D6 | | SYSCR0.EXBE = 1 |
| | 7 | SYSC | D7_E | D7 | | SYSCR0.EXBE = 1 |
| PE | 0 | SYSC,BSC | D8_E | D8 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 1 | SYSC,BSC | D9_E | D9 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 2 | SYSC,BSC | D10_E | D10 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 3 | SYSC,BSC | D11_E | D11 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 4 | SYSC,BSC | D12_E | D12 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 5 | SYSC,BSC | D13_E | D13 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 6 | SYSC,BSC | D14_E | D14 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |
| | 7 | SYSC,BSC | D15_E | D15 | | SYSCR0.EXBE = 1, CSnCNT.BSIZE[1:0] = 00, PFCR5.DHE = 1 |



Downloaded from Arrow.com.

**Table 14.9 Pin Functions in Each Operating Mode**

| | | P5 | | | | P6 | | P7 | PA | PB | PC | | PD | PE | PF | PG | PH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Selection of Operating Modes** | **Operating Mode** | **b7** | **b3** | **b2** | **b1, b0** | **b4 to b1** | **b0** | **b1, b0** | **b7 to b0** | **b7 to b0** | **b7 to b5** | **b4 to b0** | **b7 to b0** | **b7 to b0** | **b6 to b0** | **b7 to b0** | **b7 to b0** |
| Selection by the mode pins | Boot mode | P | P*/C | P | P | P | P | P | P | P | P | P | P | P | P | P | P |
| | User boot mode | P | P*/C | P | P | P | P | P | P | P | P | P | P | P | P | P | P |
| | Single-chip mode | P | P*/C | P | P | P | P | P | P | P | P | P | P | P | P | P | P |
| Transition by the register | Single-chip mode | P | P/C | P | P | P | P | P | P | P | P | P | P | P | P | P | P |
| | Expansion mode with on-chip ROM enabled | P/C | P/C | P/C | P/C | P/C | P/C | P/C | A | P/A | P/A/C | P/A | P/D | P/D | P | P | P |
| | Expansion mode with on-chip ROM disabled | P/C | P/C | P/C | P/C | P/C | P/C | P/C | A | P/A | P/A/C | P/A | P/D | P/D | P | P | P |

[Legend] P: I/O pin, A: Address bus output, D: Data bus output, C: Control signals, clock input/output, *: After reset



Downloaded from Arrow.com.

## 14.5 Treatment of Unused Pins

The treatment of unused pins is listed in table 14.10.

**Table 14.10 Treatment of Unused Pins**

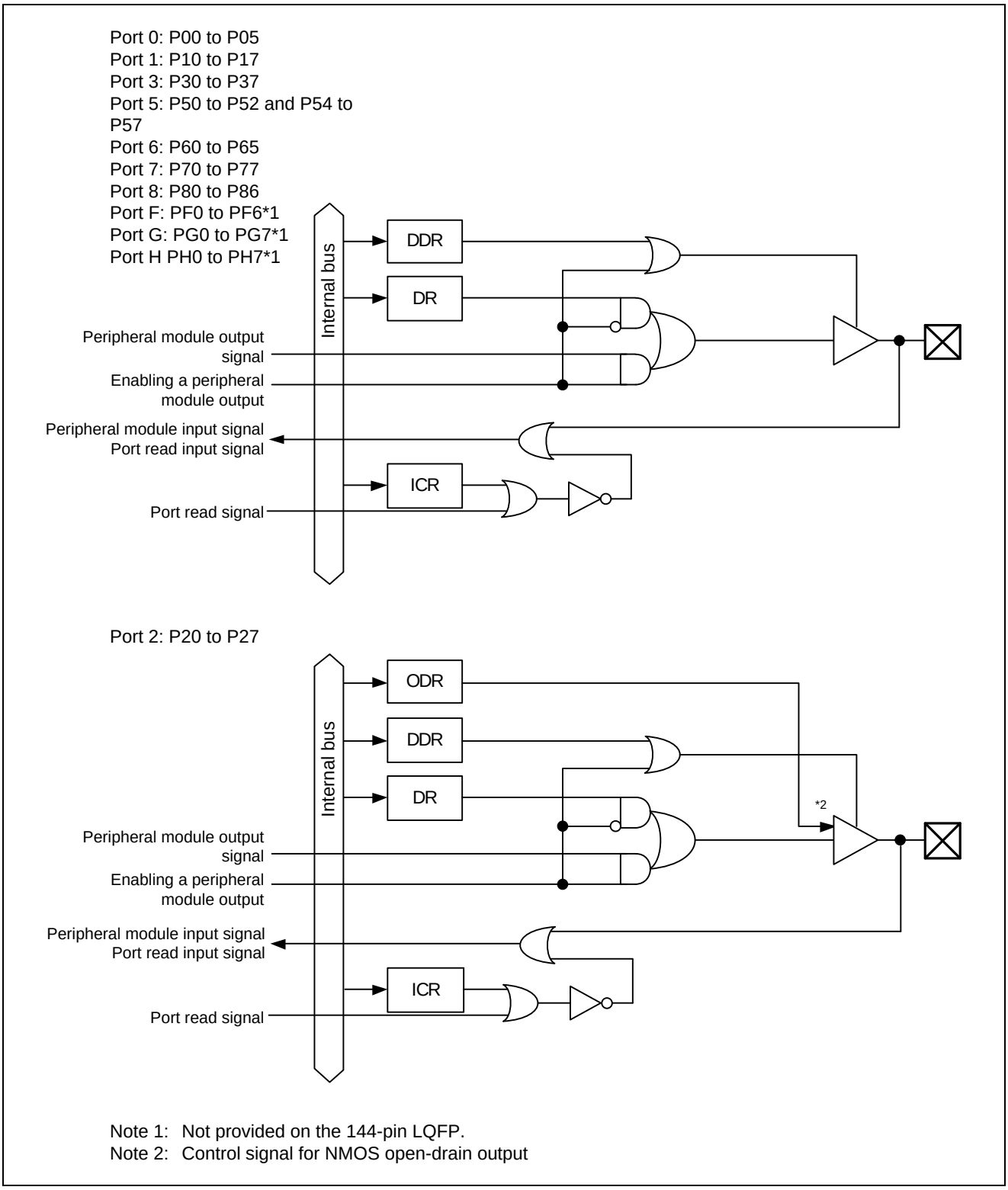
| Pin Name | Expansion Mode with On-Chip ROM Disabled (16-Bit Bus Width) | Expansion Mode with On-Chip ROM Disabled (8-Bit Bus Width) | Expansion Mode with On-Chip ROM Enabled | Single-Chip Mode |
|---|---|---|---|---|
| EMLE | Connect this pin to Vss via a pull-down resistor. | | | |
| MD1 to MD0 | (Always used as mode pins) | | | |
| MDE | (Always used as mode pins) | | | |
| NMI | Connect this pin to Vcc via a pull-up resistor. | | | |
| EXTAL | (Always used as a clock pin) | | | |
| XTAL | Leave this pin open. | | | |
| WDTOVF# | Leave this pin open. | | | |
| Ports 0 to 4, P57 to P54, P67 to P61, Ports 7 to 9, PC7 to PC5 | Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor, respectively. These pins can be left while Pm.ICR is in the initial state (the input buffer disabled)*. | | | |
| P53 | • Connect this pin to Vcc via a pull-up resistor or to Vss via a pull-down resistor.<br>• This pin can be left while Pm.ICR is in the initial state (the input buffer disabled)*. | | | • Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor, respectively.<br>• These pins can be left while Pm.ICR is in the initial state (the input buffer disabled)* |
| P52 | This pin is left open for the RD# output. | | | |
| P51 | • Connect this pin to Vcc via a pull-up resistor or to Vss via a pull-down resistor.<br>• This pin can be left while Pm.ICR is in the initial state (the input buffer disabled)*. | | | |
| P50 | This pin is left open for the WR0#/WR# output. | | | |
| P60, Ports A and B, PC4 to PC0 | • Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor.<br>• These pins can be left while Pm.ICR is in the initial state (the input buffer disabled)*. | | | |
| Port D | (Used as a data bus) | | | • Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor, respectively. |
| Port E | (Used as a data bus) | • Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor, respectively, for the general input in the initial state.<br>• These pins can be left while Pm.ICR is in the initial state (the input buffer disabled)* | | |
| Port F<br>Port G<br>Port H | • Connect these pins to Vcc via a pull-up resistor or to Vss via a pull-down resistor.<br>• These pins can be left while Pm.ICR is in the initial state (the input buffer disabled)*. | | | |
| VREFH | Connect this pin to AVcc | | | |

Note: * Do not change the initial value of Pm.ICR. Changing the initial value may generate shoot-through current.



Downloaded from Arrow.com.

## 14.6 I/O Port Configuration




Note 1: Not provided on the 144-pin LQFP.
Note 2: Control signal for NMOS open-drain output

**Figure 14.2 I/O Port Configuration (1)**



Downloaded from Arrow.com.

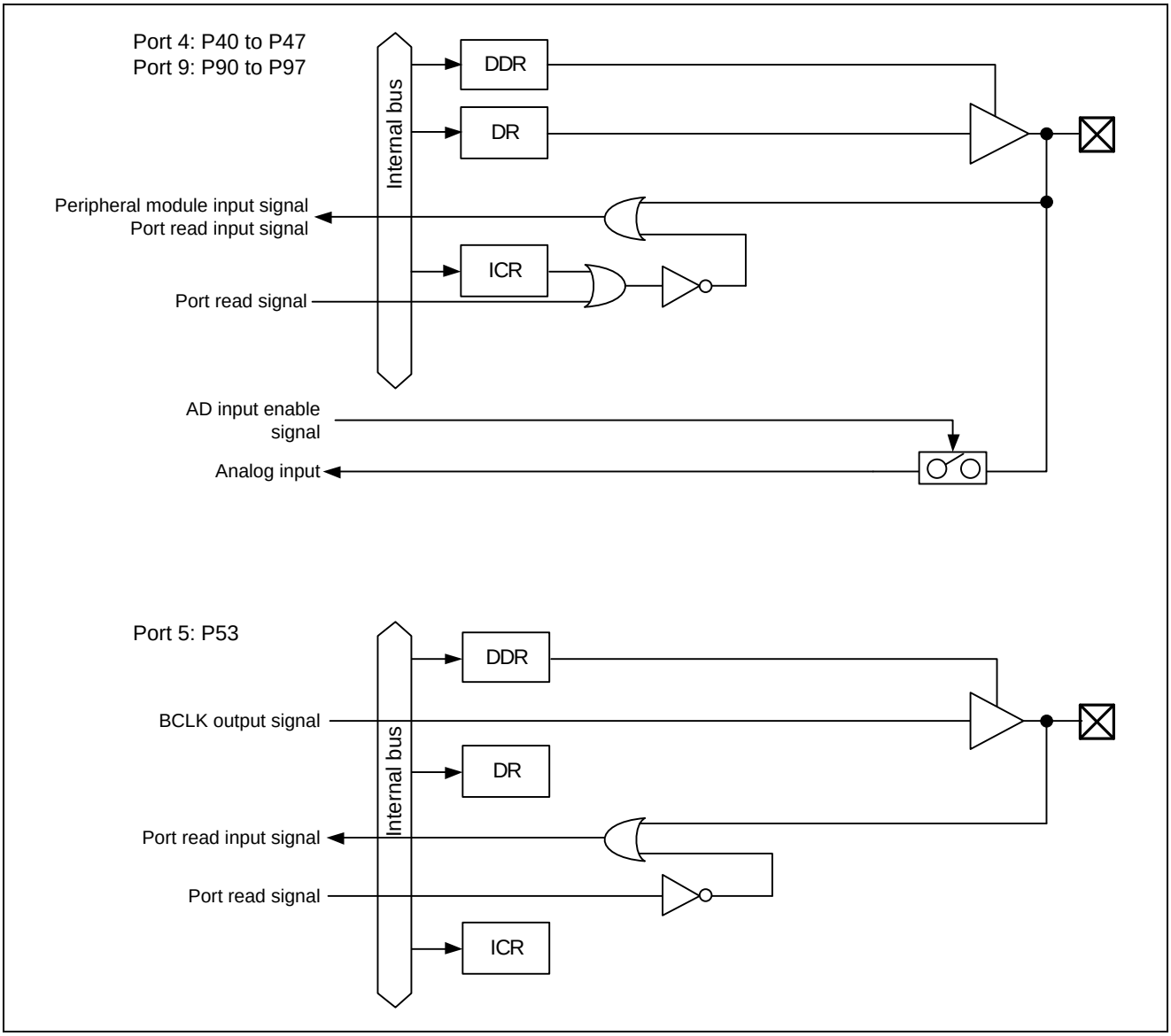



**Figure 14.3 I/O Port Configuration (2)**



Downloaded from Arrow.com.

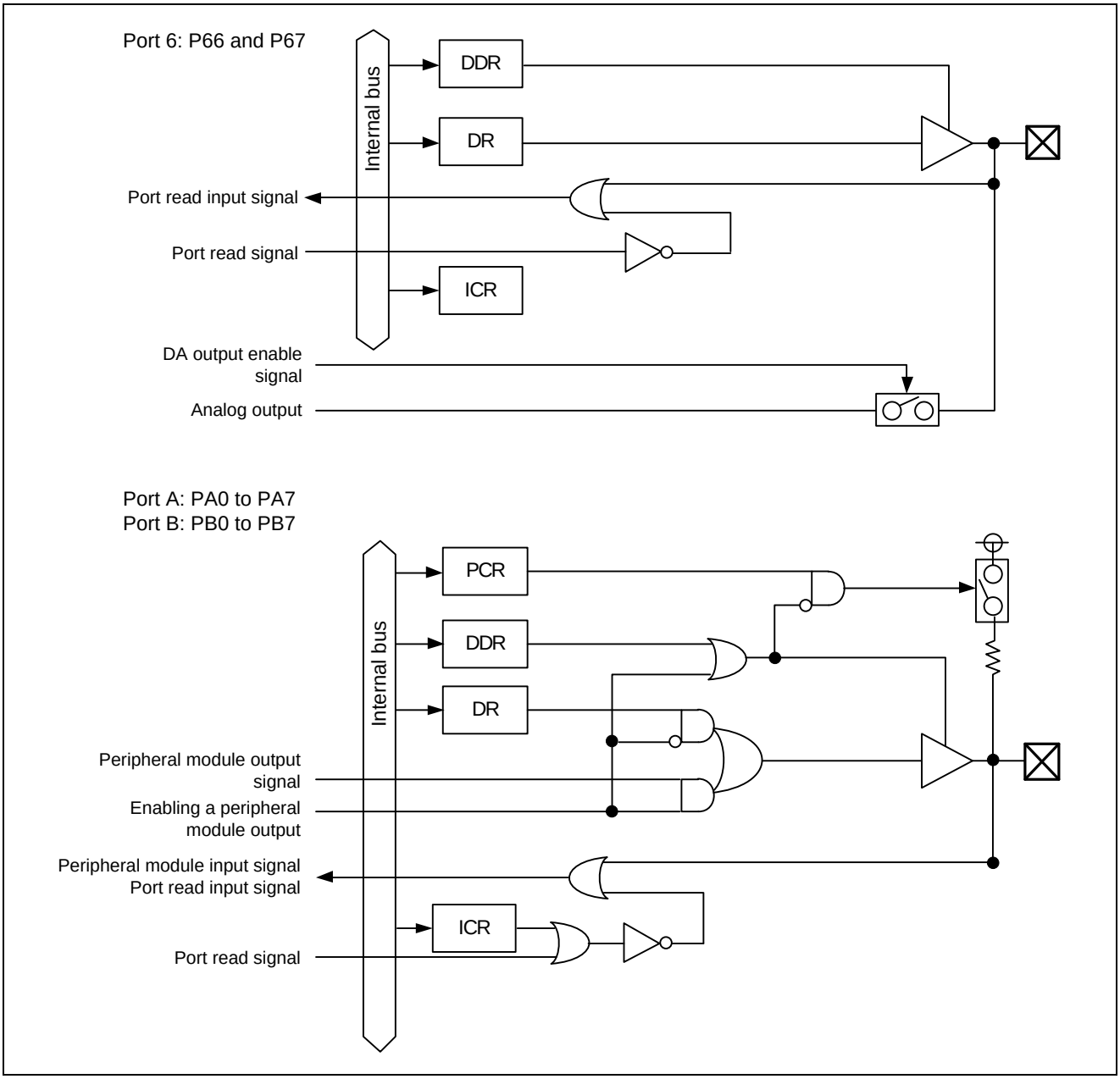



**Figure 14.4 I/O Port Configuration (3)**



Downloaded from Arrow.com.

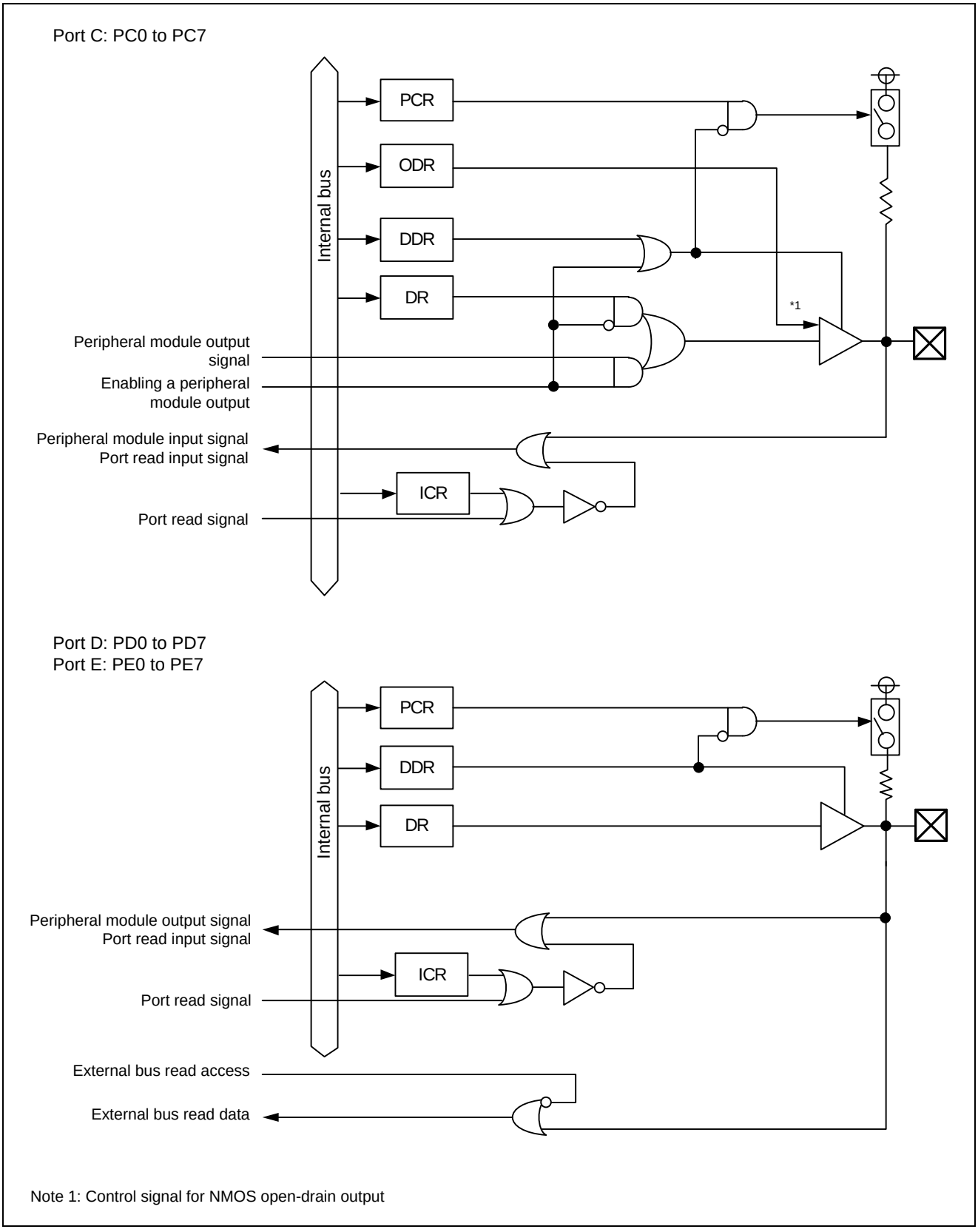



**Figure 14.5 I/O Port Configuration (4)**



Downloaded from Arrow.com.

## 14.7 Usage Notes

### 14.7.1 Setting the Input Buffer Control Register (Pm.ICR)

Changes to Pm.ICR settings can lead to the generation of internal edges, depending on the setting and the pin states at the time it is made. Change the setting of a Pm.ICR while the input signal from the pins is fixed to the high level or, if a peripheral function has been assigned to a pin, while the input function is disabled.

If a Pm.ICR setting enables an input buffer, and several input functions have been allocated to the corresponding pin, the pin state is reflected in the values of all of the input signals. Thus, even for input functions that are not in use, attention must be paid to the settings of the corresponding peripheral modules.

If a pin is being used as an output pin, the output value is taken in as the pin state when the input buffer is enabled by a Pm.ICR setting. For pins being used as output pins, disable the input buffer by setting the corresponding bits in the given Pm.ICR.

### 14.7.2 Setting the Port Function Control Register (PFCRn)

Each PFCRn controls an I/O port. When setting input or output functions for individual pins, select a pin for the input or output and then enable or disable the input or output function.

If the levels for a pin before and after it has been switched to operate as an input differ from each other, an internal edge is generated. This may lead to operation that was not intended. To avoid this, follow the procedure below when switching a pin to input operation.

(1) Disable the input in the peripheral module settings that correspond to functions of the pin to be switched.

(2) Make the PFCRn setting to select input operation for the pin.

(3) Enable the input in the peripheral module settings that correspond to functions of the pin to be switched.

A single pin function may correspond to a pin selection bit for changing a pin for the input or output and a pin enable bit for enabling a pin function. In such cases, set the pin for the input or output and then set the enable bit to enable the pin function.

### 14.7.3 Port Setting when A/D Converter Input is Used

If any of a pin in ports 4 and 9 is used as an A/D converter input, use other pins in ports 4 and 9, that are not used for A/D converter input, as input pin or interrupt input by setting P4.DDR.Bj = 0 and P9.DDR.Bj = 0.



Downloaded from Arrow.com.

# 15. 16-Bit Timer Pulse Unit (TPU)

The RX610 Group has two on-chip 16-bit timer pulse units (TPU), unit 0 and unit 1, each comprising six channels. Therefore, this LSI includes twelve channels (TPU0 to TPU11).

## 15.1 Overview

Specifications of the TPU are shown in table 15.1. Functions of TPU (unit 0) and TPU (unit 1) are shown in table 15.2 and table 15.3, respectively.

Block diagrams of TPU (unit 0) and TPU (unit 1) are shown in figure 15.1 and figure 15.2, respectively.

**Table 15.1 Specifications of TPU**

| Item | Description |
|---|---|
| Pulse input/output | Maximum 16 |
| Count clock | Seven or eight types are provided for each channel. |
| Settable operations | • Waveform output at compare match<br>• Input capture function<br>• Counter clear operation<br>• Simultaneous writing to multiple timer counters (TCNT)<br>• Simultaneous clearing by compare match and input capture<br>• Simultaneous input/output for registers by counter synchronous operation<br>• Maximum of 15-phase PWM output by combination with synchronous operation<br>• Cascaded operation |
| Channels 0 and 3 | Buffer operation can be set. |
| Channels 1, 2, 4, and 5 | Phase counting mode can be set independently. |
| Interrupt source | 26 sources |
| Buffer operation | Automatic transfer of register data |
| Generation of trigger | Programmable pulse generator (PPG) output trigger can be generated.<br>Conversion start trigger for the A/D converter can be generated. |
| Power-down function | Module stop state can be set for each unit. |



Downloaded from Arrow.com.

**Table 15.2 TPU (Unit 0) Functions**

| Item | | TPU0 | TPU1 | TPU2 | TPU3 | TPU4 | TPU5 |
|---|---|---|---|---|---|---|---|
| Count clock | | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>TCLKA<br>TCLKB<br>TCLKC<br>TCLKD | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>TCLKA<br>TCLKB | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/1024<br>TCLKA<br>TCLKB<br>TCLKC | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>PCLK/1024<br>PCLK/4096<br>TCLKA | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/1024<br>TCLKA<br>TCLKC | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>TCLKA<br>TCLKC<br>TCLKD |
| Timer general registers (TGRy) (y = A to D) | | TGRA<br>TGRB<br>TGRC*[1]<br>TGRD*[1] | TGRA<br>TGRB | TGRA<br>TGRB | TGRA<br>TGRB<br>TGRC*[1]<br>TGRD*[1] | TGRA<br>TGRB | TGRA<br>TGRB |
| I/O pins | | TIOCA0<br>TIOCB0<br>TIOCC0<br>TIOCD0 | TIOCA1<br>TIOCB1 | TIOCA2<br>TIOCB2 | TIOCA3<br>TIOCB3<br>TIOCC3<br>TIOCD3 | TIOCA4<br>TIOCB4 | TIOCA5<br>TIOCB5 |
| Counter clear function | | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture |
| Compare match output | Low output | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| | High output | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| | Toggle output | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| Input capture function | | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| Synchronous operation | | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| PWM mode | | Poss ble | Possible | Possible | Poss ble | Possible | Possible |
| Phase counting mode | | Not possible | Possible | Possible | Not possible | Possible | Possible |
| Buffer operation | | Poss ble | Not poss ble | Not possible | Poss ble | Not poss ble | Not poss ble |
| DTC activation | | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture |



Downloaded from Arrow.com.

| | | | | | | |
|---|---|---|---|---|---|---|
| DMAC activation | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture |
| A/D conversion start trigger | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture |
| | TGRA to TGRD compare match or input capture | Not poss ble | Not possible | Not possible | Not poss ble | Not poss ble |
| PPG trigger | TGRA/TGRB compare match or input capture | TGRA/TGRB compare match or input capture | TGRA/TGRB compare match or input capture | TGRA/TGRB compare match or input capture | Not poss ble | Not poss ble |
| Interrupt sources | 5 sources<br>• Compare match or input capture 0A<br>• Compare match or input capture 0B<br>• Compare match or input capture 0C<br>• Compare match or input capture 0D<br>• Overflow | 4 sources<br>• Compare match or input capture 1A<br>• Compare match or input capture 1B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 2A<br>• Compare match or input capture 2B<br>• Overflow<br>• Underflow | 5 sources<br>• Compare match or input capture 3A<br>• Compare match or input capture 3B<br>• Compare match or input capture 3C<br>• Compare match or input capture 3D<br>• Overflow | 4 sources<br>• Compare match or input capture 4A<br>• Compare match or input capture 4B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 5A<br>• Compare match or input capture 5B<br>• Overflow<br>• Underflow |
| Module stop setting*2 | MSTPA13 bit in MSTPCRA | | | | | |

Notes: 1. TGRC and TGRD can be set as a buffer register.

2. For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

**Table 15.3 TPU (Unit 1) Functions**

| Item | | TPU6 | TPU7 | TPU8 | TPU9 | TPU10 | TPU11 |
|---|---|---|---|---|---|---|---|
| Count clock | | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>TCLKE<br>TCLKF<br>TCLKG<br>TCLKH | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>TCLKE<br>TCLKF | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/1024<br>TCLKE<br>TCLKF<br>TCLKG | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>PCLK/1024<br>PCLK/4096<br>TCLKE | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/1024<br>TCLKE<br>TCLKG | PCLK/1<br>PCLK/4<br>PCLK/16<br>PCLK/64<br>PCLK/256<br>TCLKE<br>TCLKG<br>TCLKH |
| Timer general registers (TGRy) (y = A to D) | | TGRA<br>TGRB<br>TGRC*1<br>TGRD*1 | TGRA<br>TGRB | TGRA<br>TGRB | TGRA<br>TGRB<br>TGRC*1<br>TGRD*1 | TGRA<br>TGRB | TGRA<br>TGRB |
| I/O pins | | TIOCA6<br>TIOCB6<br>TIOCC6<br>TIOCD6 | TIOCA7<br>TIOCB7 | TIOCA8<br>TIOCB8 | TIOCA9<br>TIOCB9<br>TIOCC9<br>TIOCD9 | TIOCA10<br>TIOCB10 | TIOCA11<br>TIOCB11 |
| Counter clear function | | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture |
| Compare match output | Low output | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| | High output | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| | Toggle output | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| Input capture function | | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| Synchronous operation | | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| PWM mode | | Poss ble | Possible | Possible | Possible | Possible | Poss ble |
| Phase counting mode | | Not possible | Possible | Possible | Not possible | Possible | Poss ble |
| Buffer operation | | Poss ble | Not poss ble | Not possible | Possible | Not possible | Not possible |
| DTC activation | | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture | TGRy compare match or input capture |



Downloaded from Arrow.com.

| | | | | | | |
|---|---|---|---|---|---|---|
| DMAC activation | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture |
| A/D conversion start trigger | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture | TGRA compare match or input capture |
| PPG trigger | TGRA/TGRB compare match | TGRA/TGRB compare match | TGRA/TGRB compare match | TGRA/TGRB compare match | Not possible | Not possible |
| Interrupt sources | 5 sources<br>• Compare match or input capture 6A<br>• Compare match or input capture 6B<br>• Compare match or input capture 6C<br>• Compare match or input capture 6D<br>• Overflow | 4 sources<br>• Compare match or input capture 7A<br>• Compare match or input capture 7B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 8A<br>• Compare match or input capture 8B<br>• Overflow<br>• Underflow | 5 sources<br>• Compare match or input capture 9A<br>• Compare match or input capture 9B<br>• Compare match or input capture 9C<br>• Compare match or input capture 9D<br>• Overflow | 4 sources<br>• Compare match or input capture 10A<br>• Compare match or input capture 10B<br>• Overflow<br>• Underflow | 4 sources<br>• Compare match or input capture 11A<br>• Compare match or input capture 11B<br>• Overflow<br>• Underflow |
| Module stop setting*2 | MSTPA12 bit in MSTPCRA | | | | | |

Notes: 1. TGRC and TGRD can be set as a buffer register.

2. For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

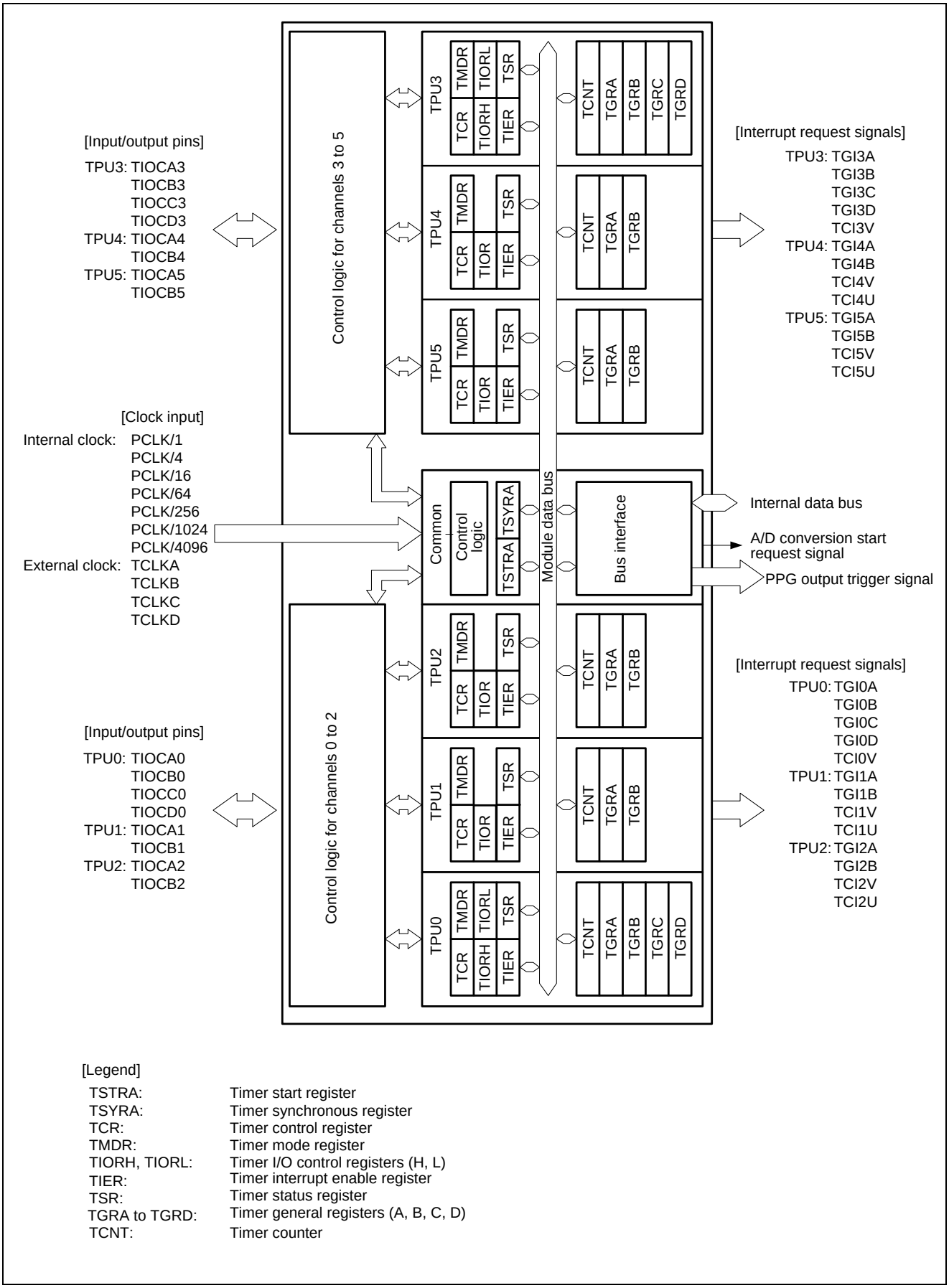



**Figure 15.1 Block Diagram of TPU (Unit 0)**



Downloaded from Arrow.com.

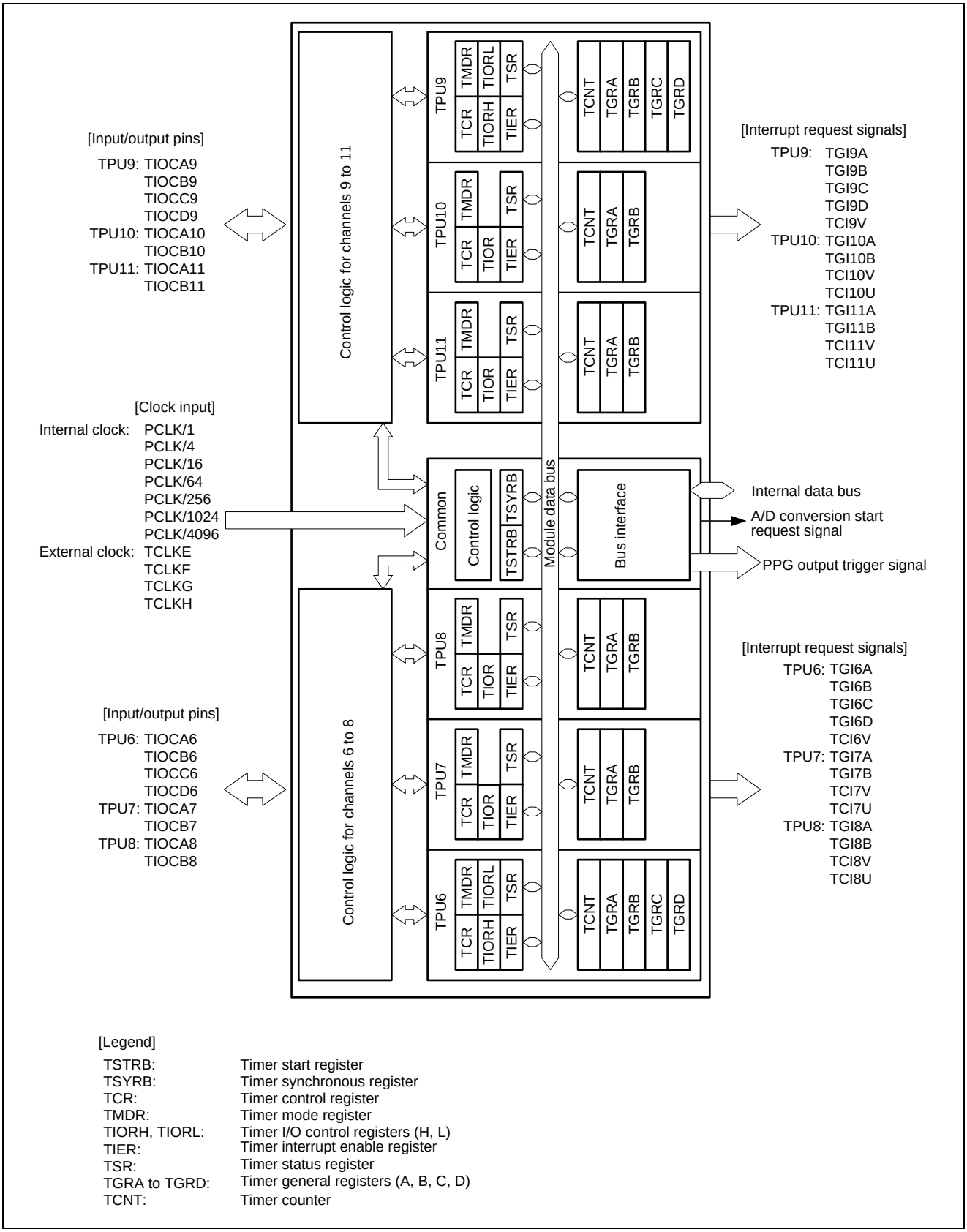



**Figure 15.2 Block Diagram of TPU (Unit 1)**



Downloaded from Arrow.com.

Table 15.4 lists the input/output pins of the TPU.

**Table 15.4 Pin Configuration of TPU**

| Unit | Channel | Pin Name | I/O | Description |
|---|---|---|---|---|
| Unit 0 | All | TCLKA | Input | External clock A input pin (TPU1 and TPU5 phase counting mode A phase input) |
| | | TCLKB | Input | External clock B input pin (TPU1 and TPU5 phase counting mode B phase input) |
| | | TCLKC | Input | External clock C input pin (TPU2 and TPU4 phase counting mode A phase input) |
| | | TCLKD | Input | External clock D input pin (TPU2 and TPU4 phase counting mode B phase input) |
| | TPU0 | TIOCA0 | I/O | TPU0.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB0 | I/O | TPU0.TGRB input capture input/output compare output/PWM output pin |
| | | TIOCC0 | I/O | TPU0.TGRC input capture input/output compare output/PWM output pin |
| | | TIOCD0 | I/O | TPU0.TGRD input capture input/output compare output/PWM output pin |
| | TPU1 | TIOCA1 | I/O | TPU1.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB1 | I/O | TPU1.TGRB input capture input/output compare output/PWM output pin |
| | TPU2 | TIOCA2 | I/O | TPU2.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB2 | I/O | TPU2.TGRB input capture input/output compare output/PWM output pin |
| | TPU3 | TIOCA3 | I/O | TPU3.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB3 | I/O | TPU3.TGRB input capture input/output compare output/PWM output pin |
| | | TIOCC3 | I/O | TPU3.TGRC input capture input/output compare output/PWM output pin |
| | | TIOCD3 | I/O | TPU3.TGRD input capture input/output compare output/PWM output pin |
| | TPU4 | TIOCA4 | I/O | TPU4.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB4 | I/O | TPU4.TGRB input capture input/output compare output/PWM output pin |
| | TPU5 | TIOCA5 | I/O | TPU5.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB5 | I/O | TPU5.TGRB input capture input/output compare output/PWM output pin |
| Unit 1 | All | TCLKE | Input | External clock E input pin (TPU7 and TPU11 phase counting mode A phase input) |
| | | TCLKF | Input | External clock F input pin (TPU7 and TPU11 phase counting mode B phase input) |
| | | TCLKG | Input | External clock G input pin (TPU8 and TPU10 phase counting mode A phase input) |
| | | TCLKH | Input | External clock H input pin (TPU8 and TPU10 phase counting mode B phase input) |
| | TPU6 | TIOCA6 | I/O | TPU6.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB6 | I/O | TPU6.TGRB input capture input/output compare output/PWM output pin |
| | | TIOCC6 | I/O | TPU6.TGRC input capture input/output compare output/PWM output pin |
| | | TIOCD6 | I/O | TPU6.TGRD input capture input/output compare output/PWM output pin |
| | TPU7 | TIOCA7 | I/O | TPU7.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB7 | I/O | TPU7.TGRB input capture input/output compare output/PWM output pin |
| | TPU8 | TIOCA8 | I/O | TPU8.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB8 | I/O | TPU8.TGRB input capture input/output compare output/PWM output pin |
| | TPU9 | TIOCA9 | I/O | TPU9.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB9 | I/O | TPU9.TGRB input capture input/output compare output/PWM output pin |
| | | TIOCC9 | I/O | TPU9.TGRC input capture input/output compare output/PWM output pin |
| | | TIOCD9 | I/O | TPU9.TGRD input capture input/output compare output/PWM output pin |
| | TPU10 | TIOCA10 | I/O | TPU10.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB10 | I/O | TPU10.TGRB input capture input/output compare output/PWM output pin |
| | TPU11 | TIOCA11 | I/O | TPU11.TGRA input capture input/output compare output/PWM output pin |
| | | TIOCB11 | I/O | TPU11.TGRB input capture input/output compare output/PWM output pin |



Downloaded from Arrow.com.

## 15.2 Register Descriptions

Table 15.5 lists the registers of the TPU.

**Table 15.5 Registers of TPU**

| Unit | Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| Unit 0 | TPU0 | Timer control register | TCR | 00h | 0008 8110h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8111h | 8 |
| | | Timer I/O control register H | TIORH | 00h | 0008 8112h | 8 |
| | | Timer I/O control register L | TIORL | 00h | 0008 8113h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8114h | 8 |
| | | Timer status register | TSR | xxh | 0008 8115h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8116h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8118h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 811Ah | 16 |
| | | Timer general register C | TGRC | FFFFh | 0008 811Ch | 16 |
| | | Timer general register D | TGRD | FFFFh | 0008 811Eh | 16 |
| | TPU1 | Timer control register | TCR | 00h | 0008 8120h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8121h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 8122h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8124h | 8 |
| | | Timer status register | TSR | xxh | 0008 8125h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8126h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8128h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 812Ah | 16 |
| | TPU2 | Timer control register | TCR | 00h | 0008 8130h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8131h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 8132h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8134h | 8 |
| | | Timer status register | TSR | xxh | 0008 8135h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8136h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8138h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 813Ah | 16 |
| | TPU3 | Timer control register | TCR | 00h | 0008 8140h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8141h | 8 |
| | | Timer I/O control register H | TIORH | 00h | 0008 8142h | 8 |
| | | Timer I/O control register L | TIORL | 00h | 0008 8143h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8144h | 8 |
| | | Timer status register | TSR | xxh | 0008 8145h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8146h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8148h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 814Ah | 16 |
| | | Timer general register C | TGRC | FFFFh | 0008 814Ch | 16 |
| | | Timer general register D | TGRD | FFFFh | 0008 814Eh | 16 |



Downloaded from Arrow.com.

| Unit | Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| Unit 0 | TPU4 | Timer control register | TCR | 00h | 0008 8150h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8151h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 8152h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8154h | 8 |
| | | Timer status register | TSR | xxh | 0008 8155h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8156h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8158h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 815Ah | 16 |
| | TPU5 | Timer control register | TCR | 00h | 0008 8160h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8161h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 8162h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8164h | 8 |
| | | Timer status register | TSR | xxh | 0008 8165h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8166h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8168h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 816Ah | 16 |
| | All | Timer start register | TSTRA | 00h | 0008 8100h | 8 |
| | | Timer synchronous register | TSYRA | 00h | 0008 8101h | 8 |
| Unit 1 | TPU6 | Timer control register | TCR | 00h | 0008 8180h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8181h | 8 |
| | | Timer I/O control register H | TIORH | 00h | 0008 8182h | 8 |
| | | Timer I/O control register L | TIORL | 00h | 0008 8183h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8184h | 8 |
| | | Timer status register | TSR | xxh | 0008 8185h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8186h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8188h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 818Ah | 16 |
| | | Timer general register C | TGRC | FFFFh | 0008 818Ch | 16 |
| | | Timer general register D | TGRD | FFFFh | 0008 818Eh | 16 |
| | TPU7 | Timer control register | TCR | 00h | 0008 8190h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 8191h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 8192h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 8194h | 8 |
| | | Timer status register | TSR | xxh | 0008 8195h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 8196h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 8198h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 819Ah | 16 |



Downloaded from Arrow.com.

| Unit | Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| Unit 1 | TPU8 | Timer control register | TCR | 00h | 0008 81A0h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 81A1h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 81A2h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 81A4h | 8 |
| | | Timer status register | TSR | xxh | 0008 81A5h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 81A6h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 81A8h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 81AAh | 16 |
| | TPU9 | Timer control register | TCR | 00h | 0008 81B0h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 81B1h | 8 |
| | | Timer I/O control register H | TIORH | 00h | 0008 81B2h | 8 |
| | | Timer I/O control register L | TIORL | 00h | 0008 81B3h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 81B4h | 8 |
| | | Timer status register | TSR | xxh | 0008 81B5h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 81B6h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 81B8h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 81BAh | 16 |
| | | Timer general register C | TGRC | FFFFh | 0008 81BCh | 16 |
| | | Timer general register D | TGRD | FFFFh | 0008 81BEh | 16 |
| | TPU10 | Timer control register | TCR | 00h | 0008 81C0h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 81C1h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 81C2h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 81C4h | 8 |
| | | Timer status register | TSR | xxh | 0008 81C5h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 81C6h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 81C8h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 81CAh | 16 |
| | TPU11 | Timer control register | TCR | 00h | 0008 81D0h | 8 |
| | | Timer mode register | TMDR | 00h | 0008 81D1h | 8 |
| | | Timer I/O control register | TIOR | 00h | 0008 81D2h | 8 |
| | | Timer interrupt enable register | TIER | 40h | 0008 81D4h | 8 |
| | | Timer status register | TSR | xxh | 0008 81D5h | 8 |
| | | Timer counter | TCNT | 0000h | 0008 81D6h | 16 |
| | | Timer general register A | TGRA | FFFFh | 0008 81D8h | 16 |
| | | Timer general register B | TGRB | FFFFh | 0008 81DAh | 16 |
| | All | Timer start register | TSTRB | 00h | 0008 8170h | 8 |
| | | Timer synchronous register | TSYRB | 00h | 0008 8171h | 8 |



Downloaded from Arrow.com.

## 15.2.1 Timer Control Register (TCR)

Addresses: TPU0.TCR 0008 8110h, TPU1.TCR 0008 8120h, TPU2.TCR 0008 8130h
TPU3.TCR 0008 8140h, TPU4.TCR 0008 8150h, TPU5.TCR 0008 8160h
TPU6.TCR 0008 8180h, TPU7.TCR 0008 8190h, TPU8.TCR 0008 81A0h
TPU9.TCR 0008 81B0h, TPU10.TCR 0008 81C0h, TPU11.TCR 0008 81D0h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | TPSC[2:0] | Timer Prescaler Select | See tables 15.6 to 15.11. | R/W |
| b4, b3 | CKEG[1:0] | Input Clock Edge Select | See table 15.12. | R/W |
| b7 to b5 | CCLR[2:0]* | Counter Clear Source Select | See tables 15.13 and 15.14. | R/W |

Note: * Bit 7 in TCR of TPU1, TPU2, TPU4, and TPU5 of unit 0 and bit 7 in TCR of TPU7, TPU8, TPU10, and TPU11 of unit 1 are reserved. These bits are read as 0. The write value should always be 0.

The TPU has twelve TCR registers, one for each channel.

TPUm.TCR controls TPUm.TCNT counter of each channel.

TPUm.TCR settings should be made while TPUm.TCNT counter operation is stopped.

### TPSC[2:0] Bits (Timer Prescaler Select)

These bits select the TCNT counter clock. The clock source can be selected independently for each channel.

To select the external clock as the clock source, set the bit in the data direction register (DDR) for the corresponding pin to 0 (input port), and set the bit in the input buffer control register (ICR) to 1 (input buffer of the corresponding pin is enabled). For details, see section 14, I/O Ports.

### CKEG[1:0] Bits (Input Clock Edge Select)

These bits select the input clock edge.

When the internal clock is counted using both edges, the input clock period is halved (e.g. PCLK/4 both edges = PCLK/2 rising edge).

Internal clock edge selection is valid when the input clock is PCLK/4 or slower. This setting is ignored if the input clock is PCLK/1, or when overflow/underflow of another channel is selected.

### CCLR[2:0] Bits (Counter Clear Source Select)

These bits select the TCNT counter clearing source.



Downloaded from Arrow.com.

**Table 15.6 Bits TPSC[2:0] (TPU0, TPU6)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU0 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU6 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | External clock: counts on TCLKB or TCLKF pin input |
| | 1 | 1 | 0 | External clock: counts on TCLKC or TCLKG pin input |
| | 1 | 1 | 1 | External clock: counts on TCLKD or TCLKH pin input |

**Table 15.7 Bits TPSC[2:0] (TPU1, TPU7)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU1 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU7 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | External clock: counts on TCLKB or TCLKF pin input |
| | 1 | 1 | 0 | Internal clock: counts on PCLK/256 |
| | 1 | 1 | 1 | • TPU1 (unit 0)<br>Counts on TPU2.TCNT counter overflow/underflow<br>• TPU7 (unit 1)<br>Counts on TPU8.TCNT counter overflow/underflow |

Note: This setting is invalid when TPU1 or TPU7 is in phase counting mode.

**Table 15.8 Bits TPSC[2:0] (TPU2, TPU8)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU2 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU8 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | External clock: counts on TCLKB or TCLKF pin input |
| | 1 | 1 | 0 | External clock: counts on TCLKC or TCLKG pin input |
| | 1 | 1 | 1 | Internal clock: counts on PCLK/1024 |

Note: This setting is invalid when TPU2 or TPU8 is in phase counting mode.



Downloaded from Arrow.com.

**Table 15.9 Bits TPSC[2:0] (TPU3, TPU9)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU3 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU9 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | Internal clock: counts on PCLK/1024 |
| | 1 | 1 | 0 | Internal clock: counts on PCLK/256 |
| | 1 | 1 | 1 | Internal clock: counts on PCLK/4096 |

**Table 15.10 Bits TPSC[2:0] (TPU4, TPU10)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU4 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU10 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | External clock: counts on TCLKC or TCLKG pin input |
| | 1 | 1 | 0 | Internal clock: counts on PCLK/1024 |
| | 1 | 1 | 1 | • TPU4 (unit 0)<br>Counts on TPU5.TCNT counter overflow/underflow<br>• TPU10 (unit 1)<br>Counts on TPU11.TCNT counter overflow/underflow |

Note: This setting is invalid when TPU4 or TPU10 is in phase counting mode.

**Table 15.11 Bits TPSC[2:0] (TPU5, TPU11)**

| | Bits TPSC[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b2** | **b1** | **b0** | **Description** |
| TPU5 (unit 0) | 0 | 0 | 0 | Internal clock: counts on PCLK/1 |
| TPU11 (unit 1) | 0 | 0 | 1 | Internal clock: counts on PCLK/4 |
| | 0 | 1 | 0 | Internal clock: counts on PCLK/16 |
| | 0 | 1 | 1 | Internal clock: counts on PCLK/64 |
| | 1 | 0 | 0 | External clock: counts on TCLKA or TCLKE pin input |
| | 1 | 0 | 1 | External clock: counts on TCLKC or TCLKG pin input |
| | 1 | 1 | 0 | Internal clock: counts on PCLK/256 |
| | 1 | 1 | 1 | External clock: counts on TCLKD pin input |

Note: This setting is invalid when TPU5 or TPU11 is in phase counting mode.



Downloaded from Arrow.com.

**Table 15.12 Bits CKEG[1:0]**

| Bits CKEG[1:0] | | Input Clock | |
|---|---|---|---|
| **b4** | **b3** | **Internal Clock** | **External clock** |
| 0 | 0 | Counted at falling edge | Counted at rising edge |
| 0 | 1 | Counted at rising edge | Counted at falling edge |
| 1 | 0 | Counted at both edges | Counted at both edges |
| 1 | 1 | Counted at both edges | Counted at both edges |

**Table 15.13 Bits CCLR[2:0] (TPU0, TPU3, TPU6, TPU9)**

| | Bits CCLR[2:0] | | | |
|---|---|---|---|---|
| **Channel** | **b7** | **b6** | **b5** | **Description** |
| (Unit 0) TPU0, TPU3 | 0 | 0 | 0 | TCNT counter clearing disabled |
| | 0 | 0 | 1 | TCNT counter cleared by TGRA compare match/input capture |
| | 0 | 1 | 0 | TCNT counter cleared by TGRB compare match/input capture |
| (Unit 1) TPU6, TPU9 | 0 | 1 | 1 | TCNT counter cleared by counter clearing for another channel performing synchronous clearing/synchronous operation*1 |
| | 1 | 0 | 0 | TCNT counter clearing disabled |
| | 1 | 0 | 1 | TCNT counter cleared by TGRC compare match/input capture*1 |
| | 1 | 1 | 0 | TCNT counter cleared by TGRD compare match/input capture*1 |
| | 1 | 1 | 1 | TCNT counter cleared by counter clearing for another channel performing synchronous clearing/synchronous operation*2 |

Notes: 1. When TGRC or TGRD is used as a buffer register, TCNT counter is not cleared because the buffer register setting has priority, and compare match/input capture does not occur.

2. Synchronous operation is selected by setting the SYNCi bit (i = 0, 3) bit in TSYRm (m = A, B) to 1.

**Table 15.14 Bits CCLR[2:0] (TPU1, TPU2, TPU4, TPU5, TPU7, TPU8, TPU10, TPU11)**

| | Bits CCLR[2:0]*1 | | | |
|---|---|---|---|---|
| **Channel** | **b7** | **b6** | **b5** | **Description** |
| (Unit 0) TPU1, TPU2, TPU4, TPU5 | 0 | 0 | 0 | TCNT counter clearing disabled |
| | 0 | 0 | 1 | TCNT counter cleared by TGRA compare match/input capture |
| | 0 | 1 | 0 | TCNT counter cleared by TGRB compare match/input capture |
| (Unit 1) TPU7, TPU8, TPU10, TPU11 | 0 | 1 | 1 | TCNT counter cleared by counter clearing for another channel performing synchronous clearing/synchronous operation*2 |
| | 1 | 0 | 0 | Setting prohibited |
| | 1 | 0 | 1 | Setting prohibited |
| | 1 | 1 | 0 | Setting prohibited |
| | 1 | 1 | 1 | Setting prohibited |

Notes: 1. Bit 7 in TCR of TPU1, TPU2, TPU4, and TPU5 of unit 0 and bit 7 in TCR of TPU7, TPU8, TPU10, and TPU11 of unit 1 are reserved. These bits are read as 0. The write value should always be 0.

2. Synchronous operation is selected by setting the SYNCi bit (i = 1, 2, 4, 5) bit in TSYRm (m = A, B) to 1.



Downloaded from Arrow.com.

## 15.2.2 Timer Mode Register (TMDR)

Addresses: TPU0.TMDR 0008 8111h, TPU1.TMDR 0008 8121h, TPU2.TMDR 0008 8131h
TPU3.TMDR 0008 8141h, PU4.TMDR 0008 8151h, TPU5.TMDR 0008 8161h
TPU6.TMDR 0008 8181h, TPU7.TMDR 0008 8191h, TPU8.TMDR 0008 81A1h
TPU9.TMDR 0008 81B1h, TPU10.TMDR 0008 81C1h, TPU11.TMDR 0008 81D1h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ICSELD | ICSELB | BFB | BFA | MD[3:0] | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | MD[3:0] | Mode Select | b3*[1] b0<br>0 0 0 0: Normal operation<br>0 0 0 1: Setting prohibited<br>0 0 1 0: PWM mode 1<br>0 0 1 1: PWM mode 2<br>0 1 0 0: Phase counting mode 1*[2]<br>0 1 0 1: Phase counting mode 2*[2]<br>0 1 1 0: Phase counting mode 3*[2]<br>0 1 1 1: Phase counting mode 4*[2]<br>(Settings other than above are prohibited.) | R/W |
| b4 | BFA*[3] | Buffer Operation A | 0: TPUm.TGRA operates normally<br>1: TPUm.TGRA and TPUm.TGRC used together for buffer operation (n = 0, 3, 6, 9) | R/W |
| b5 | BFB*[4] | Buffer Operation B | 0: TPUm.TGRB operates normally<br>1: TPUm.TGRB and TPUm.TGRD used together for buffer operation (n = 0, 3, 6, 9) | R/W |
| b6 | ICSELB | TGRB Input Capture Input Select | 0: Input capture input source is TIOCBn pin<br>1: Input capture input source is TIOCAn pin<br>(n = 0 to 11) | R/W |
| b7 | ICSELD*[4] | TGRD Input Capture Input Select | 0: Input capture input source is TIOCDn pin<br>1: Input capture input source is TIOCCn pin<br>(n = 0, 3, 6, 9) | R/W |

Notes: 1. Bit 3 is reserved. This bit is always read as 0. The write value should always be 0.

2. Phase counting mode cannot be set for TPU0, TPU3 (unit 0), TPU6, and TPU9 (unit 1). A 0 should always be written to bit 2 for them.

3. Bit 4 of TPU1, TPU2, TPU4, TPU5 (unit 0), TPU7, TPU8, TPU10, and TPU11 (unit 1) that do not have TGRC is reserved. This bit is always read as 0. The write value should always be 0.

4. Bits 5 and 7 of TPU1, TPU2, TPU4, TPU5 (unit 0), TPU7, TPU8, TPU10, and TPU11 (unit 1) that do not have TGRD are reserved. These bits are always read as 0. The write value should always be 0.

The TPU has twelve TMDR registers, one for each channel.

TPUm.TMDR sets the operating mode for each channel.

TPUm.TMDR settings should be made while TPUm.TCNT counter operation is stopped.



Downloaded from Arrow.com.

#### MD[3:0] Bits (Mode Select)

Set the timer operating mode.

#### BFA Bit (Buffer Operation A)

Specifies whether TPUm.TGRA (m = 0, 3, 6, 9) is to normally operate, or TPUm.TGRA and TPUm.TGRC (m = 0, 3, 6, 9) are to be used together for buffer operation.

When TGRC is used as a buffer register, TGRC input capture/output compare is not generated.

#### BFB Bit (Buffer Operation B)

Specifies whether TPUm.TGRB (m = 0, 3, 6, 9) is to normally operate, or TPUm.TGRB and TPUm.TGRD (m = 0, 3, 6, 9) are to be used together for buffer operation.

When TGRD is used as a buffer register, TGRD input capture/output compare is not generated.

#### ICSELB Bit (TGRB Input Capture Input Select)

Selects the input capture input for TPUm.TGRB (m = 0 to 11).

This function allows measurement of high-level width and period of the input pulse on a TIOCAn input pin.

#### ICSELD (TGRD Input Capture Input Select)

Selects the input capture input for TPUm.TGRD (m = 0, 3, 6, 9).

This function allows measurement of high-level width and period of the input pulse on a TIOCCn input pin.



Downloaded from Arrow.com.

### 15.2.3 Timer I/O Control Register (TIORH, TIORL, TIOR)

- Unit 0 (TPU0.TIORH, TPU1.TIOR, TPU2.TIOR, TPU3.TIORH, TPU4.TIOR, TPU5.TIOR)
  Unit 1 (TPU6.TIORH, TPU7.TIOR, TPU8.TIOR, TPU9.TIORH, TPU10.TIOR, TPU11.TIOR)

Addresses: TPU0.TIORH 0008 8112h, TPU1.TIOR 0008 8122h, TPU2.TIOR 0008 8132h
TPU3.TIORH 0008 8142h, TPU4.TIOR 0008 8152h, TPU5.TIOR 0008 8162h
TPU6.TIORH 0008 8182h, TPU7.TIOR 0008 8192h, TPU8.TIOR 0008 81A2h
TPU9.TIORH 0008 81B2h, TPU10.TIOR 0008 81C2h, TPU11.TIOR 0008 81D2h




Value after reset: 0 0 0 0 0 0 0 0

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | IOA[3:0] | TGRA Control | See tables 15.15 to 15.20. | R/W |
| b7 to b4 | IOB[3:0] | TGRB Control | See tables 15.15 to 15.20. | R/W |

- Unit 0 (TPU0.TIORL, TPU3.TIORL)
  Unit 1 (TPU6.TIORL, TPU9.TIORL)

Addresses: TPU0.TIORL 0008 8113h, TPU3.TIORL 0008 8143h
TPU6.TIORL 0008 8183h, TPU9.TIORL 0008 81B3h




Value after reset: 0 0 0 0 0 0 0 0

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | IOC[3:0] | TGRC Control | See tables 15.21 and 15.22. | R/W |
| b7 to b4 | IOD[3:0] | TGRD Control | See tables 15.21 and 15.22. | R/W |

The TPU has four TIORH registers, one for TPU0, TPU3, TPU6, and TPU9, and four TIORL registers, one for TPU0, TPU3, TPU6, and TPU9, and also has eight TIOR registers, one for TPU1, TPU2, TPU4, TPU5, TPU7, TPU8, TPU10, and TPU11. Thus the TPU has sixteen timer I/O control registers in total.

TIORH, TIORL, and TIOR control registers TGRA, TGRB, TGRC, and TGRD.

Note that TIORH, TIORL, and TIOR are affected by the TMDR setting.

The initial output specified by TIORH, TIORL, and TIOR is valid when the counter is stopped (the CSTj bit (j = 0 to 5) in TSTRm (m = A, B) is cleared to 0). In PWM mode 2, the output at the time when the TCNT counter is cleared to 0 is specified.

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

To specify the input capture pin in TIORH, TIORL, or TIOR, set the bit in the data direction register (DDR) for the corresponding pin to 0 (input port), and set the bit in the input buffer control register (ICR) to 1 (input buffer of the corresponding pin is enabled). For details, see section 14, I/O Ports.



Downloaded from Arrow.com.

**IOA[3:0] Bits (TGRA Control)**

Select the function of TPUm.TGRA (m = 0 to 11).

**IOB[3:0] Bits (TGRB Control)**

Select the function of TPUm.TGRB (m = 0 to 11).

**IOC[3:0] Bits (TGRC Control)**

Select the function of TPUm.TGRC (m = 0, 3, 6, 9).

**IOD[3:0] Bits (TGRD Control)**

Select the function of TPUm.TGRD (m = 0, 3, 6, 9).



Downloaded from Arrow.com.

**Table 15.15 TPU0.TIORH, TPU6.TIORH**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 0, 6) Function** | **TIOCAn Pin (n = 0, 6) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU0<br>Capture input source is TPU1 count clock<br>Input capture at TPU1.TCNT count-up/count-down*1<br>• TPU6<br>Capture input source is TPU7 count clock<br>Input capture at TPU7.TCNT count-up/count-down*1 |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 0, 6) Function** | **TIOCBn Pin (n = 0, 6) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*2; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*2; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*2; input capture at both edges |
| 1 | 1 | x | x | | • TPU0<br>Capture input source is TPU1 count clock<br>Input capture at TPU1.TCNT count-up/count-down*1<br>• TPU6<br>Capture input source is TPU7 count clock<br>Input capture at TPU7.TCNT count-up/count-down*1 |

Notes: 1. When the TPSC[2:0] bits in TPUm.TCR are set to 000b and PCLK/1 is used as the TPUm.TCNT count clock, this setting is invalid and input capture is not generated (m = 1, 7).

2. Selected by the ICSELB bit in TPUm.TMDR (m = 0, 6).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.16 TPU1.TIOR, TPU7.TIOR**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 1, 7) Function** | **TIOCAn Pin (n = 1, 7) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is lowoutput; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU1<br>Capture input source is TPU0.TGRA compare match/input capture<br>Input capture at generation of TPU0.TGRA compare match/input capture<br>• TPU7<br>Capture input source is TPU6.TGRA compare match/input capture<br>Input capture at generation of TPU6.TGRA compare match/input capture |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 1, 7) Function** | **TIOCBn Pin (n = 1, 7) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*; input capture at both edges |
| 1 | 1 | x | x | | • TPU1<br>Capture input source is TPU0.TGRC compare match/input capture<br>Input capture at generation of TPU0.TGRC compare match/input capture<br>• TPU7<br>Capture input source is TPU6.TGRC compare match/input capture<br>Input capture at generation of TPU6.TGRC compare match/input capture |

Note: * Selected by the ICSELB bit in TPUm.TMDR (m = 1, 7).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.17 TPU2.TIOR, TPU8.TIOR**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 2, 8) Function** | **TIOCAn Pin (n = 2, 8) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | x | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | x | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | x | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 2, 8) Function** | **TIOCBn Pin (n = 2, 8) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | x | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*; input capture at rising edge |
| 1 | x | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*; input capture at falling edge |
| 1 | x | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*; input capture at both edges |

Note: * Selected by the ICSELB bit in TPUm.TMDR (m = 2, 8).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.18 TPU3.TIORH, TPU9.TIORH**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 3, 9) Function** | **TIOCAn Pin (n = 3, 9) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU3<br>Capture input source is TPU4 count clock<br>Input capture at TPU4.TCNT count-up/count-down*1<br>• TPU9<br>Capture input source is TPU10 count clock<br>Input capture at TPU10.TCNT count-up/count-down*1 |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 3, 9) Function** | **TIOCBn Pin (n = 3, 9) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*2; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*2; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*2; input capture at both edges |
| 1 | 1 | x | x | | • TPU3<br>Capture input source is TPU4 count clock<br>Input capture at TPU4.TCNT count-up/count-down*1<br>• TPU9<br>Capture input source is TPU10 count clock<br>Input capture at TPU10.TCNT count-up/count-down*1 |

Notes: 1. When the TPSC[2:0] bits in TPUm.TCR are set to 000b and PCLK/1 is used as the TPUm.TCNT count clock, this setting is invalid and input capture is not generated (m = 4, 10).

2. Selected by the ICSELB bit in TPUm.TMDR (m = 3, 9).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.19 TPU4.TIOR, TPU10.TIOR**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 4, 10) Function** | **TIOCAn Pin (n = 4, 10) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU4<br>Capture input source is TPU3.TGRA compare match/input capture<br>Input capture at generation of TPU3.TGRA compare match/input capture<br>• TPU10<br>Capture input source is TPU9.TGRA compare match/input capture<br>Input capture at generation of TPU9.TGRA compare match/input capture |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 4, 10) Function** | **TIOCBn Pin (n = 4, 10) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*; input capture at both edges |
| 1 | 1 | x | x | | • TPU4<br>Capture input source is TPU3.TGRC compare match/input capture<br>Input capture at generation of TPU3.TGRC compare match/input capture<br>• TPU10<br>Capture input source is TPU9.TGRC compare match/input capture<br>Input capture at generation of TPU9.TGRC compare match/input capture |

Note: * Selected by the ICSELB bit in TPUm.TMDR (m = 4, 10).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.20 TPU5.TIOR, TPU11.TIOR**

| Bits IOA[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b3** | **b2** | **b1** | **b0** | **TPUm.TGRA (m = 5, 11) Function** | **TIOCAn Pin (n = 5, 11) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | x | 0 | 0 | Input capture register | Capture input source is TIOCAn pin; input capture at rising edge |
| 1 | x | 0 | 1 | | Capture input source is TIOCAn pin; input capture at falling edge |
| 1 | x | 1 | x | | Capture input source is TIOCAn pin; input capture at both edges |

| Bits IOB[3:0] | | | | Description | |
|---|---|---|---|---|---|
| **b7** | **b6** | **b5** | **b4** | **TPUm.TGRB (m = 5, 11) Function** | **TIOCBn Pin (n = 5, 11) Function** |
| 0 | 0 | 0 | 0 | Output compare register | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | x | 0 | 0 | Input capture register | Capture input source is TIOCBn or TIOCAn pin*; input capture at rising edge |
| 1 | x | 0 | 1 | | Capture input source is TIOCBn or TIOCAn pin*; input capture at falling edge |
| 1 | x | 1 | x | | Capture input source is TIOCBn or TIOCAn pin*; input capture at both edges |

Note: * Selected by the ICSELB bit in TPUm.TMDR (m = 5, 11).

[Legend] x: Don't care



Downloaded from Arrow.com.

**Table 15.21 TPU0.TIORL, TPU6.TIORL**

| Bits IOC[3:0] | | | | Description | |
|---|---|---|---|---|---|
| b3 | b2 | b1 | b0 | TPUm.TGRC (m = 0, 6) Function | TIOCCn Pin (n = 0, 6) Function |
| 0 | 0 | 0 | 0 | Output compare register*1 | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register*1 | Capture input source is TIOCCn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCCn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCCn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU0<br>Capture input source is TPU1 count clock<br>Input capture at TPU1.TCNT count-up/count-down*3<br>• TPU6<br>Capture input source is TPU7 count clock<br>Input capture at TPU7.TCNT count-up/count-down*3 |

| Bits IOD[3:0] | | | | Description | |
|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | TPUm.TGRD (m = 0, 6) Function | TIOCDn Pin (n = 0, 6) Function |
| 0 | 0 | 0 | 0 | Output compare register*2 | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register*2 | Capture input source is TIOCDn or TIOCCn pin*4; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCDn or TIOCCn pin*4; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCDn or TIOCCn pin*4; input capture at both edges |
| 1 | 1 | x | x | | • TPU0<br>Capture input source is TPU1 count clock<br>Input capture at TPU1.TCNT count-up/count-down*3<br>• TPU6<br>Capture input source is TPU7 count clock<br>Input capture at TPU7.TCNT count-up/count-down*3 |

Notes: 1. When the BFA bit in TPUm.TMDR is set to 1 (TPUm.TGRA and TPUm.TGRC are used for buffer operation) and TPUm.TGRC is used as a buffer register, this setting is invalid and input capture/output compare is not generated (m = 0, 6).

2. When the BFB bit in TPUm.TMDR is set to 1 (TPUm.TGRB and TPUm.TGRD are used for buffer operation) and TPUmn.TGRD is used as a buffer register, this setting is invalid and input capture/output compare is not generated (m = 0, 6).

3. When the TPSC[2:0] bits in TPUm.TCR are set to 000b and PCLK/1 is used as the TPUm.TCNT count clock, this setting is invalid and input capture is not generated (m = 1, 7).

4. Selected by the ICSELD bit in TPUm.TMDR (m = 0, 6).

[Legend] x: Don't care



Downloaded from Arrow.com.

Table 15.22 TPU3.TIORL, TPU9.TIORL

| Bits IOC[3:0] | | | | Description | |
|---|---|---|---|---|---|
| b3 | B2 | b1 | b0 | TPUm.TGRC (m = 3, 9) Function | TIOCCn Pin (n = 3, 9) Function |
| 0 | 0 | 0 | 0 | Output compare register*1 | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input capture register*1 | Capture input source is TIOCCn pin; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCCn pin; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCCn pin; input capture at both edges |
| 1 | 1 | x | x | | • TPU3<br>Capture input source is TPU4 count clock<br>Input capture at TPU4.TCNT count-up/count-down*3<br>• TPU9<br>Capture input source is TPU10 count clock<br>Input capture at TPU10.TCNT count-up/count-down*3 |

| Bits IOD[3:0] | | | | Description | |
|---|---|---|---|---|---|
| b7 | B6 | b5 | b4 | TPUm.TGRD (m = 3, 9) Function | TIOCDn Pin (n = 3, 9) Function |
| 0 | 0 | 0 | 0 | Output compare register*2 | Output disabled |
| 0 | 0 | 0 | 1 | | Initial output is low output; low output at compare match |
| 0 | 0 | 1 | 0 | | Initial output is low output; high output at compare match |
| 0 | 0 | 1 | 1 | | Initial output is low output; toggle output at compare match |
| 0 | 1 | 0 | 0 | | Output disabled |
| 0 | 1 | 0 | 1 | | Initial output is high output; low output at compare match |
| 0 | 1 | 1 | 0 | | Initial output is high output; high output at compare match |
| 0 | 1 | 1 | 1 | | Initial output is high output; toggle output at compare match |
| 1 | 0 | 0 | 0 | Input Capture register*2 | Capture input source is TIOCDn or TIOCCn pin*4; input capture at rising edge |
| 1 | 0 | 0 | 1 | | Capture input source is TIOCDn or TIOCCn pin*4; input capture at falling edge |
| 1 | 0 | 1 | x | | Capture input source is TIOCDn or TIOCCn pin*4; input capture at both edges |
| 1 | 1 | x | x | | • TPU3<br>Capture input source is TPU4 count clock<br>Input capture at TPU4.TCNT count-up/count-down*3<br>• TPU9<br>Capture input source is TPU10 count clock<br>Input capture at TPU10.TCNT count-up/count-down*3 |

Notes: 1. When the BFA bit in TPUm.TMDR is set to 1 (TPUm.TGRA and TPUm.TGRC are used for buffer operation) and TPUm.TGRC is used as a buffer register, this setting is invalid and input capture/output compare is not generated (m = 0, 6).

2. When the BFB bit in TPUm.TMDR is set to 1 (TPUm.TGRB and TPUm.TGRD are used for buffer operation) and TPUm.TGRD is used as a buffer register, this setting is invalid and input capture/output compare is not generated (m = 0, 6).

3. When the TPSC[2:0] bits in TPUm.TCR are set to 000b and PCLK/1 is used as the TPUm.TCNT count clock, this setting is invalid and input capture is not generated (m = 1, 7).

4. Selected by the ICSELD bit in TPUm.TMDR (m = 0, 6).

[Legend] x: Don't care



Downloaded from Arrow.com.

## 15.2.4 Timer Interrupt Enable Register (TIER)

Addresses: TPU0.TIER 0008 8114h, TPU1.TIER 0008 8124h, TPU2.TIER 0008 8134h
TPU3.TIER 0008 8144h, TPU4.TIER 0008 8154h, TPU5.TIER 0008 8164h
TPU6.TIER 0008 8184h, TPU7.TIER 0008 8194h, TPU8.TIER 0008 81A4h
TPU9.TIER 0008 81B4h, TPU10.TIER 0008 81C4h, TPU11.TIER 0008 81D4h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TTEG | – | TCIEU | TCIEV | TGIED | TGIEC | TGIEB | TGIEA |
| Value after reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TGIEA | TGRA Interrupt Enable | 0: Interrupt requests (TGImA) disabled<br>1: Interrupt requests (TGImA) enabled<br>(m = 0 to 11) | R/W |
| b1 | TGIEB | TGRB Interrupt Enable | 0: Interrupt requests (TGImB) disabled<br>1: Interrupt requests (TGImB) enabled<br>(m = 0 to 11) | R/W |
| b2 | TGIEC*1 | TGRC Interrupt Enable | 0: Interrupt requests (TGImC) disabled<br>1: Interrupt requests (TGImC) enabled<br>(m = 0, 3, 6, 9) | R/W |
| b3 | TGIED*1 | TGRD Interrupt Enable | 0: Interrupt requests (TGImD) disabled<br>1: Interrupt requests (TGImD) enabled<br>(m = 0, 3, 6, 9) | R/W |
| b4 | TCIEV | Overflow Interrupt Enable | 0: Interrupt requests (TCImV) disabled<br>1: Interrupt requests (TCImV) enabled<br>(m = 0 to 11) | R/W |
| b5 | TCIEU*2 | Underflow Interrupt Enable | 0: Interrupt requests (TCImU) disabled<br>1: Interrupt requests (TCImU) enabled<br>(m = 1, 2, 4, 5, 7, 8, 10, 11) | R/W |
| b6 | — | Reserved | This bit is read as 1. The write value should always be 1. | R/W |
| b7 | TTGE | A/D Conversion Start Request Enable | 0: A/D conversion start request generation disabled<br>1: A/D conversion start request generation enabled | R/W |

Notes: 1. Bits 3 and 2 in TIER of TPU1, TPU2, TPU4, TPU5 (unit 0), TPU7, TPU8, TPU10, and TPU11 (unit 1) are reserved. These bits are read as 0. The write value should always be 0.

2. Bits 5 in TIER of TPU0, TPU3 (unit 0), TPU6, and TPU9 (unit 1) is reserved. This bit is read as 0. The write value should always be 0.

The TPU has twelve TIER registers, one for each channel.

TPUm.TIER controls enabling or disabling of interrupt requests for each channel.

**TGIEA Bit (TGRA Interrupt Enable)**

Enables/disables interrupt requests (TGImA) (m = 0 to 11).



Downloaded from Arrow.com.

**TGIEB Bit (TGRB Interrupt Enable)**

Enables/disables interrupt requests (TGImB) (m = 0 to 11).

**TGIEC Bit (TGRC Interrupt Enable)**

Enables/disables interrupt requests (TGImC) (m = 0, 3, 6, 9).

**TGIED Bit (TGRD Interrupt Enable)**

Enables/disables interrupt requests (TGImD) (m = 0, 3, 6, 9).

**TCIEV Bit (Overflow Interrupt Enable)**

Enables/disables interrupt requests (TCImV) (m = 0 to 11).

**TCIEU Bit (Underflow Interrupt Enable)**

Enables/disables interrupt requests (TCImU) (m = 1, 2, 4, 5, 7, 8, 10, 11).

**TTGE Bit (A/D Conversion Start Request Enable)**

Enables/disables generation of A/D conversion start requests by TPUm.TGRA (m = 0 to 11) input capture/compare match.



Downloaded from Arrow.com.

## 15.2.5 Timer Status Register (TSR)

Addresses: TPU0.TSR 0008 8115h, TPU1.TSR 0008 8125h, TPU2.TSR 0008 8135h
TPU3.TSR 0008 8145h, TPU4.TSR 0008 8155h, TPU5.TSR 0008 8165h
TPU6.TSR 0008 8185h, TPU7.TSR 0008 8195h, TPU8.TSR 0008 81A5h
TPU9.TSR 0008 81B5h, TPU10.TSR 0008 81C5h, TPU11.TSR 0008 81D5h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TCFD | – | – | – | – | – | – | – |
| Value after reset: | 1 | 1 | x | x | x | x | x | x |

[Legend] x: Undefined

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b5 to b0 | — | Reserved | The read value is undefined. The write value should always be 0. | R/W |
| b6 | — | Reserved | This bit is read as 1 and cannot be modified. | R |
| b7 | TCFD* | Count Direction Flag | 0: TPUm.TCNT counts down<br>1: TPUm.TCNT counts up<br>(m = 1, 2, 4, 5, 7, 8, 10, 11) | R |

Note: * Bit 7 in TSR of TPU0, TPU3 (unit 0), TPU6, and TPU9 (unit 1) is reserved. This bit is read as 1. The write value should always be 1.

The TPU has twelve TSR registers, one for each channel.

TPUm.TSR indicates the count direction of the TPUm.TCNT counter.

### TCFD Flag (Count Direction Flag)

Status flag that shows that TPUm.TCNT (m = 1, 2, 4, 5, 7, 8, 10, 11) counts up or down.



Downloaded from Arrow.com.

### 15.2.6 Timer Counter (TCNT)

Addresses: TPU0.TCNT 0008 8116h, TPU1.TCNT 0008 8126h, TPU2.TCNT 0008 8136h
TPU3.TCNT 0008 8146h, TPU4.TCNT 0008 8156h, TPU5.TCNT 0008 8166h
TPU6.TCNT 0008 8186h, TPU7.TCNT 0008 8196h, TPU8.TCNT 0008 81A6h
TPU9.TCNT 0008 81B6h, TPU10.TCNT 0008 81C6h, TPU11.TCNT 0008 81D6h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The TPU has twelve TCNT counters, one for each channel.

TPUm.TCNT is a 16-bit counter that counts the internal clock or external events.

This counter can be read/written in 16-bit units.

This counter is initialized to 0000h by a reset.

### 15.2.7 Timer General Register A (TGRA)
### Timer General Register B (TGRB)
### Timer General Register C (TGRC)
### Timer General Register D (TGRD)

Addresses: TPU0.TGRA 0008 8118h, TPU0.TGRB 0008 811Ah, TPU0.TGRC 0008 811Ch, TPU0.TGRD 0008 811Eh
TPU1.TGRA 0008 8128h, TPU1.TGRB 0008 812Ah
TPU2.TGRA 0008 8138h, TPU2.TGRB 0008 813Ah
TPU3.TGRA 0008 8148h, TPU3.TGRB 0008 814Ah, TPU3.TGRC 0008 814Ch, TPU3.TGRD 0008 814Eh
TPU4.TGRA 0008 8158h, TPU4.TGRB 0008 815Ah
TPU5.TGRA 0008 8168h, TPU5.TGRB 0008 816Ah
TPU6.TGRA 0008 8188h, TPU6.TGRB 0008 818Ah, TPU6.TGRC 0008 818Ch, TPU6.TGRD 0008 818Eh
TPU7.TGRA 0008 8198h, TPU7.TGRB 0008 819Ah
TPU8.TGRA 0008 81A8h, TPU8.TGRB 0008 81AAh
TPU9.TGRA 0008 81B8h, TPU9.TGRB 0008 81BAh, TPU9.TGRC 0008 81BCh, TPU9.TGRD 0008 81BEh
TPU10.TGRA 0008 81C8h, TPU10.TGRB 0008 81CAh
TPU11.TGRA 0008 81D8h, TPU11.TGRB 0008 81DAh

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

The TPU has 32 TGR registers in total, four each for TPU0, TPU3, TPU6, and TPU9 and two each for TPU1, TPU2, TPU4, TPU5, TPU7, TPU8, TPU10, and TPU11.

TPUm.TGRA (m = 0 to 11), TPUm.TGRB (m = 0 to 11), TPUm.TGRC (m = 0, 3, 6, 9), and TPUm.TGRD (m = 0, 3, 6, 9) are 16-bit registers with a dual function as output compare and input capture registers.

These registers can be read/written in 16-bit units.

TPUm.TGRC and TPUm.TGRD can also be specified for operation as buffer registers. Register combinations during buffer operations are TPUm.TGRA–TPUm.TGRC and TPUm.TGRB–TPUm.TGRD.



Downloaded from Arrow.com.

## 15.2.8 Timer Start Register (TSTRA, TSTRB)

Addresses: TSTRA 0008 8100h, TSTRB 0008 8170h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | CST5 | CST4 | CST3 | CST2 | CST1 | CST0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | CST0 | Counter Start 0 | 0: TCNT count operation is stopped | R/W |
| b1 | CST1 | Counter Start 1 | 1: TCNT performs count operation | R/W |
| b2 | CST2 | Counter Start 2 | | R/W |
| b3 | CST3 | Counter Start 3 | | R/W |
| b4 | CST4 | Counter Start 4 | | R/W |
| b5 | CST5 | Counter Start 5 | | R/W |
| b7, b6 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

TSTRA starts or stops TCNT count operation for TPU0 to TPU5.

TSTRB starts or stops TCNT count operation for TPU6 to TPU11.

Before setting the operating mode in TPUm.TMDR or setting the TPUm.TCNT count clock in TPUm.TCR, stop the TPUm.TCNT counter.

**CSTj Bits (Counter Start) (j = 0 to 5)**

These bits start or stop the TCNT counter.

When the CSTj bit is cleared to 0 with CSTj = 1 and the corresponding TIOCyn pin (y = A to D, n = 0 to 11) specified for output, the counter stops but the output compare output level of the corresponding TIOCyn pin is retained.

If TIORH, TIORL, or TIOR is written to when the CSTj bit is 0, the pin output level will be changed to the set initial output value.



Downloaded from Arrow.com.

## 15.2.9 Timer Synchronous Register (TSYRA, TSYRB)

Addresses: TSYRA 0008 8101h, TSYRB 0008 8171h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | SYNC5 | SYNC4 | SYNC3 | SYNC2 | SYNC1 | SYNC0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SYNC0 | Timer Synchronization 0 | 0: TCNT operates independently (TCNT presetting/clearing is unrelated to other channels)<br>1: TCNT performs synchronous operation* (TCNT synchronous presetting/synchronous clearing is possible) | R/W |
| b1 | SYNC1 | Timer Synchronization 1 | | R/W |
| b2 | SYNC2 | Timer Synchronization 2 | | R/W |
| b3 | SYNC3 | Timer Synchronization 3 | | R/W |
| b4 | SYNC4 | Timer Synchronization 4 | | R/W |
| b5 | SYNC5 | Timer Synchronization 5 | | R/W |
| b7, b6 | — | Reserved | These bits are read as 0. The write value should always be 0. | R/W |

Note: * To set synchronous operation, the SYNCj bit (j = 0 to 5) for at least two channels must be set to 1. To set synchronous clearing, the TCNT clearing source must also be set by the CCLR[2:0] bits in TCR in addition to the SYNCj bit.

TSYRA selects independent operation or synchronous operation for the TCNT counters of TPU0 to TPU5.

TSYRB selects independent operation or synchronous operation for the TCNT counters of TPU6 to TPU11.

**SYNCj Bits (Timer Synchronization) (j = 0 to 5)**

These bits select whether the TCNT operation is independent of or synchronized with TCNT of other channels.

When synchronous operation is selected, synchronous presetting of multiple TCNT counters and synchronous clearing through counter clearing on another channel are possible.



Downloaded from Arrow.com.

## 15.3 Operation

### 15.3.1 Basic Functions

Each channel has a TPUm.TCNT counter and a TPUm.TGRy register (y = A to D).

TCNT is a 16-bit up-counter, and is also capable of free-running operation, periodic counting, and external event counting.

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

#### (1) Counter Operation

When the CSTj bit (j = 0 to 5) in TSTRA or the CSTj bit (j = 0 to 5) in TSTRB is set to 1, the TCNT counter for the corresponding channel starts counting. TCNT can operate as a free-running counter, periodic counter, and so on.

#### (a) Example of count operation setting procedure

Figure 15.3 shows an example of the count operation setting procedure.

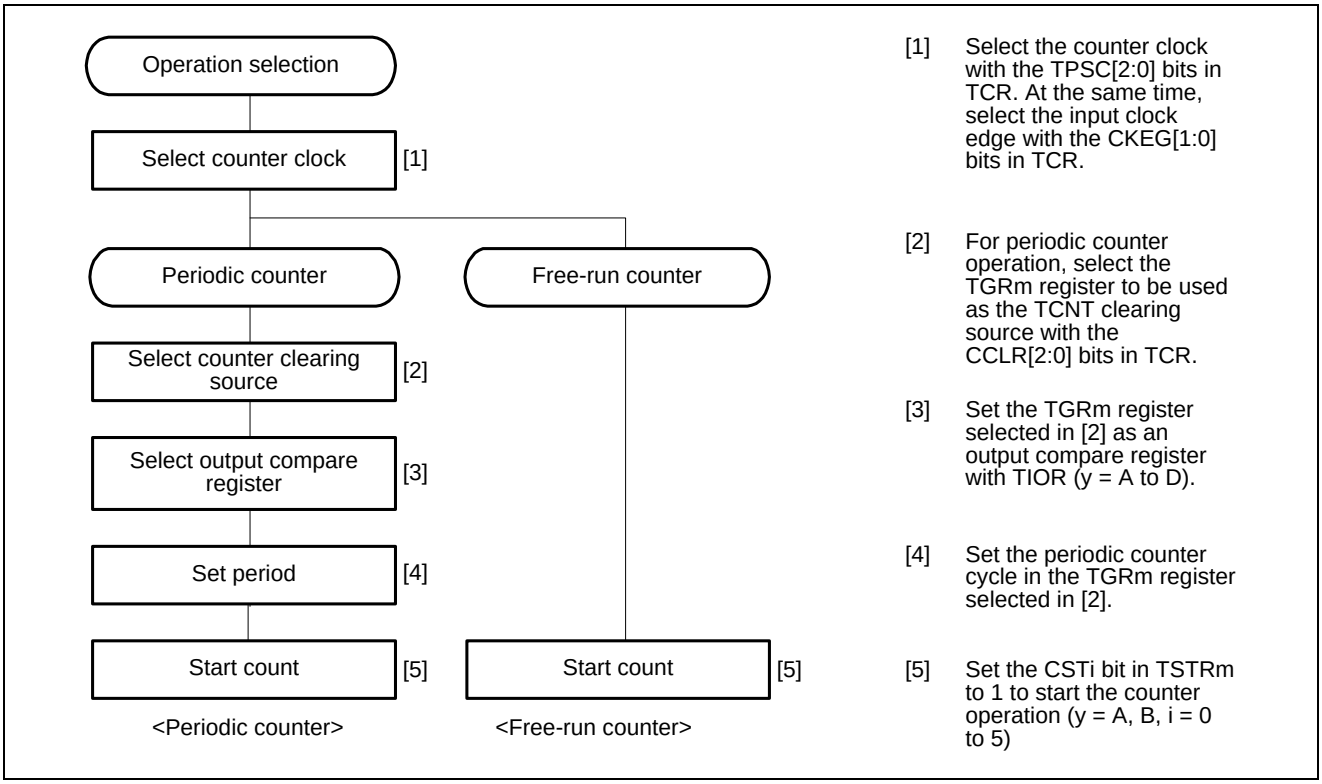



**Figure 15.3 Example of Counter Operation Setting Procedure**



Downloaded from Arrow.com.

### (b) Free-running count operation and periodic count operation

Immediately after a reset, the TPUm.TCNT counters are all set as free-running counters. When the relevant bit in TSTRA or TSTRB is set to 1, the corresponding TCNT counter starts up-count operation as a free-running counter. When TCNT overflows (changes from FFFFh to 0000h), the TPU requests an interrupt. After an overflow, TCNT restarts counting up from 0000h.

Figure 15.4 shows free-running counter operation.

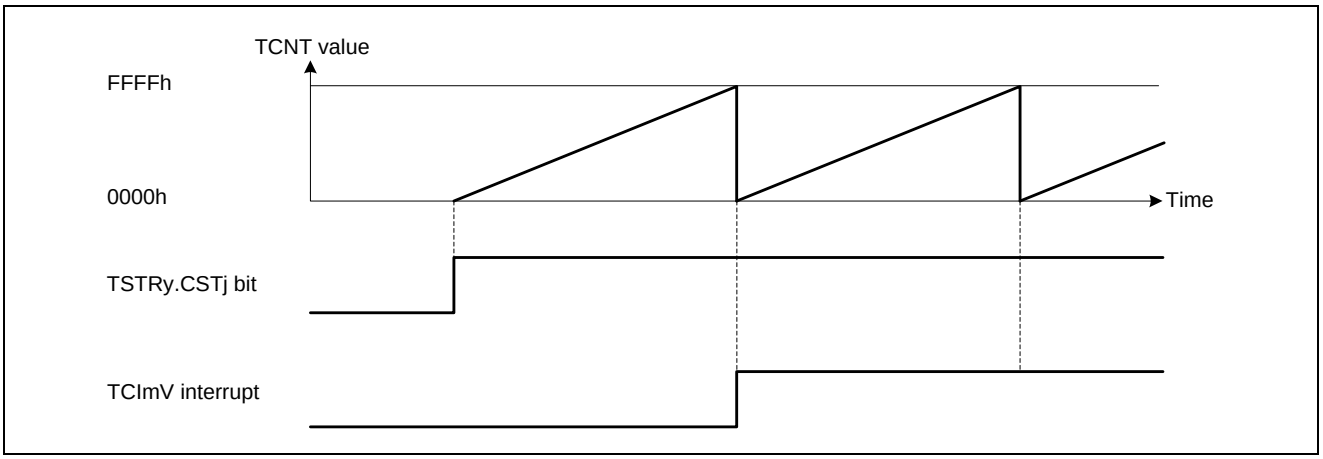



**Figure 15.4 Free-Running Counter Operation**

When compare match is selected as the TCNT clearing source, the TCNT counter for the relevant channel performs periodic count operation. The TPUm.TGRy for setting the period is set as an output compare register, and counter clearing by compare match is selected by the CCLR[2:0] bits in TPUm.TCR. After the settings have been made, TCNT starts count-up operation as a periodic counter when the corresponding bit in TSTRA or TSTRB is set to 1. When the count value matches the TGRy value, TCNT is cleared to 0000h.

At this time, the TPU requests an interrupt. After a compare match, TCNT restarts counting up from 0000h.

Figure 15.5 shows periodic counter operation.

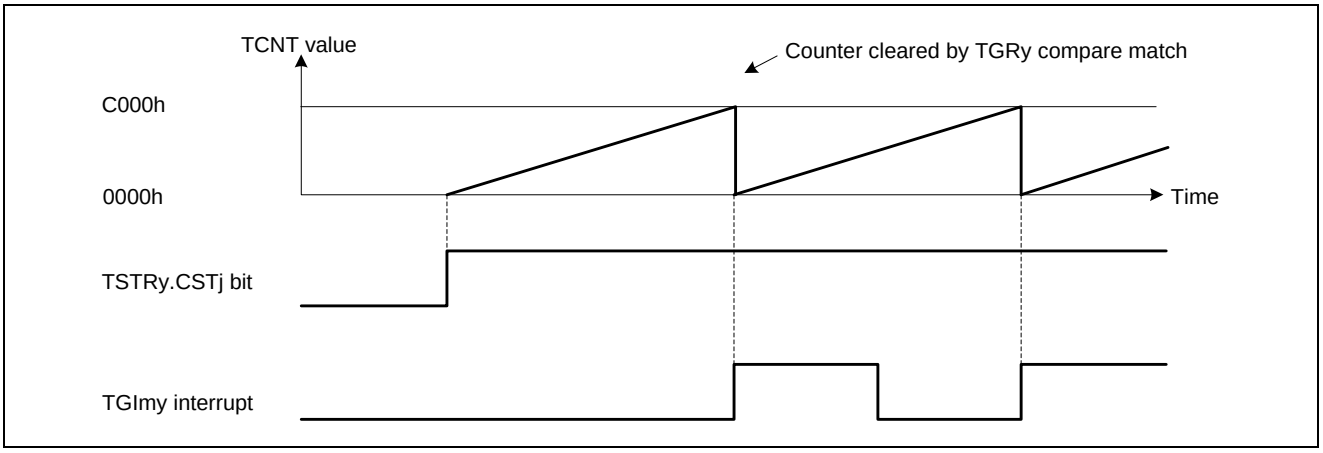



**Figure 15.5 Periodic Counter Operation**



Downloaded from Arrow.com.

### (2) Waveform Output by Compare Match

The TPU can perform low, high, or toggle output from the corresponding output pin using a compare match.

#### (a) Example of setting procedure for waveform output by compare match

Figure 15.6 shows an example of the setting procedure for waveform output by a compare match.

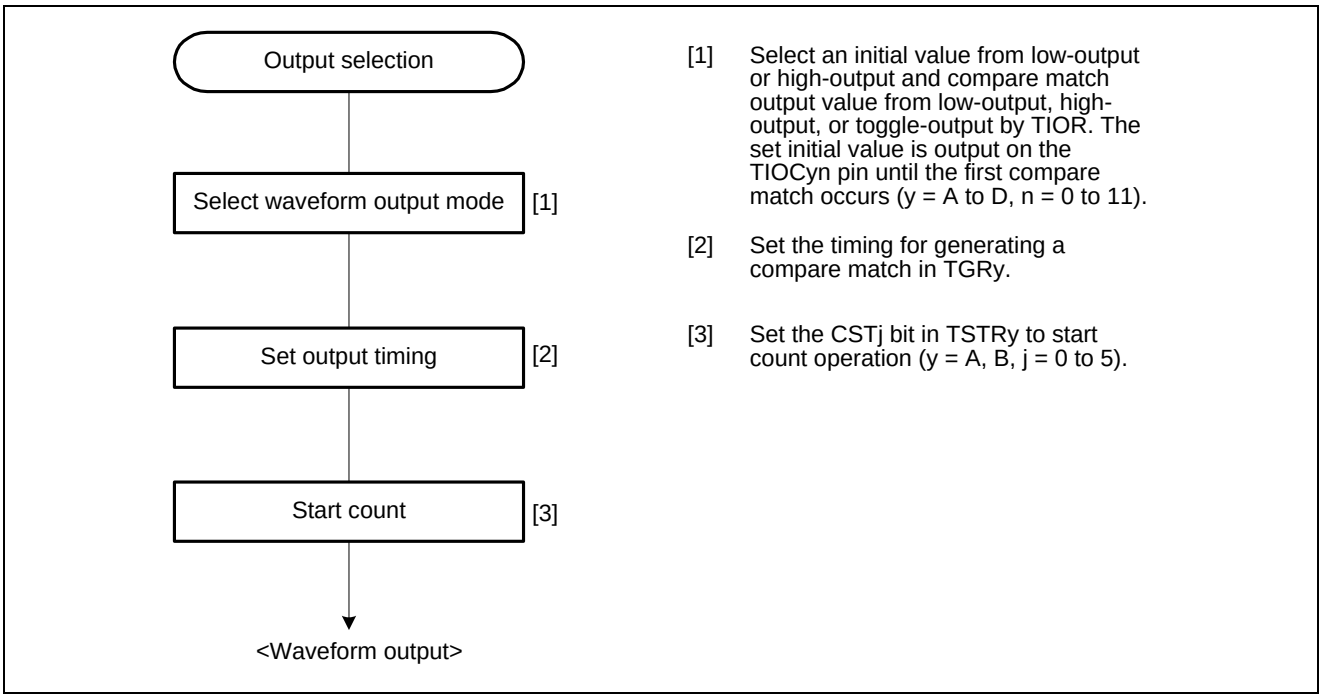



**Figure 15.6 Example of Setting Procedure for Waveform Output by Compare Match**

#### (b) Examples of waveform output operation

Figure 15.7 shows an example of low output/high output.

In this example, TPUm.TCNT has been set as a free-running counter, and settings have been made so that high is output by compare match A and low is output by compare match B. When the set level and the pin level match, the pin level does not change.

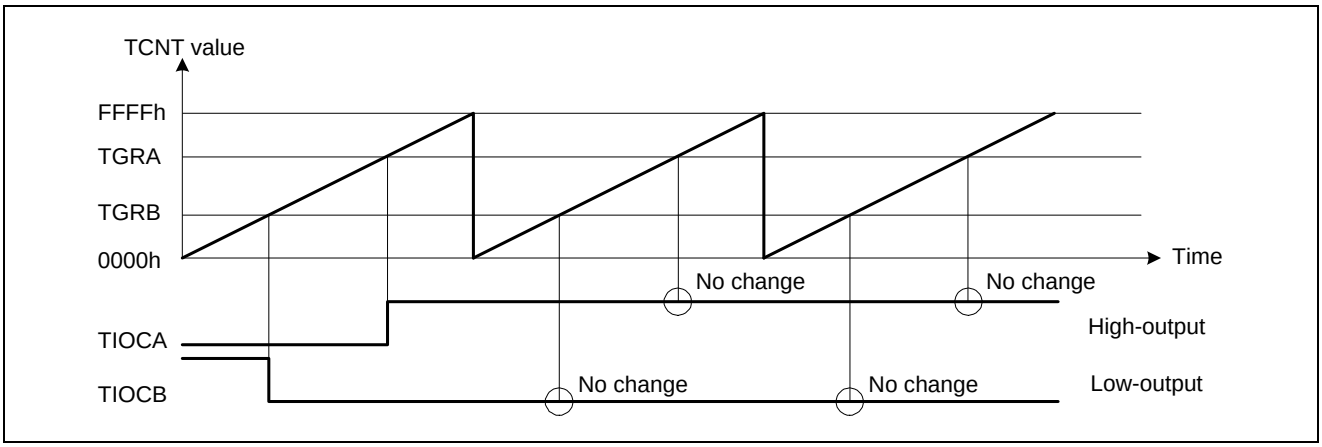



**Figure 15.7 Example of Low-Output/High-Output Operation**



Downloaded from Arrow.com.

Figure 15.8 shows an example of toggle output.

In this example, TPUm.TCNT has been set 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.

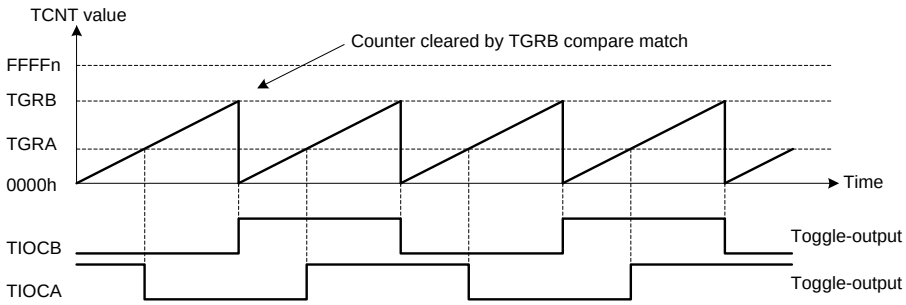



**Figure 15.8 Example of Toggle Output Operation**

### (3) Input Capture Function

The TPUm.TCNT value can be transferred to TPUm.TGRy on detection of the TIOCyn pin (y = A to D, n = 0 to 11) input edge.

The rising edge, the falling edge, or both edges can be selected as the detection edge. It is also possible to specify the counter input clock or compare match signal of TPU0, TPU1, TPU3, and TPU4 (TPU6, TPU7, TPU9, and TPU10) as the input capture source.

Note: When another channel's counter input clock is used as the input capture input for TPU0 and TPU3 (TPU6 and TPU9), PCLK/1 should not be selected as the counter input clock used for input capture input. Input capture will not be generated if PCLK/1 is selected.

#### (a) Example of setting procedure for input capture operation

Figure 15.9 shows an example of the setting procedure for input capture operation.

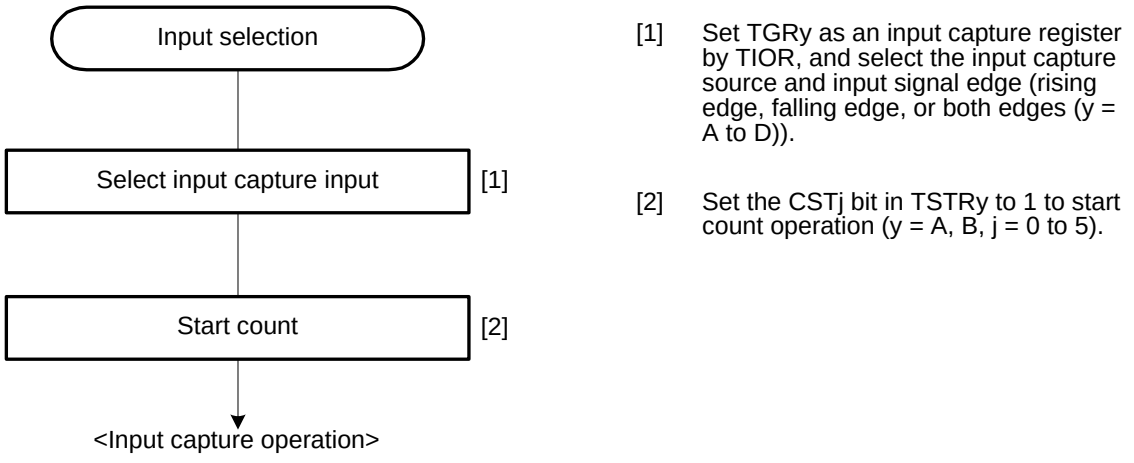



**Figure 15.9 Example of Setting Procedure for Input Capture Operation**



Downloaded from Arrow.com.

(b) Example of input capture operation

Figure 15.10 shows an example of input capture operation.

In this example, both rising and falling edges have been selected as the TIOCAn pin input capture input edge, the falling edge has been selected as the TIOCBn pin input capture input edge, and counter clearing by TPUm.TGRB input capture has been set for TPUm.TCNT.

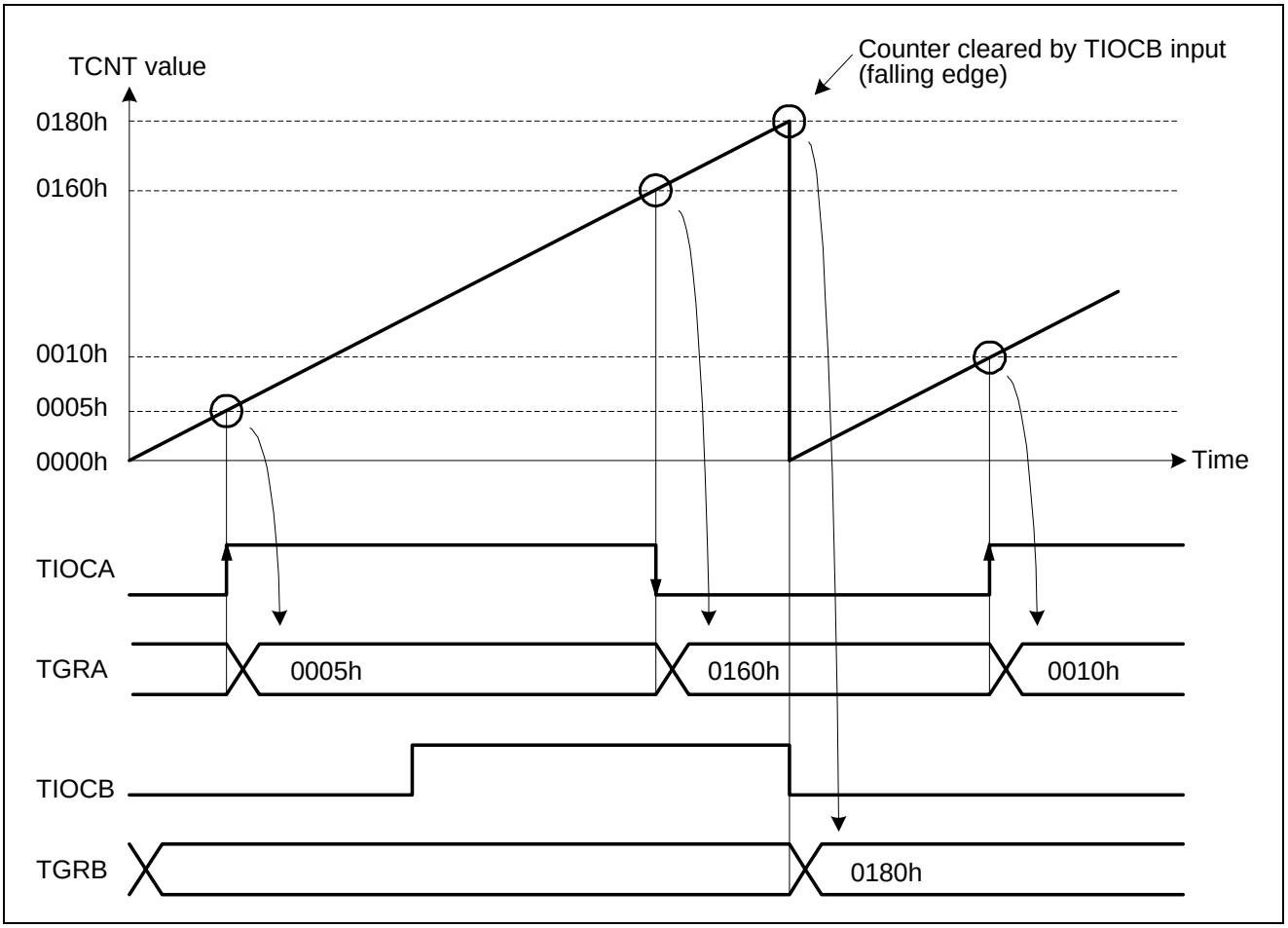



**Figure 15.10 Example of Input Capture Operation**



Downloaded from Arrow.com.

## 15.3.2 Synchronous Operation

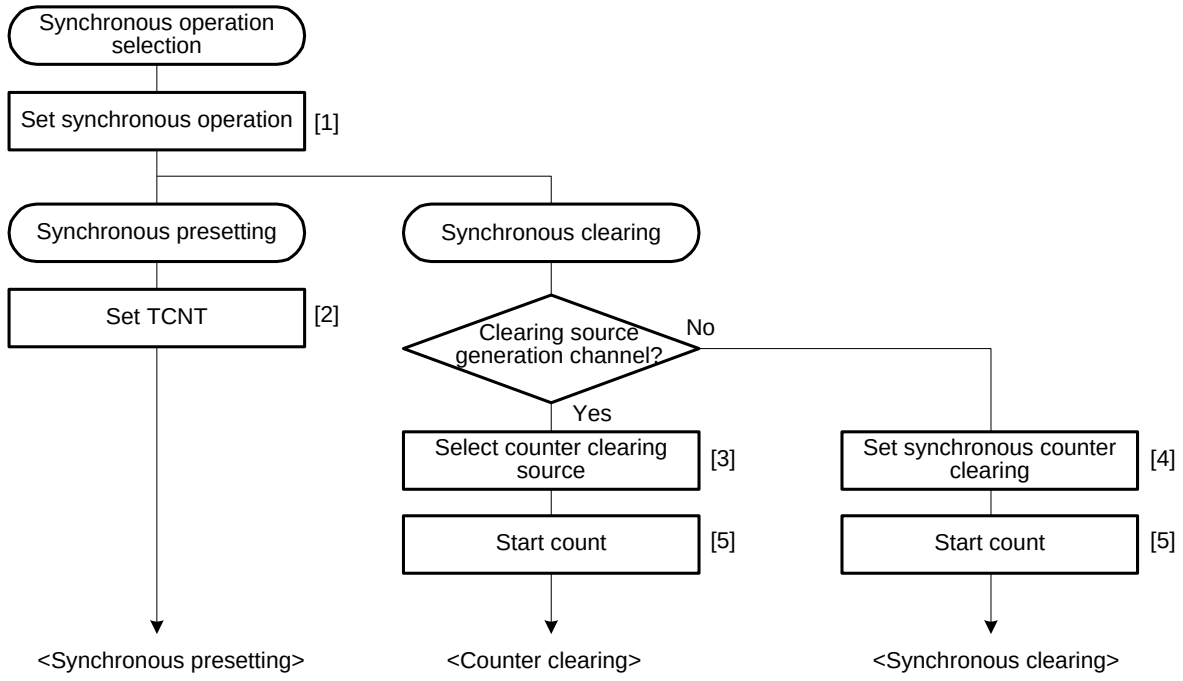
In synchronous operation, the values in multiple TPUm.TCNT counters can be rewritten simultaneously (synchronous presetting). Also, multiple TCNT counters can be cleared simultaneously (synchronous clearing) by making the appropriate setting in TPUm.TCR.

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

TPU0 to TPU5 (TPU6 to TPU11) can all be set for synchronous operation.

### (1) Example of Synchronous Operation Setting Procedure

Figure 15.11 shows an example of the synchronous operation setting procedure.




[1] Set the SYNCj bit in TSYRy corresponding to the channels to be set for synchronous operation to 1 (y = A, B, j = 0 to 5).

[2] When any of TCNT of the channels set for synchronous operation is written to, the same value is simultaneously written to the other TCNT counters.

[3] Use the CCLR[2:0] bits in TCR to specify TCNT clearing by input capture/output compare, etc.

[4] Use the CCLR[2:0] bits in TCR to specify synchronous clearing for the counter clearing source.

[5] Set the CSTj bit in TSTRy for the relevant channels to 1 to start count operation (y = A, B, j = 0 to 5).

**Figure 15.11 Example of Synchronous Operation Setting Procedure**



Downloaded from Arrow.com.

### (2) Example of Synchronous Operation

Figure 15.12 shows an example of synchronous operation.

In this example, synchronous operation and PWM mode 1 have been set for TPU0 to TPU2, TPU0.TGRB compare match has been set as the TPU0 counter clearing source, and synchronous clearing has been set for the TPU1 and TPU2 counter clearing source.

Three-phase PWM waveforms are output from pins TIOCA0, TIOCA1, and TIOCA2. At this time, synchronous presetting and synchronous clearing by TPU0.TGRB compare match are performed for TPUm.TCNT of TPU0 to TPU2, and the data set in TPU0.TGRB is used as the PWM cycle.

For details on PWM modes, see section 15.3.5, PWM Modes.

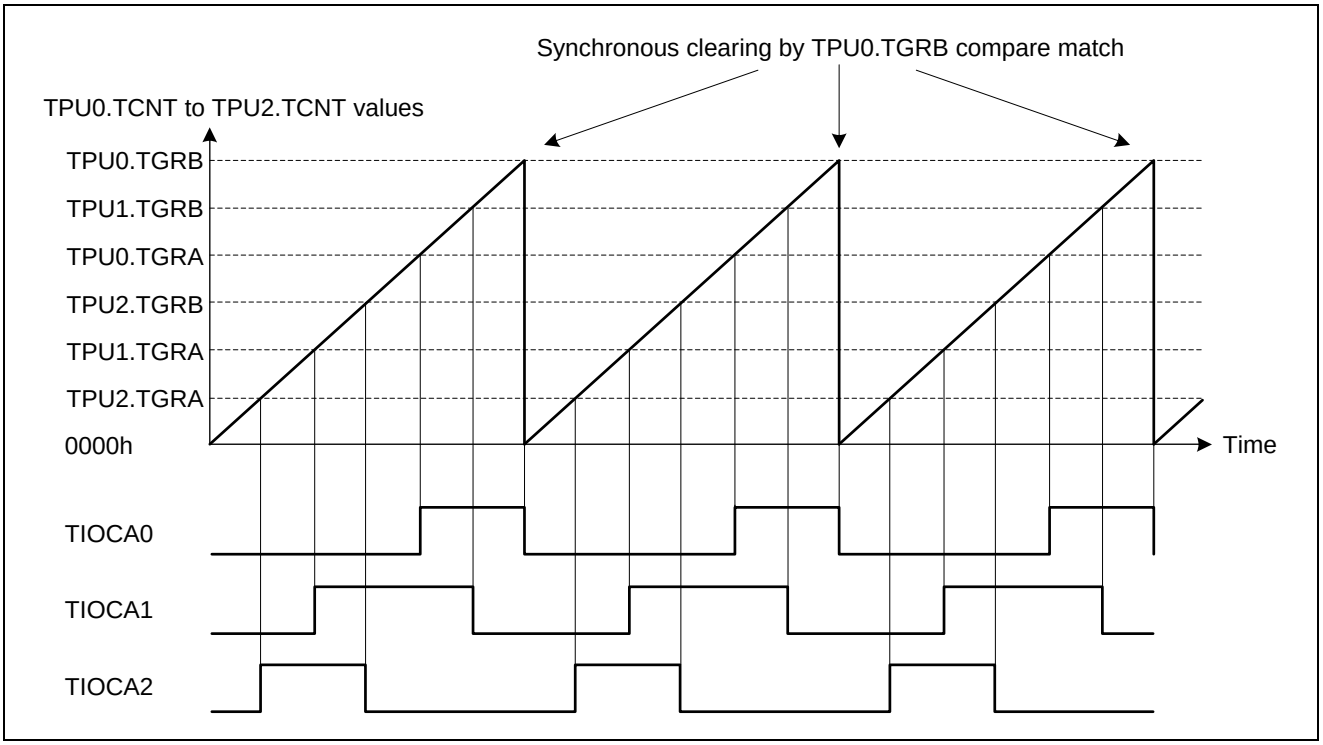



**Figure 15.12 Example of Synchronous Operation**



Downloaded from Arrow.com.

### 15.3.3 Buffer Operation

Buffer operation, provided for TPU0 and TPU3 (TPU6 and TPU9), enables TPUm.TGRC and TPUm.TGRD to be used as buffer registers.

Buffer operation differs depending on whether TPUm.TGRy has been set as an input capture register or a compare match register.

Table 15.23 lists the register combinations used in buffer operation.

**Table 15.23 Register Combinations in Buffer Operation**

| Unit | Channel | Timer General Register | Buffer Register |
|---|---|---|---|
| 0 | TPU0 | TPU0.TGRA | TPU0.TGRC |
| | | TPU0.TGRB | TPU0.TGRD |
| | TPU3 | TPU3.TGRA | TPU3.TGRC |
| | | TPU3.TGRB | TPU3.TGRD |
| 1 | TPU6 | TPU6.TGRA | TPU6.TGRC |
| | | TPU6.TGRB | TPU6.TGRD |
| | TPU9 | TPU9.TGRA | TPU9.TGRC |
| | | TPU9.TGRB | TPU9.TGRD |

- When TPUm.TGRy 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 shown in figure 15.13.

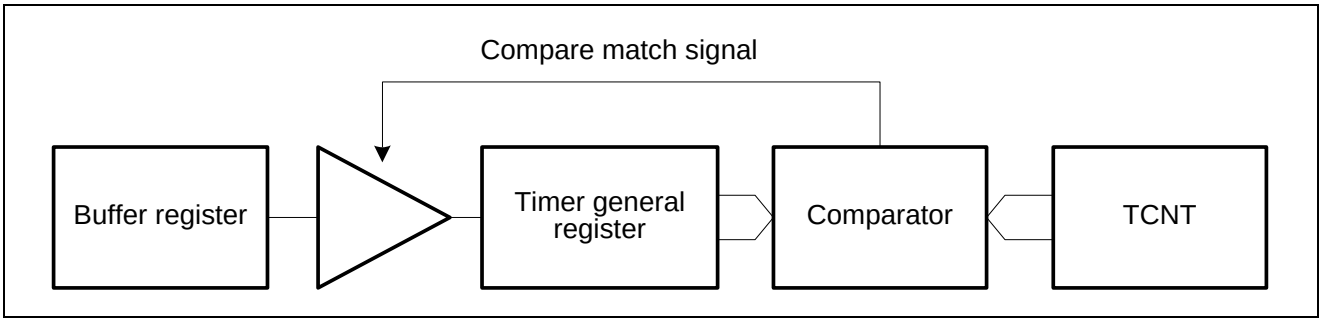



**Figure 15.13 Compare Match Buffer Operation**



Downloaded from Arrow.com.

- When TPUm.TGRy is an input capture register
  When input capture occurs, the value in TPUm.TCNT is transferred to TGRy and the value previously held in TGRy is transferred to the buffer register.

  This operation is shown in figure 15.14.

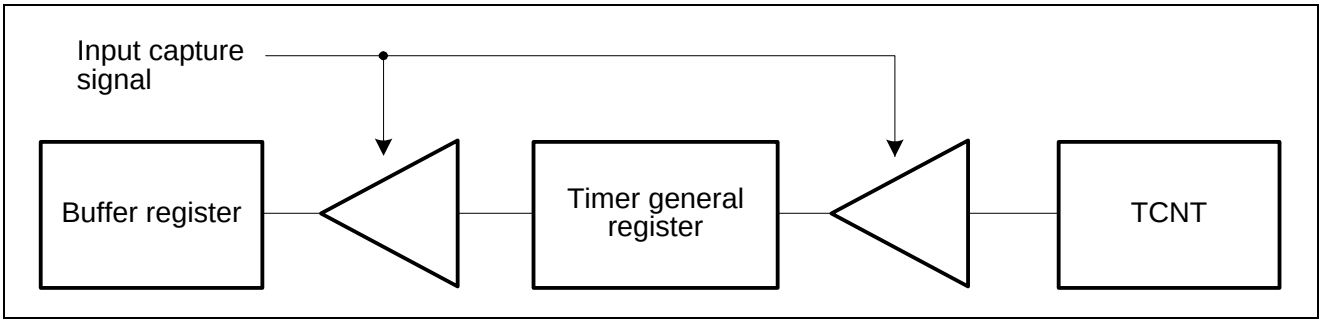



**Figure 15.14 Input Capture Buffer Operation**

### (1) Example of Buffer Operation Setting Procedure

Figure 15.15 shows an example of the buffer operation setting procedure.

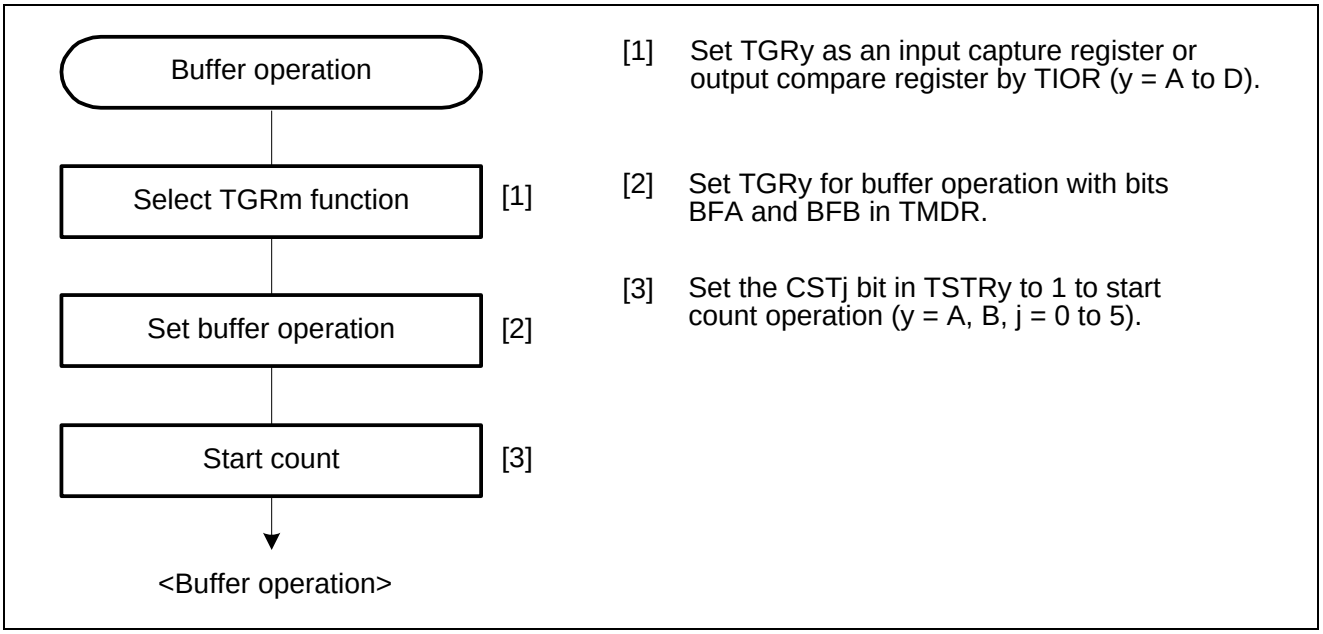



**Figure 15.15 Example of Buffer Operation Setting Procedure**



Downloaded from Arrow.com.

### (2) Examples of Buffer Operation

#### (a) When TPUm.TGRy is an output compare register

Figure 15.16 shows an operation example in which PWM mode 1 has been set for TPU0, and buffer operation has been set for TPU0.TGRA and TPU0.TGRC. The settings used in this example are TPU0.TCNT clearing by compare match B, high output at compare match A, and low output at compare match B.

As buffer operation has been set, when compare match A occurs, the output changes and the TPU0.TGRC value is simultaneously transferred to TPU0.TGRA. This operation is repeated each time compare match A occurs.

For details on PWM modes, see section 15.3.5, PWM Modes.

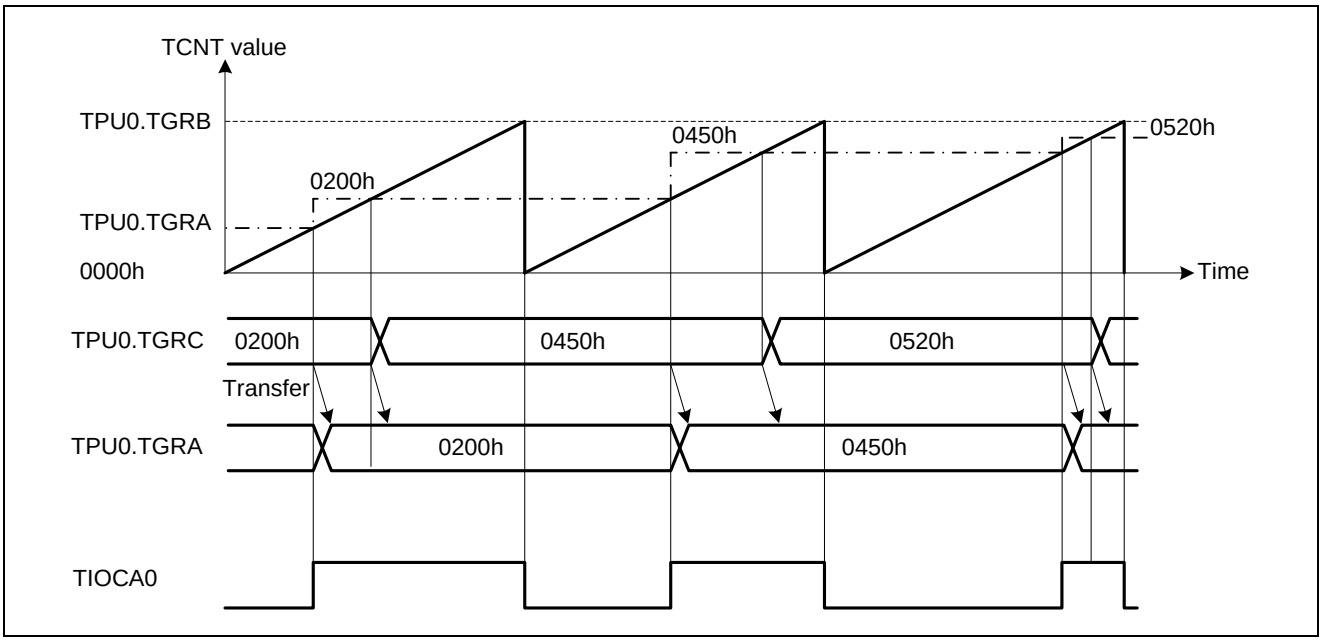



**Figure 15.16 Example of Buffer Operation (1)**



Downloaded from Arrow.com.

### (b) When TPUm.TGRy is an input capture register

Figure 15.17 shows an operation example in which TPUm.TGRA has been set as an input capture register, and buffer operation has been set for the TGRA register and TPUm.TGRC.

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

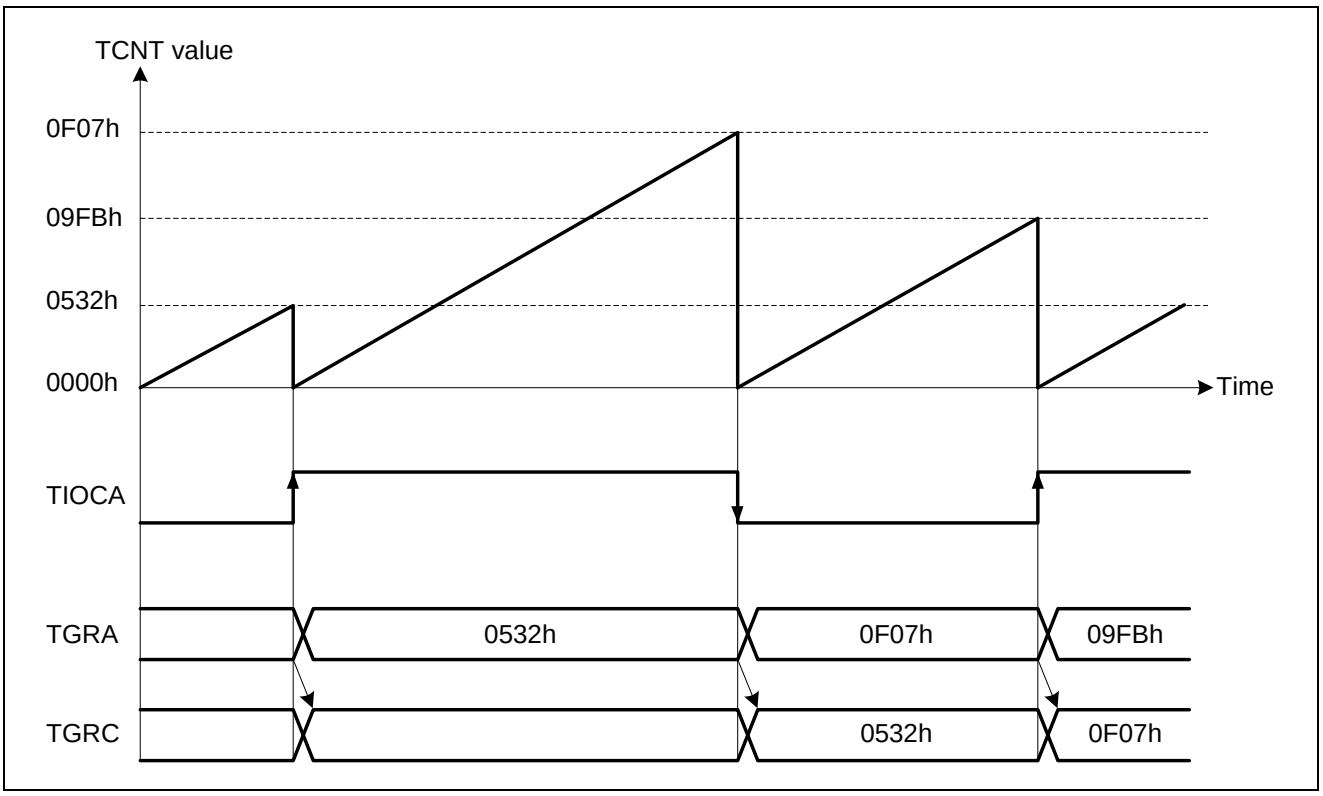



**Figure 15.17 Example of Buffer Operation (2)**



Downloaded from Arrow.com.

### 15.3.4 Cascaded Operation

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

In the case of unit 0, this function works by counting the TPU1 (TPU4) counter clock at overflow/underflow of TPU2.TCNT (TPU5.TCNT) as set by the TPSC[2:0] bits in TPU1.TCR (TPSC[2:0] bits in TPU4.TCR).

In the case of unit 1, this function works by counting the TPU7 (TPU10) counter clock at overflow/underflow of TPU8.TCNT (TPU11.TCNT) as set by the TPSC[2:0] bits in TPU7.TCR (TPSC[2:0] bits in TPU10.TCR).

Underflow occurs only when the lower 16-bit TPUm.TCNT is in phase counting mode.

Table 15.24 lists the register combinations used in cascaded operation.

Note: When phase counting mode is set for TPU1 or TPU4 (TPU7 or TPU10), the counter clock setting is invalid and the counter operates independently in phase counting mode.

**Table 15.24 Cascaded Combinations**

| Unit | Combination | Upper 16 Bits | Lower 16 Bits |
|---|---|---|---|
| 0 | TPU1 and TPU 2 | TPU1.TCNT | TPU2.TCNT |
| | TPU 4 and TPU 5 | TPU4.TCNT | TPU5.TCNT |
| 1 | TPU 7 and TPU 8 | TPU7.TCNT | TPU8.TCNT |
| | TPU 10 and TPU 11 | TPU10.TCNT | TPU11.TCNT |

#### (1) Example of Cascaded Operation Setting Procedure

Figure 15.18 shows an example of the setting procedure for cascaded operation.

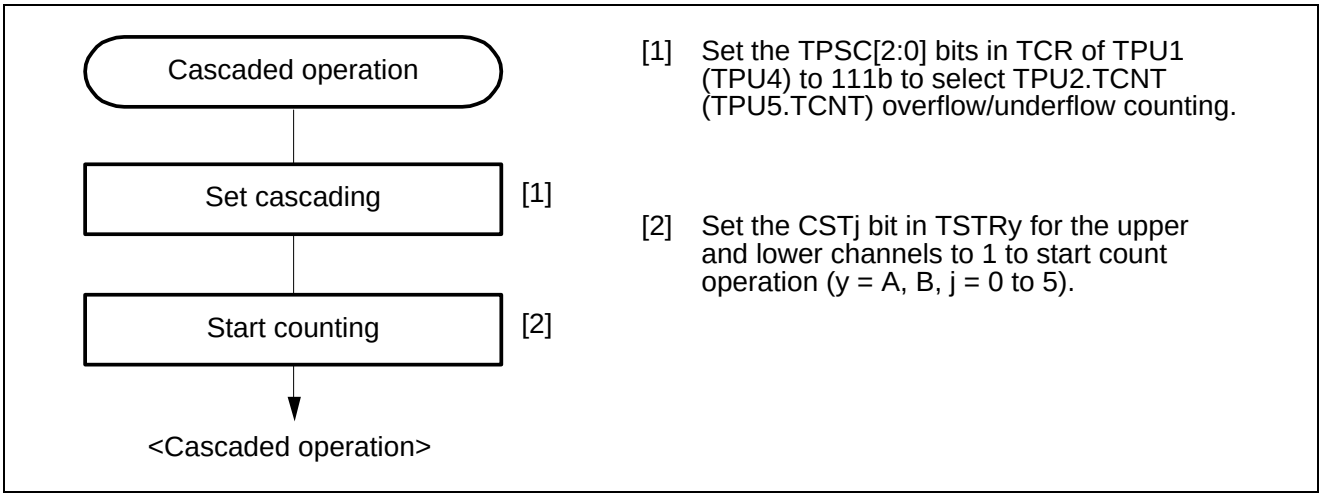



**Figure 15.18 Cascaded Operation Setting Procedure**



Downloaded from Arrow.com.

### (2) Examples of Cascaded Operation

Figure 15.19 shows the operation when counting upon TPU2.TCNT overflow/underflow has been set for TPU1.TCNT, TPU1.TGRA and TPU2.TGRA have been set as input capture registers, and the rising edge of the TIOCA1 and TIOCA2 pins has been selected.

When a rising edge is input to the TIOCA1 and TIOCA2 pins simultaneously, the upper 16 bits of the 32-bit data are transferred to TPU1.TGRA, and the lower 16 bits to TPU2.TGRA.

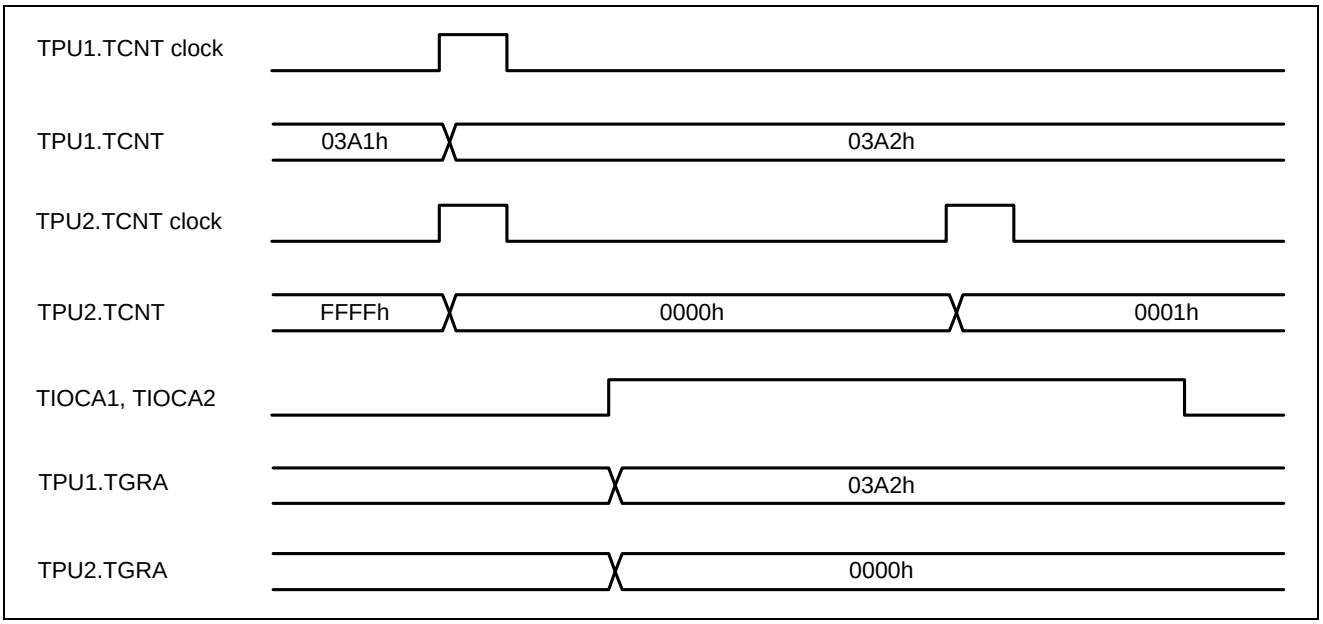



**Figure 15.19 Example of Cascaded Operation (1)**

Figure 15.20 shows the operation when counting upon TPU2.TCNT overflow/underflow has been set for TPU1.TCNT, and phase counting mode has been specified for TPU2.

TPU1.TCNT is incremented by TPU2.TCNT overflow and decremented by TPU2.TCNT underflow.

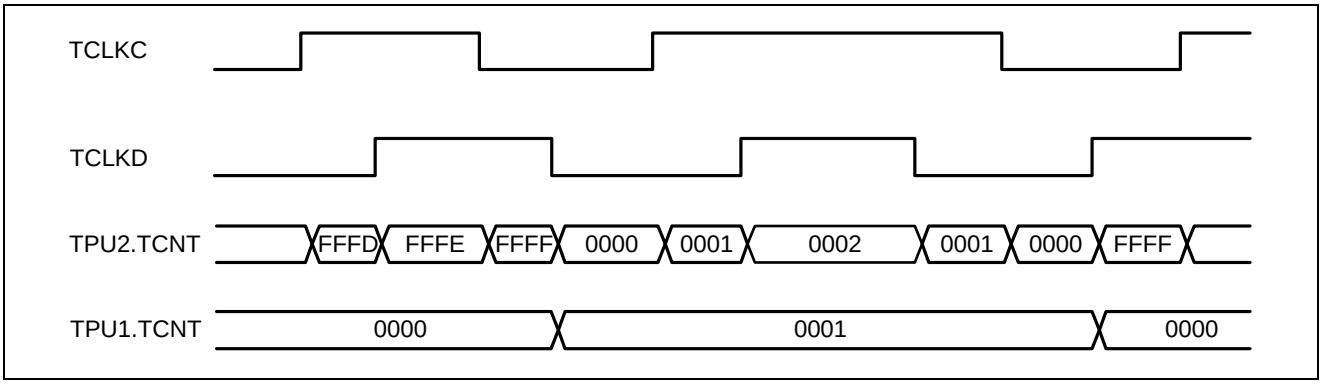



**Figure 15.20 Example of Cascaded Operation (2)**



Downloaded from Arrow.com.

### 15.3.5 PWM Modes

In PWM mode, PWM waveforms are output from the output pins. low-, high-, or toggle-output can be selected as the output level in response to compare match of each TPUm.TGRy.

Settings of TGRy registers can output a PWM waveform in the range of 0% to 100% duty cycle.

Specifying TGRy compare match as the counter clearing source enables the cycle to be set in that register. All channels can be set for PWM mode independently. Synchronous operation is also possible.

There are two PWM modes, as described below.

1. PWM mode 1
   PWM waveform is generated from the TIOCAn and TIOCCn pins by pairing TPUm.TGRA with TPUm.TGRB and TPUm.TGRC with TPUm.TGRD. The outputs specified by the IOA[3:0] bits in TPUm.TIOR(H) and IOC[3:0] bits in TPUm.TIORL are output from the TIOCAn and TIOCCn pins at compare matches A and C, respectively. The outputs specified by the IOB[3:0] bits in TPUm.TIOR(H) and IOD[3:0] bits in TPUm.TIORL are output from the TIOCAn and TIOCCn pins at compare matches B and D, respectively. The initial output value is the value set in TGRA or TGRC. If the set values of paired TGRy registers are identical, the output value does not change even when a compare match occurs.
   In PWM mode 1, a maximum 8-phase PWM output is possible.
2. PWM mode 2
   PWM waveform is generated by using one TPUm.TGRy as the cycle register and the others as duty cycle registers. The output specified in TPUm.TIORH, TPUm.TIORL, or TPUm.TIOR is performed by compare matches. Upon counter clearing by a synchronous register compare match, the output value of each pin is the initial value set in TIORH, TIORL, or TIOR. If the set values of the cycle register and duty cycle register are identical, the output value does not change even when a compare match occurs.
   In PWM mode 2, a maximum 15-phase PWM waveform is possible by combined use with synchronous operation.



Downloaded from Arrow.com.

The correspondence between PWM output pins and registers is listed in table 15.25.

**Table 15.25 PWM Output Registers and Output Pins**

| | | | Output Pin | |
|---|---|---|---|---|
| **Unit** | **Channel** | **Register** | **PWM Mode 1** | **PWM Mode 2** |
| 0 | TPU 0 | TPU0.TGRA | TIOCA0 | TIOCA0 |
| | | TPU0.TGRB | | TIOCB0 |
| | | TPU0.TGRC | TIOCC0 | TIOCC0 |
| | | TPU0.TGRD | | TIOCD0 |
| | TPU 1 | TPU1.TGRA | TIOCA1 | TIOCA1 |
| | | TPU1.TGRB | | TIOCB1 |
| | TPU 2 | TPU2.TGRA | TIOCA2 | TIOCA2 |
| | | TPU2.TGRB | | TIOCB2 |
| | TPU 3 | TPU3.TGRA | TIOCA3 | TIOCA3 |
| | | TPU3.TGRB | | TIOCB3 |
| | | TPU3.TGRC | TIOCC3 | TIOCC3 |
| | | TPU3.TGRD | | TIOCD3 |
| | TPU 4 | TPU4.TGRA | TIOCA4 | TIOCA4 |
| | | TPU4.TGRB | | TIOCB4 |
| | TPU 5 | TPU5.TGRA | TIOCA5 | TIOCA5 |
| | | TPU5.TGRB | | TIOCB5 |
| 1 | TPU 6 | TPU6.TGRA | TIOCA6 | TIOCA6 |
| | | TPU6.TGRB | | TIOCB6 |
| | | TPU6.TGRC | TIOCC6 | TIOCC6 |
| | | TPU6.TGRD | | TIOCD6 |
| | TPU 7 | TPU7.TGRA | TIOCA7 | TIOCA7 |
| | | TPU7.TGRB | | TIOCB7 |
| | TPU 8 | TPU8.TGRA | TIOCA8 | TIOCA8 |
| | | TPU8.TGRB | | TIOCB8 |
| | TPU 9 | TPU9.TGRA | TIOCA9 | TIOCA9 |
| | | TPU9.TGRB | | TIOCB9 |
| | | TPU9.TGRC | TIOCC9 | TIOCC9 |
| | | TPU9.TGRD | | TIOCD9 |
| | TPU 10 | TPU10.TGRA | TIOCA10 | TIOCA10 |
| | | TPU10.TGRB | | TIOCB10 |
| | TPU 11 | TPU11.TGRA | TIOCA11 | TIOCA11 |
| | | TPU11.TGRB | | TIOCB11 |

Note: In PWM mode 2, PWM waveform is not possible for the TPUm.TGRy register in which the cycle is set.



Downloaded from Arrow.com.

### (1) Example of PWM Mode Setting Procedure

Figure 15.21 shows an example of the PWM mode setting procedure.

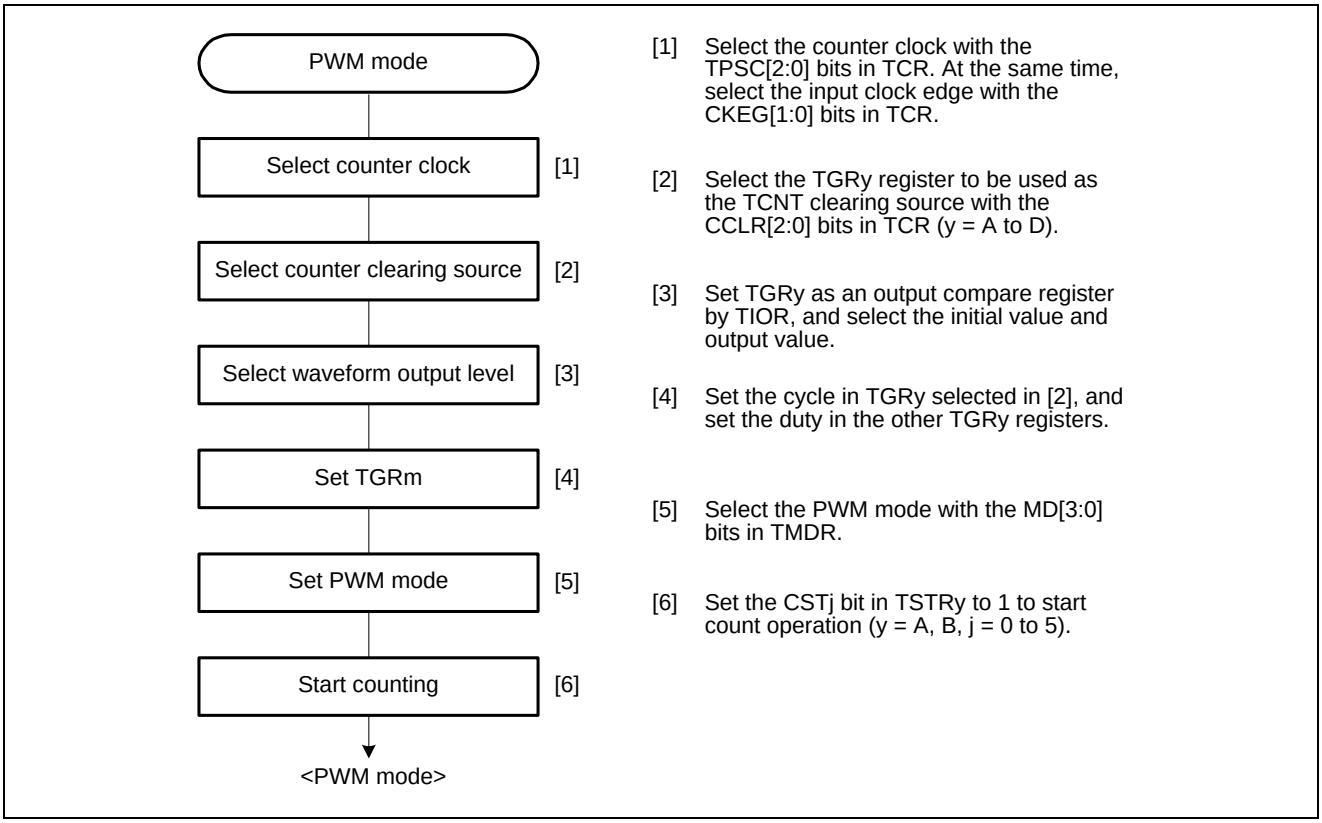



**Figure 15.21 Example of PWM Mode Setting Procedure**



Downloaded from Arrow.com.

### (2) Examples of PWM Mode Operation

Figure 15.22 shows an example of PWM mode 1 operation.

In this example, TPUm.TGRA compare match is set as the TPUm.TCNT clearing source, 0 is set for the TGRA initial output value and output value, and 1 is set as the TPUm.TGRB output value.

In this case, the value set in TGRA is used as the cycle, and the value set in TGRB is used as the duty cycle.

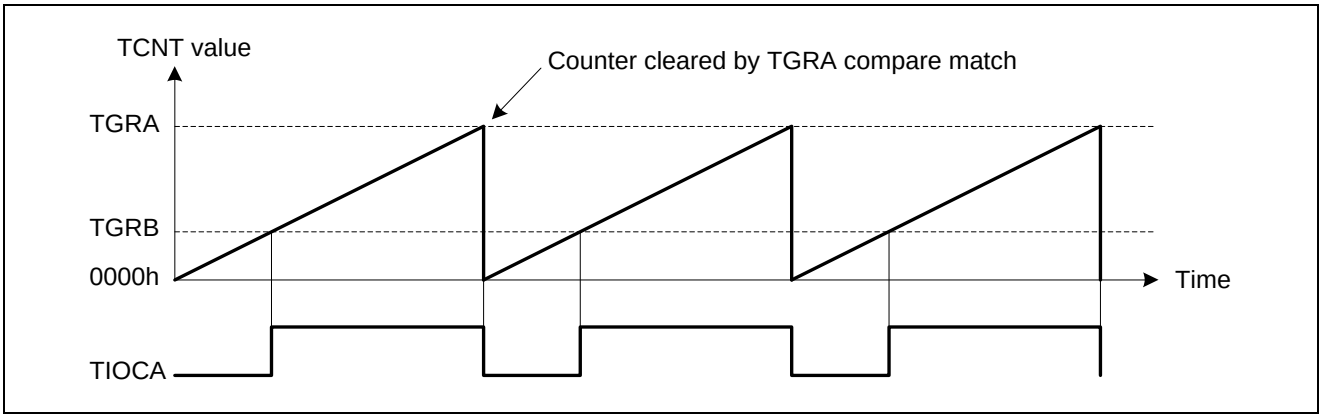



**Figure 15.22 Example of PWM Mode Operation (1)**

Figure 15.23 shows an example of PWM mode 2 operation.

In this example, synchronous operation is specified for TPU0 and TPU1, TPU1.TGRB compare match is set as the TPUm.TCNT clearing source, and 0 is set for the initial output value and 1 for the output value of the other TPUm.TGRy registers (TPU0.TGRA to TPU0.TGRD and TPU1.TGRA), to output a 5-phase PWM waveform.

In this case, the value set in TPU1.TGRB is used as the cycle, and the values set in the other TGRy registers are used as the duty cycle.

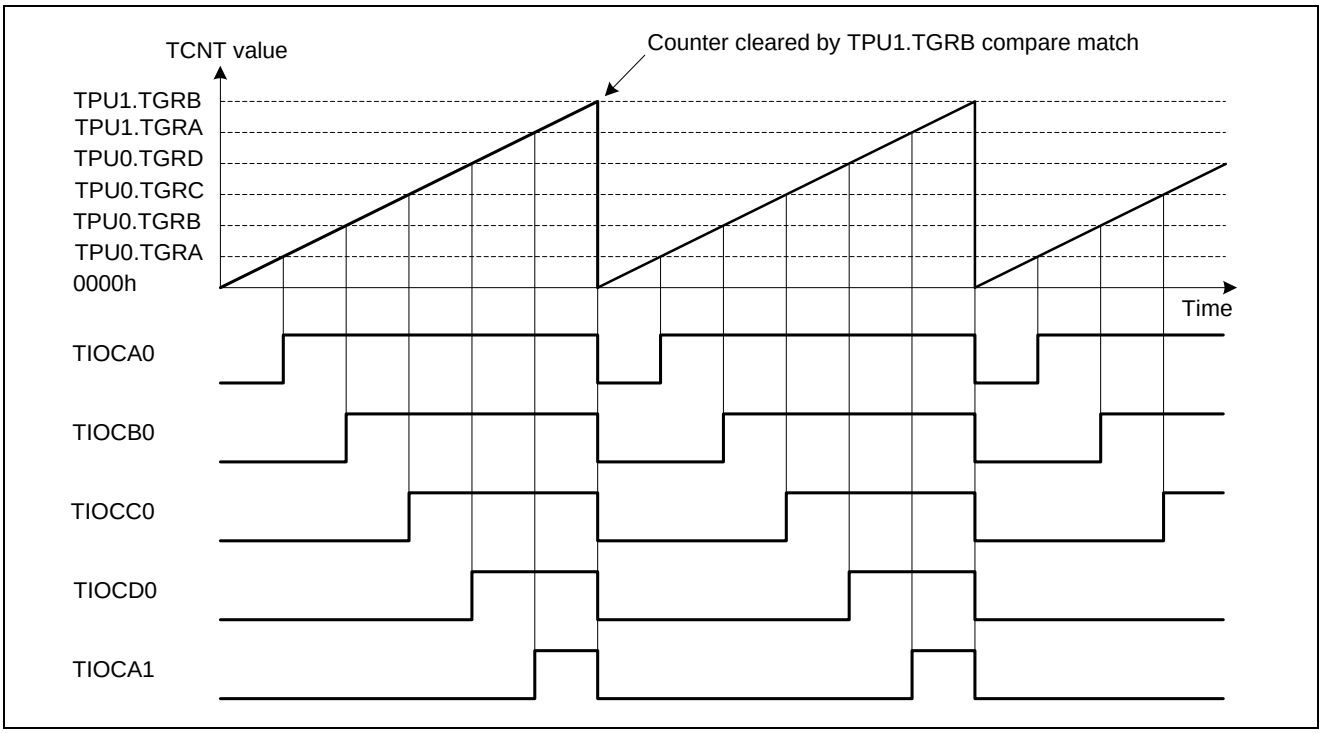



**Figure 15.23 Example of PWM Mode Operation (2)**



Downloaded from Arrow.com.

Figure 15.24 shows examples of PWM waveform output with 0% duty cycle and 100% duty cycle in PWM mode.

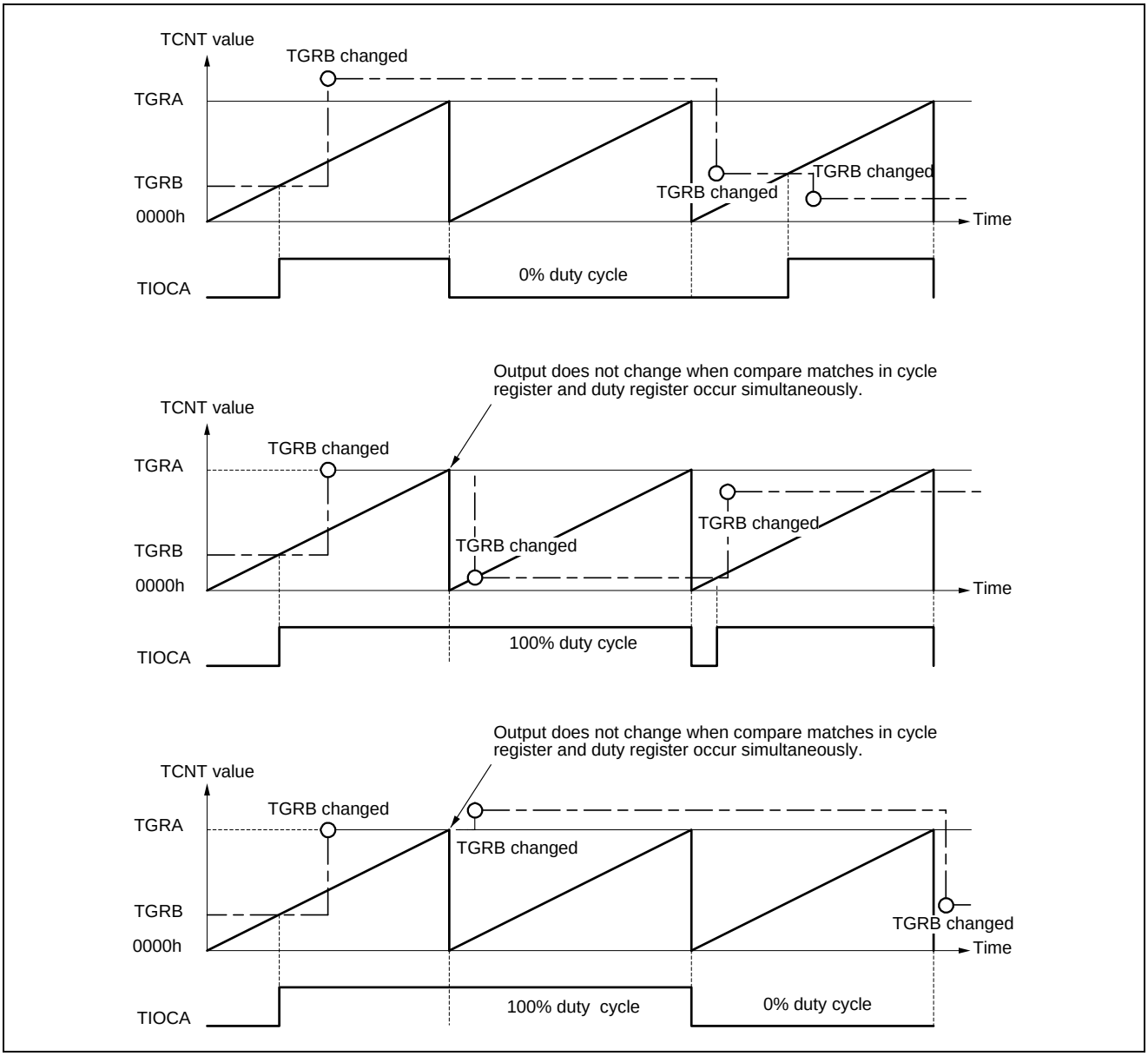



**Figure 15.24 Example of PWM Mode Operation (3)**



Downloaded from Arrow.com.

### 15.3.6 Phase Counting Mode

In phase counting mode, the phase difference between two external clock inputs is detected by the settings for channels 1, 2, 4, and 5 (unit 0) and channels 7, 8, 10, and 11 (unit 1), and TPUm.TCNT is incremented/decremented accordingly.

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 the TPSC[2:0] bits and CKEG[1:0] bits in TPUm.TCR. However, the lower 2 bits of the CCLR[2:0] bits in TPUm.TCR and the functions of TPUm.TIORH, TPUm.TIORL, TPUm.TIOR, TPUm.TIER, and TPUm.TGRy are valid, and therefore input capture/compare match and interrupt functions are available.

This can be used for two-phase encoder pulse input.

When an overflow occurs while TCNT is counting up, a TCIV interrupt request is generated; when an underflow occurs while TCNT is counting down, a TCIU interrupt request is generated. The TCFD bit in TPUm.TSR is the count direction flag. Reading the TCFD flag provides an indication of whether TCNT is counting up or down.

Table 15.26 shows the correspondence between external clock pins and channels.

**Table 15.26 Clock Input Pins in Phase Counting Mode**

| | | External Clock Pins | |
|---|---|---|---|
| **Unit** | **Channel** | **A-Phase** | **B-Phase** |
| 0 | When TPU1 or TPU5 is set to phase counting mode | TCLKA | TCLKB |
| | When TPU2 or TPU4 is set to phase counting mode | TCLKC | TCLKD |
| 1 | When TPU7 or TPU11 is set to phase counting mode | TCLKE | TCLKF |
| | When TPU8 or TPU10 is set to phase counting mode | TCLKG | TCLKH |

#### (1) Example of Phase Counting Mode Setting Procedure

Figure 15.25 shows an example of the phase counting mode setting procedure.

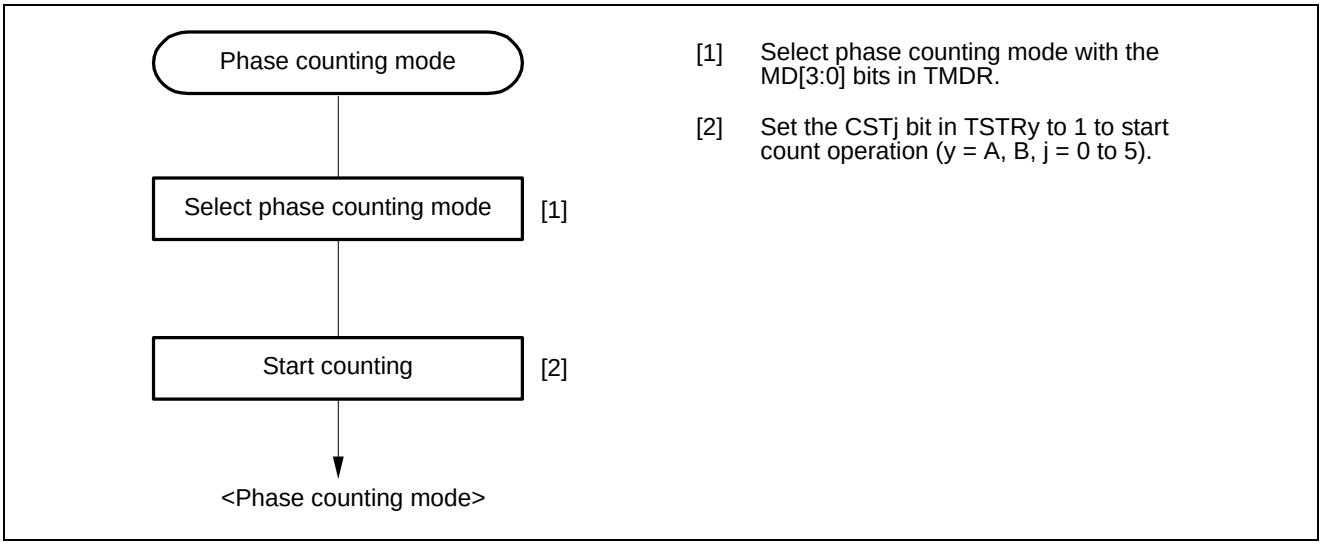



**Figure 15.25 Example of Phase Counting Mode Setting Procedure**



Downloaded from Arrow.com.

### (2) Examples of Phase Counting Mode Operation

In phase counting mode, TPUm.TCNT counts up or down according to the phase difference between two external clocks. There are four modes, according to the count conditions.

#### (a) Phase counting mode 1

Figure 15.26 shows an example of phase counting mode 1 operation, and table 15.27 lists the TCNTn up-/down-count conditions.

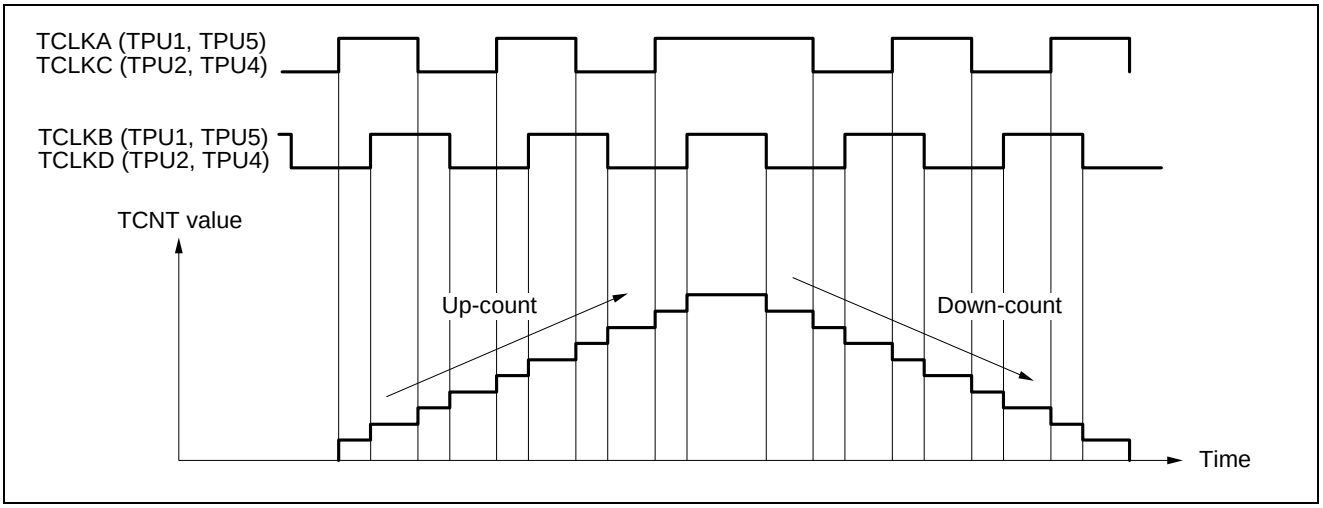



**Figure 15.26 Example of Phase Counting Mode 1 Operation**

**Table 15.27 Up-/Down-Count Conditions in Phase Counting Mode 1**

| TCLKA (TPU1, TPU5) / TCLKC (TPU2, TPU4) | TCLKB (TPU1, TPU5) / TCLKD (TPU2, TPU4) | Operation |
|---|---|---|
| High level | ↑ | Up-count |
| Low level | ↓ | |
| ↑ | Low level | |
| ↓ | High level | |
| High level | ↓ | Down-count |
| Low level | ↑ | |
| ↑ | High level | |
| ↓ | Low level | |

[Legend]
↑: Rising edge
↓: Falling edge



Downloaded from Arrow.com.

### (b) Phase counting mode 2

Figure 15.27 shows an example of phase counting mode 2 operation, and table 15.28 lists the TPUm.TCNT up-/down-count conditions.

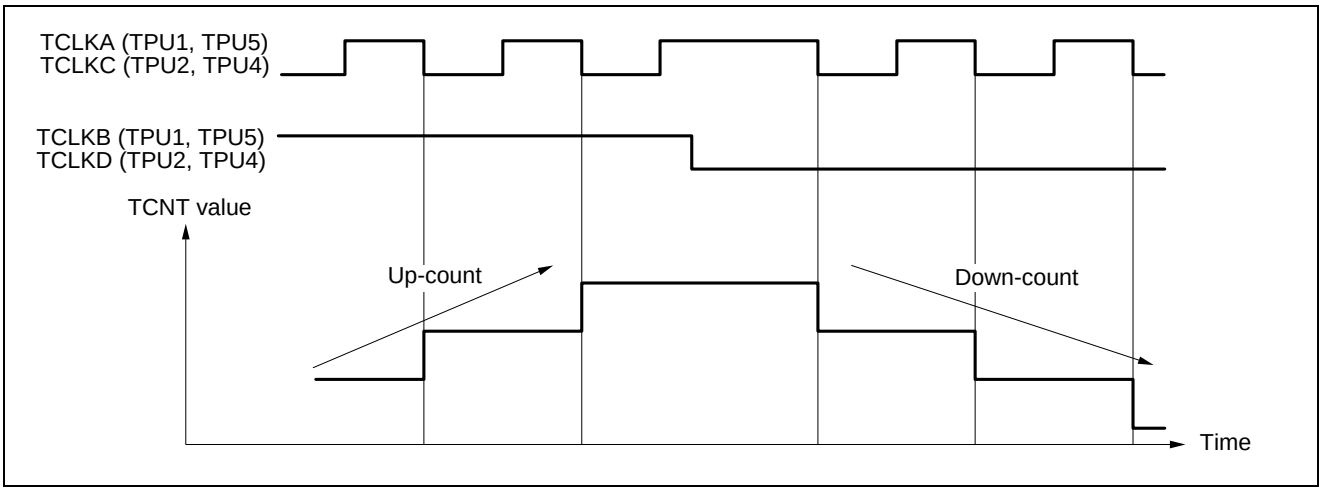



**Figure 15.27 Example of Phase Counting Mode 2 Operation**

**Table 15.28 Up-/Down-Count Conditions in Phase Counting Mode 2**

| TCLKA (TPU1, TPU5) / TCLKC (TPU2, TPU4) | TCLKB (TPU1, TPU5) / TCLKD (TPU2, TPU4) | Operation |
|---|---|---|
| High level | ↑ | Don't care |
| Low level | ↓ | Don't care |
| ↑ | Low level | Don't care |
| ↓ | High level | Up-count |
| High level | ↓ | Don't care |
| Low level | ↑ | Don't care |
| ↑ | High level | Don't care |
| ↓ | Low level | Down-count |

[Legend]
↑ : Rising edge
↓ : Falling edge



Downloaded from Arrow.com.

### (c) Phase counting mode 3

Figure 15.28 shows an example of phase counting mode 3 operation, and table 15.29 lists the TPUm.TCNT up-/down-count conditions.

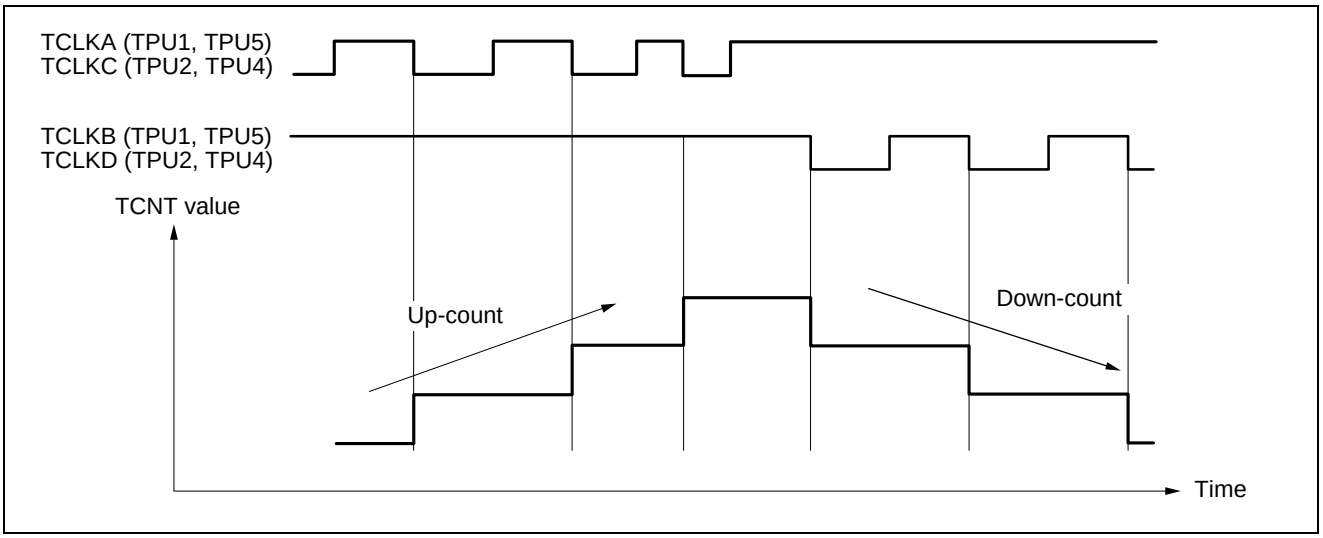



**Figure 15.28 Example of Phase Counting Mode 3 Operation**

**Table 15.29 Up-/Down-Count Conditions in Phase Counting Mode 3**

| TCLKA (TPU1, TPU5) TCLKC (TPU2, TPU4) | TCLKB (TPU1, TPU5) TCLKD (TPU2, TPU4) | Operation |
|---|---|---|
| High level | ↑ | Don't care |
| Low level | ↓ | Don't care |
| ↑ | Low level | Don't care |
| ↓ | High level | Up-count |
| High level | ↓ | Down-count |
| Low level | ↑ | Don't care |
| ↑ | High level | Don't care |
| ↓ | Low level | Don't care |

[Legend]
↑: Rising edge
↓: Falling edge



Downloaded from Arrow.com.

### (d) Phase counting mode 4

Figure 15.29 shows an example of phase counting mode 4 operation, and table 15.30 lists the TPUm.TCNT up-/down-count conditions.

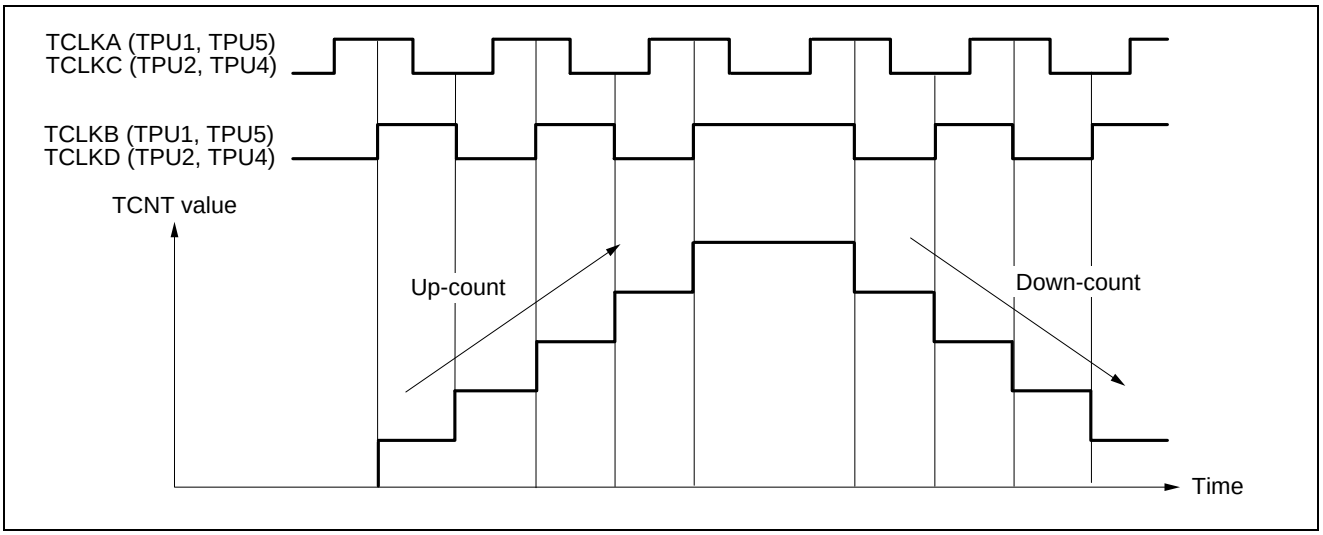



**Figure 15.29 Example of Phase Counting Mode 4 Operation**

**Table 15.30 Up-/Down-Count Conditions in Phase Counting Mode 4**

| TCLKA (TPU1, TPU5) TCLKC (TPU2, TPU4) | TCLKB (TPU1, TPU5) TCLKD (TPU2, TPU4) | Operation |
|---|---|---|
| High level | ↑ | Up-count |
| Low level | ↓ | |
| ↑ | Low level | Don't care |
| ↓ | High level | |
| High level | ↓ | Down-count |
| Low level | ↑ | |
| ↑ | High level | Don't care |
| ↓ | Low level | |

[Legend]
↑ : Rising edge
↓ : Falling edge



Downloaded from Arrow.com.

### 15.3.6.1 Phase Counting Mode Application Example

Figure 15.30 shows an example in which phase counting mode is set for TPU1, and TPU1 is coupled with TPU0 to input servo motor 2-phase encoder pulses in order to detect the position or speed.

TPU1 is set to phase counting mode 1, and the encoder pulse A-phase and B-phase are input to the TCLKA and TCLKB pins.

TPU0 operates with TPU0.TCNT counter clearing by TPU0.TGRC compare match; TPU0.TGRA and TPU0.TGRC are used for the compare match function and are set with the speed control cycle and position control cycle. TPU0.TGRB is used for input capture, with TPU0.TGRB and TPU0.TGRD operating in buffer mode. The TPU1 counter input clock is specified as the TPU0.TGRB input capture source, and the pulse width of 2-phase encoder 4-multiplication pulses is detected.

TPU1.TGRA and TPU1.TGRB for TPU1 are specified for input capture, TPU0.TGRA and TPU0.TGRC compare matches are selected as the input capture source, and the up-/down-counter values for the control cycles are stored.

This procedure enables accurate position/speed detection to be achieved.

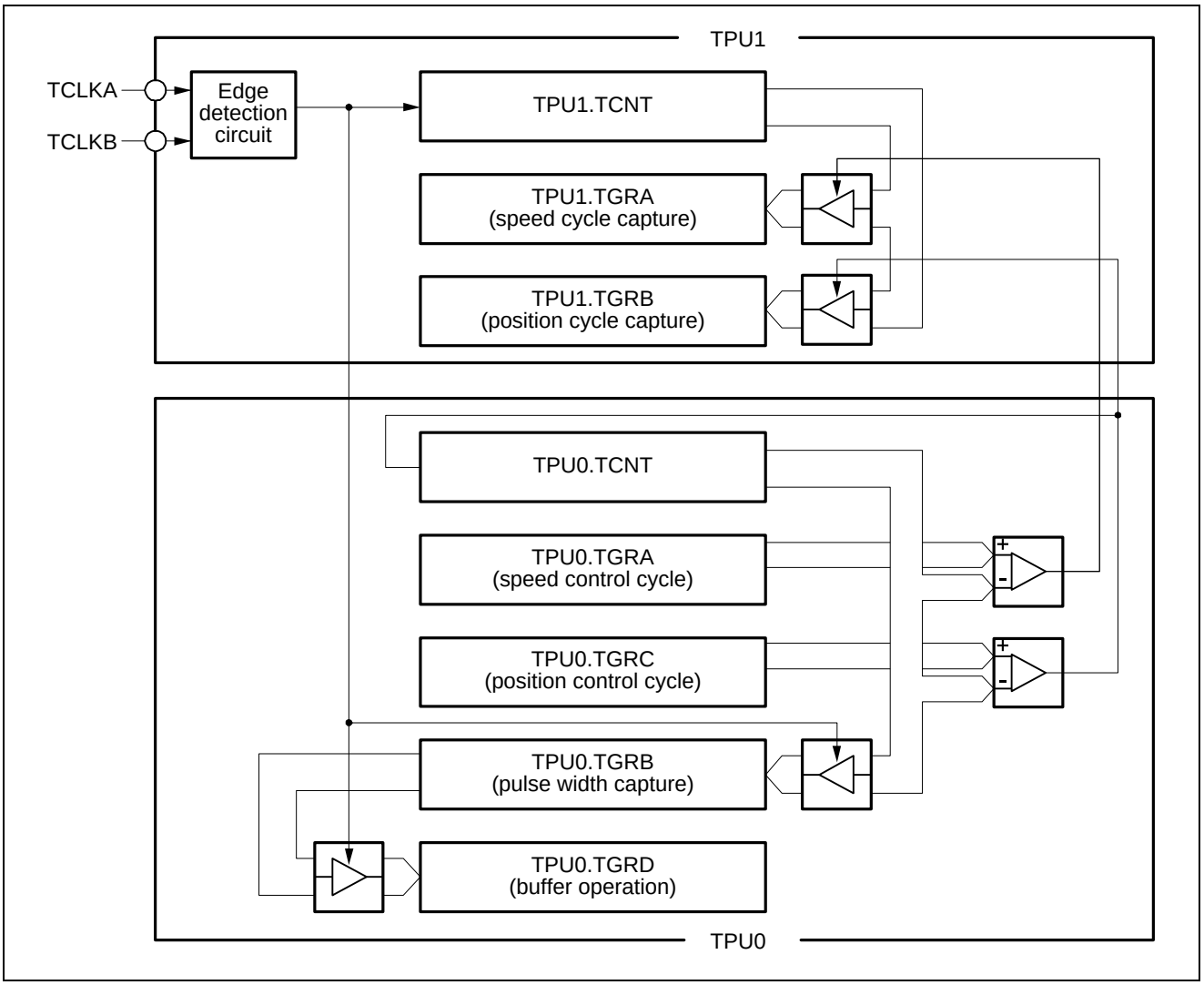



**Figure 15.30 Phase Counting Mode Application Example**



Downloaded from Arrow.com.

## 15.4 Interrupt Sources

There are three kinds of TPU interrupt sources: TPUm.TGRy input capture/compare match, TPUm.TCNT overflow, and TPUm.TCNT underflow.

Relative channel priority levels can be changed by the interrupt controller, but the priority within a channel is fixed. For details, see section 10, Interrupt Control Unit (ICU).



Downloaded from Arrow.com.

Table 15.31 lists the TPU interrupt sources.

**Table 15.31 TPU Interrupts**

| Unit | Channel | Name | Interrupt Source | DTC Activation | DMAC Activation |
|---|---|---|---|---|---|
| 0 | TPU0 | TGI0A | TPU0.TGRA input capture/compare match | Possible | Possible |
| | | TGI0B | TPU0.TGRB input capture/compare match | Possible | Not possible |
| | | TGI0C | TPU0.TGRC input capture/compare match | Possible | Not possible |
| | | TGI0D | TPU0.TGRD input capture/compare match | Possible | Not possible |
| | | TCI0V | TPU0.TCNT overflow | Not poss ble | Not possible |
| | TPU1 | TGI1A | TPU1.TGRA input capture/compare match | Possible | Possible |
| | | TGI1B | TPU1.TGRB input capture/compare match | Possible | Not possible |
| | | TCI1V | TPU1.TCNT overflow | Not poss ble | Not possible |
| | | TCI1U | TPU1.TCNT underflow | Not poss ble | Not possible |
| | TPU2 | TGI2A | TPU2.TGRA input capture/compare match | Possible | Possible |
| | | TGI2B | TPU2.TGRB input capture/compare match | Possible | Not possible |
| | | TCI2V | TPU2.TCNT overflow | Not poss ble | Not possible |
| | | TCI2U | TPU2.TCNT underflow | Not poss ble | Not possible |
| | TPU3 | TGI3A | TPU3.TGRA input capture/compare match | Possible | Possible |
| | | TGI3B | TPU3.TGRB input capture/compare match | Possible | Not possible |
| | | TGI3C | TPU3.TGRC input capture/compare match | Possible | Not possible |
| | | TGI3D | TPU3.TGRD input capture/compare match | Possible | Not possible |
| | | TCI3V | TPU3.TCNT overflow | Not poss ble | Not possible |
| | TPU4 | TGI4A | TPU4.TGRA input capture/compare match | Possible | Possible |
| | | TGI4B | TPU4.TGRB input capture/compare match | Possible | Not possible |
| | | TCI4V | TPU4.TCNT overflow | Not poss ble | Not possible |
| | | TCI4U | TPU4.TCNT underflow | Not poss ble | Not possible |
| | TPU5 | TGI5A | TPU5.TGRA input capture/compare match | Possible | Possible |
| | | TGI5B | TPU5.TGRB input capture/compare match | Possible | Not possible |
| | | TCI5V | TPU5.TCNT overflow | Not poss ble | Not possible |
| | | TCI5U | TPU5.TCNT underflow | Not poss ble | Not possible |
| 1 | TPU6 | TGI6A | TPU6.TGRA input capture/compare match | Possible | Possible |
| | | TGI6B | TPU6.TGRB input capture/compare match | Possible | Not possible |
| | | TGI6C | TPU6.TGRC input capture/compare match | Possible | Not possible |
| | | TGI6D | TPU6.TGRD input capture/compare match | Possible | Not possible |
| | | TCI6V | TPU6.TCNT overflow | Not poss ble | Not possible |
| | TPU7 | TGI7A | TPU7.TGRA input capture/compare match | Possible | Possible |
| | | TGI7B | TPU7.TGRB input capture/compare match | Possible | Not possible |
| | | TCI7V | TPU7.TCNT overflow | Not poss ble | Not possible |
| | | TCI7U | TPU7.TCNT underflow | Not poss ble | Not possible |
| | TPU8 | TGI8A | TPU8.TGRA input capture/compare match | Possible | Possible |
| | | TGI8B | TPU8.TGRB input capture/compare match | Possible | Not possible |
| | | TCI8V | TPU8.TCNT overflow | Not poss ble | Not possible |
| | | TCI8U | TPU8.TCNT underflow | Not poss ble | Not possible |



Downloaded from Arrow.com.

| | | | | | |
|---|---|---|---|---|---|
| 1 | TPU9 | TGI9A | TPU9.TGRA input capture/compare match | Possible | Possible |
| | | TGI9B | TPU9.TGRB input capture/compare match | Possible | Not possible |
| | | TGI9C | TPU9.TGRC input capture/compare match | Possible | Not possible |
| | | TGI9D | TPU9.TGRD input capture/compare match | Possible | Not possible |
| | | TCI9V | TPU9.TCNT overflow | Not poss ble | Not possible |
| | TPU10 | TGI10A | TPU10.TGRA input capture/compare match | Possible | Possible |
| | | TGI10B | TPU10.TGRB input capture/compare match | Possible | Not possible |
| | | TCI10V | TPU10.TCNT overflow | Not poss ble | Not possible |
| | | TCI10U | TPU10.TCNT underflow | Not poss ble | Not possible |
| | TPU11 | TGI11A | TPU11.TGRA input capture/compare match | Possible | Possible |
| | | TGI11B | TPU11.TGRB input capture/compare match | Possible | Not possible |
| | | TCI11V | TPU11.TCNT overflow | Not poss ble | Not possible |
| | | TCI11U | TPU11.TCNT underflow | Not poss ble | Not possible |

Note: This table shows the initial state immediately after a reset. The relative channel priority levels can be changed by the interrupt controller.

### (1) Input Capture/Compare Match Interrupt

An interrupt is requested when the TGIEy bit (y = A, B, C, D) in TPUm.TIER is set to 1 by the occurrence of a TPUm.TGRy input capture/compare match on a channel. The TPU has 32 input capture/compare match interrupts, four each for TPU0 and TPU3 (TPU6 and TPU9), and two each for TPU1, TPU2, TPU4, and TPU5 (TPU7, TPU8, TPU10, and TPU11).

### (2) Overflow Interrupt

An interrupt is requested when the TCIEV bit in TPUm.TIER is set to 1 by the occurrence of a TPUm.TCNT overflow on a channel. The TPU has twelve overflow interrupts, one for each channel.

### (3) Underflow Interrupt

An interrupt is requested when the TCIEU bit in TPUm.TIER is set to 1 by the occurrence of a TPUm.TCNT underflow on a channel. The TPU has eight underflow interrupts, one each for TPU1, TPU2, TPU4, and TPU5 (TPU7, TPU8, TPU10, and TPU11).



Downloaded from Arrow.com.

## 15.5 DTC Activation

The DTC can be activated by the TPUm.TGRy input capture/compare match interrupt of each channel. For details, see section 13, Data Transfer Controller (DTC).

A total of 32 input capture/compare match interrupts can be used as DTC activation sources, four each for TPU0 and TPU3 (TPU6 and TPU9), and two each for TPU1, TPU2, TPU4, and TPU5 (TPU7, TPU8, TPU10, and TPU11).

## 15.6 DMAC Activation

The DMAC can be activated by the TPUm.TGRA input capture/compare match interrupt of each channel. For details, see section 12, DMA controller (DMAC).

A total of twelve TPUm.TGRA input capture/compare match interrupts can be used as DMAC activation sources, one for each channel.

## 15.7 A/D Converter Activation

The TPU can activate the A/D converter by the TPUm.TGRA input capture/compare match for each channel. Moreover, the A/D converter is activated by the TGRA to TGRD input capture/compare match from TPU0.*

When the TTGE bit in TPUm.TIER is set to 1, the TPU requests the A/D converter to start A/D conversion by the occurrence of a TPUm.TGRA input capture/compare match on a particular channel. If the TPU conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is started. Moreover, the TPU requests the A/D converter to start A/D conversion by the occurrence of a TGRA to TGRD input capture/compare match from TPU0 . If the TPU conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is started.

Note: * For the corresponding unit of A/D converter, see section 23, A/D Converter.



Downloaded from Arrow.com.

## 15.8 Operation Timing

### 15.8.1 Input/Output Timing

#### (1) TPUm.TCNT Count Timing

Figure 15.31 shows TPUm.TCNT count timing in internal clock operation, and figure 15.32 shows TCNT count timing in external clock operation.

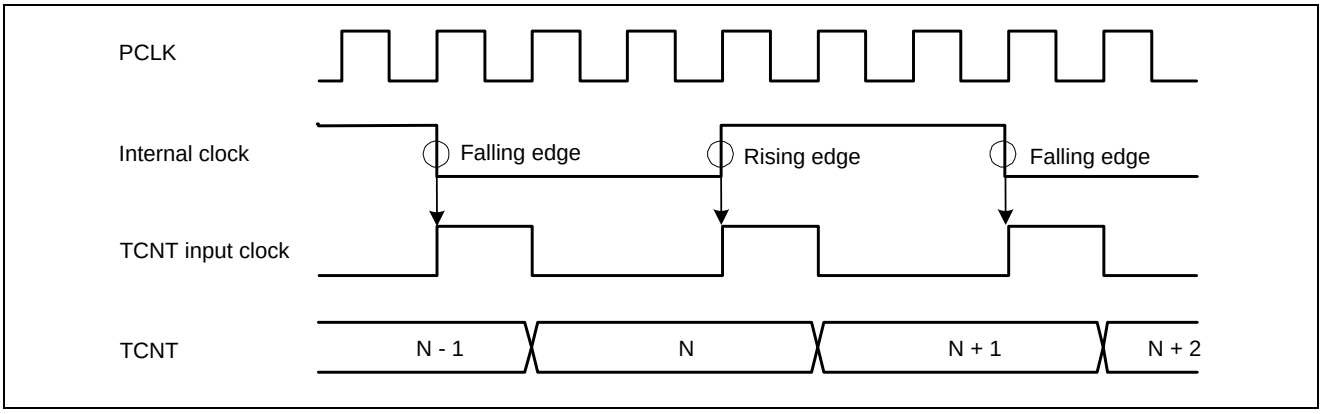



**Figure 15.31 Count Timing in Internal Clock Operation**

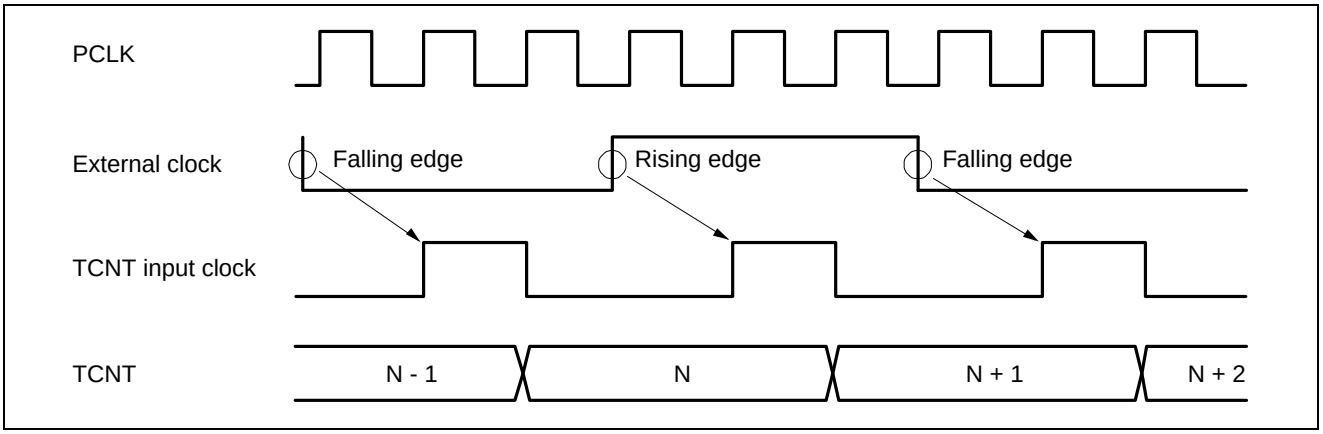



**Figure 15.32 Count Timing in External Clock Operation**



Downloaded from Arrow.com.

### (2) Output Compare Output Timing

A compare match signal is generated in the final state in which TPUm.TCNT and TPUm.TGRy 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 TPUm.TIORH, TPUm.TIORL, or TPUm.TIOR is output to the output compare output pin TIOCyn (y = A to D, n = 0 to 11). After a match between TCNT and TGRy, the compare match signal is not generated until the TCNT input clock is generated.

Figure 15.33 shows output compare output timing.

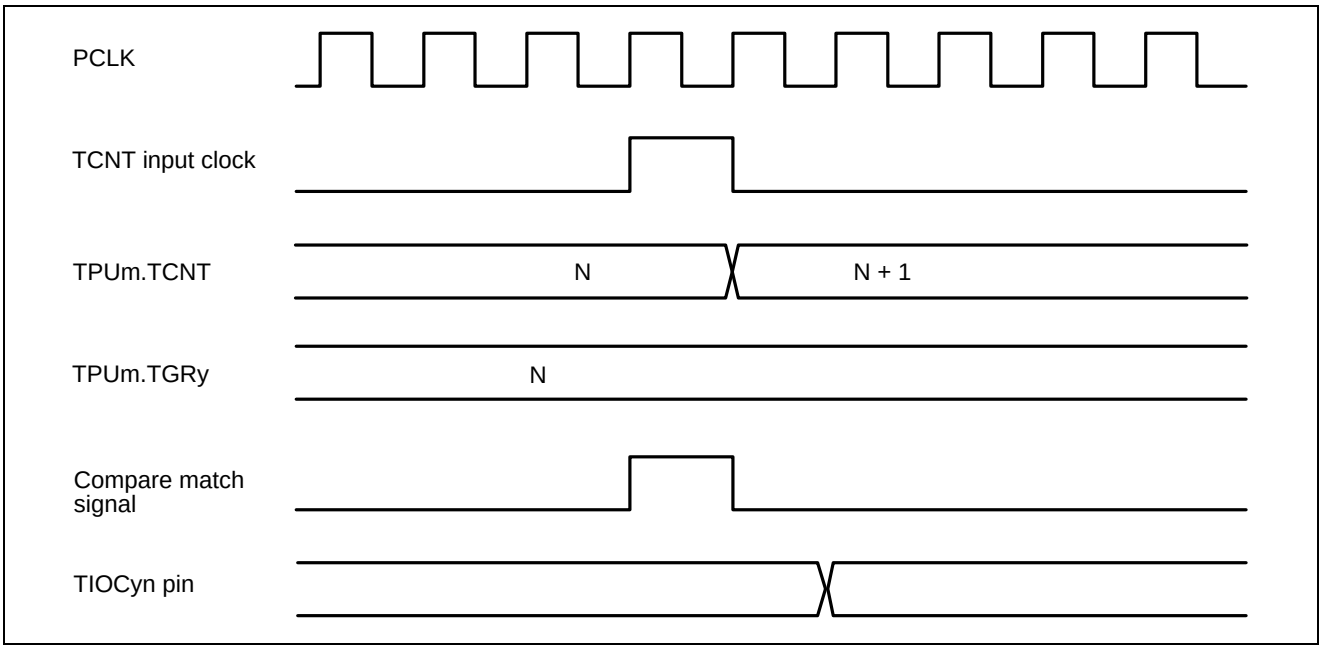



**Figure 15.33 Output Compare Output Timing**

### (3) Input Capture Signal Timing

Figure 15.34 shows input capture signal timing.

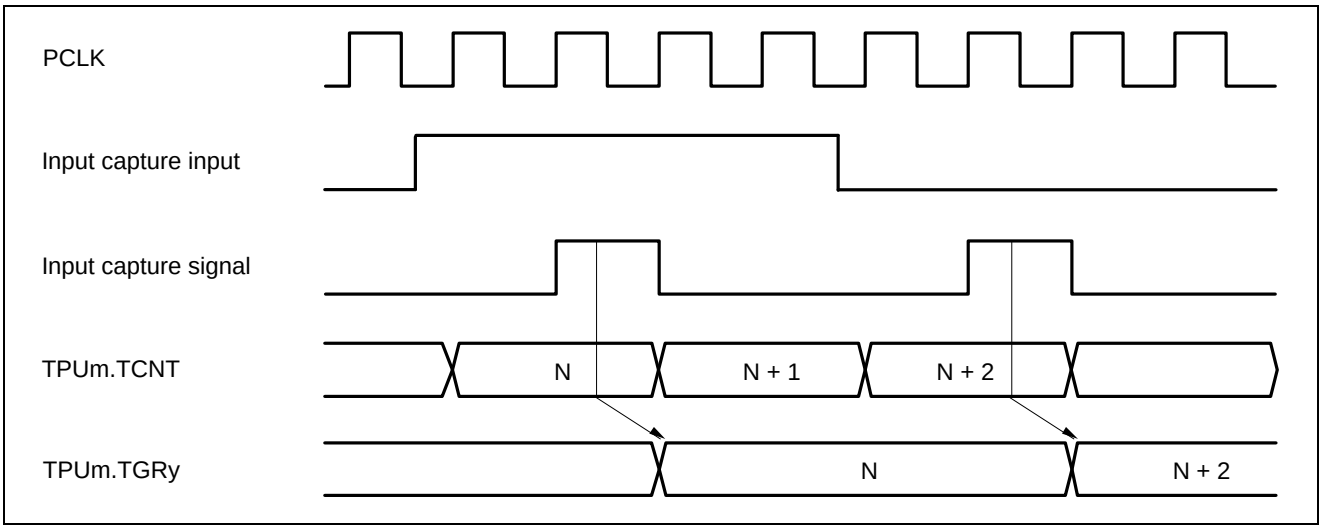



**Figure 15.34 Input Capture Signal Timing**



Downloaded from Arrow.com.

### (4) Timing for Counter Clearing by Compare Match/Input Capture

Figure 15.35 shows the timing when counter clearing by compare match occurrence is specified, and figure 15.36 shows the timing when counter clearing by input capture occurrence is specified.

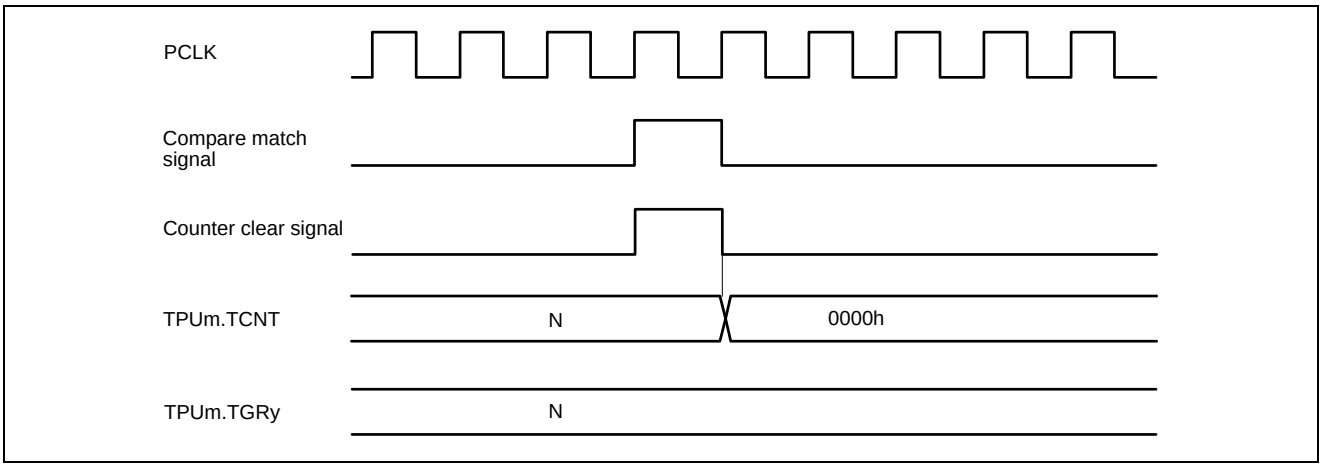



**Figure 15.35 Counter Clear Timing (Compare Match)**

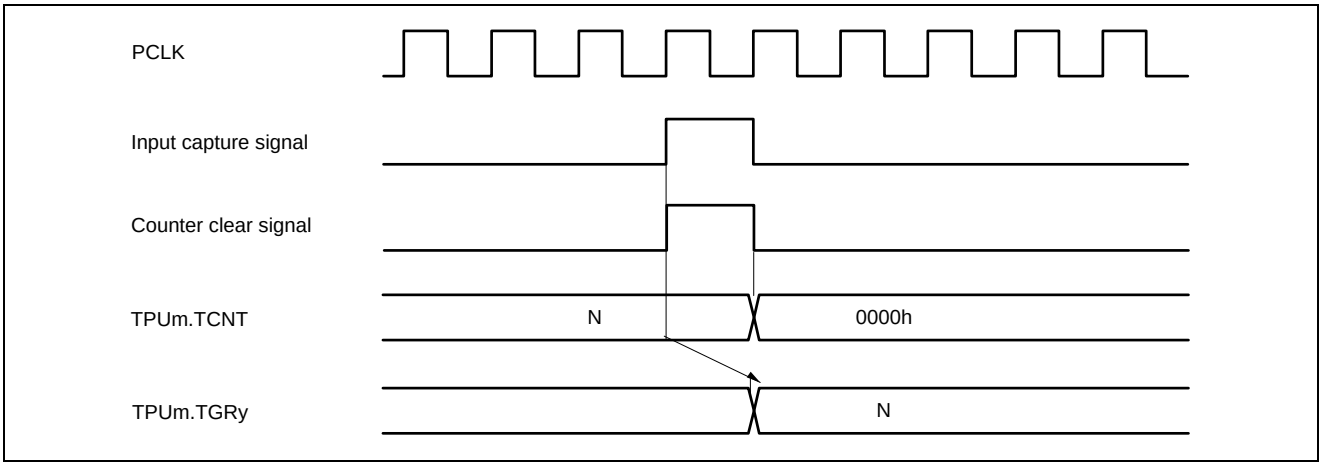



**Figure 15.36 Counter Clear Timing (Input Capture)**



Downloaded from Arrow.com.

### (5) Buffer Operation Timing

Figures 15.37 and 15.38 show the timings in buffer operation.

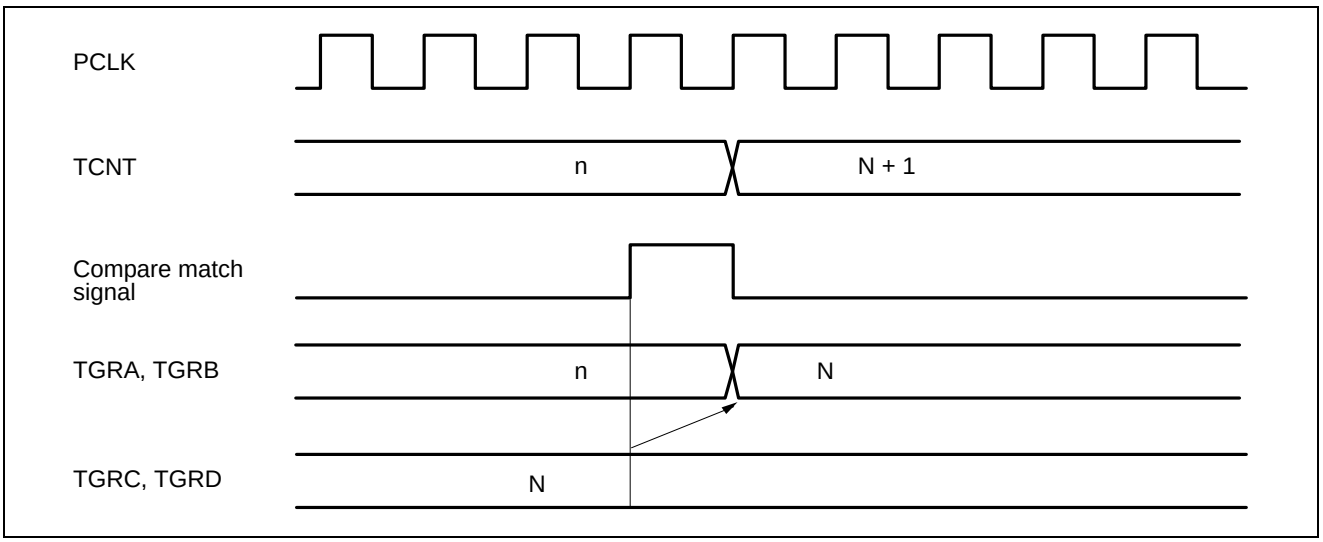



**Figure 15.37 Buffer Operation Timing (Compare Match)**

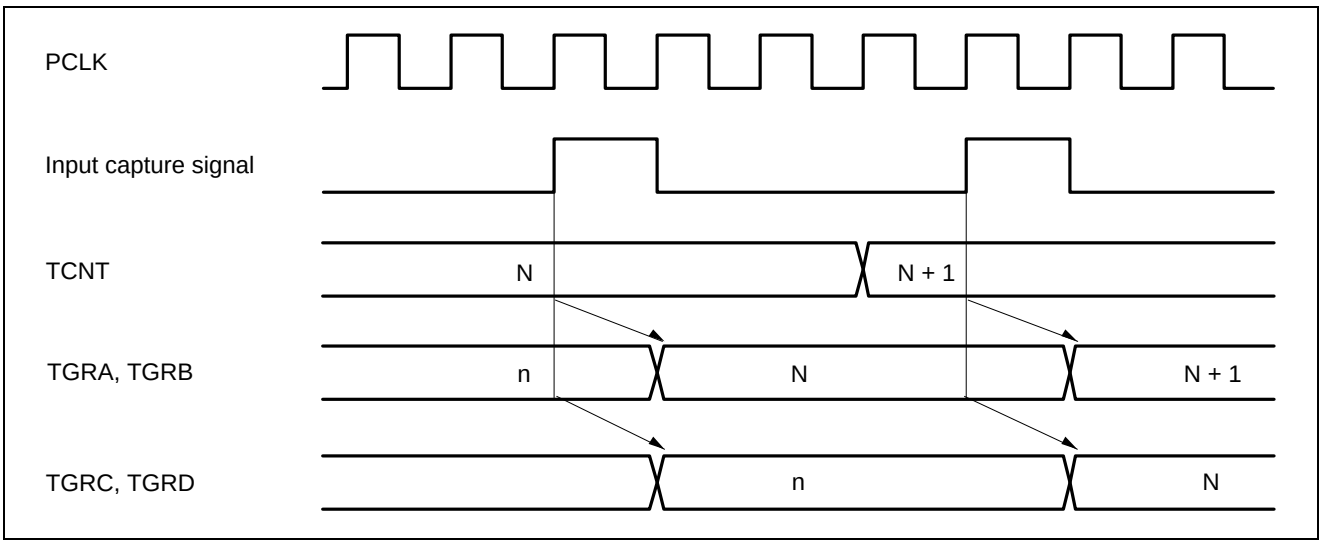



**Figure 15.38 Buffer Operation Timing (Input Capture)**



Downloaded from Arrow.com.

## 15.8.2 Interrupt Signal Timing

### (1) Interrupt Flag Setting to 1 in Case of Compare Match

Figure 15.39 shows the timing for setting the interrupt flag by compare match occurrence.

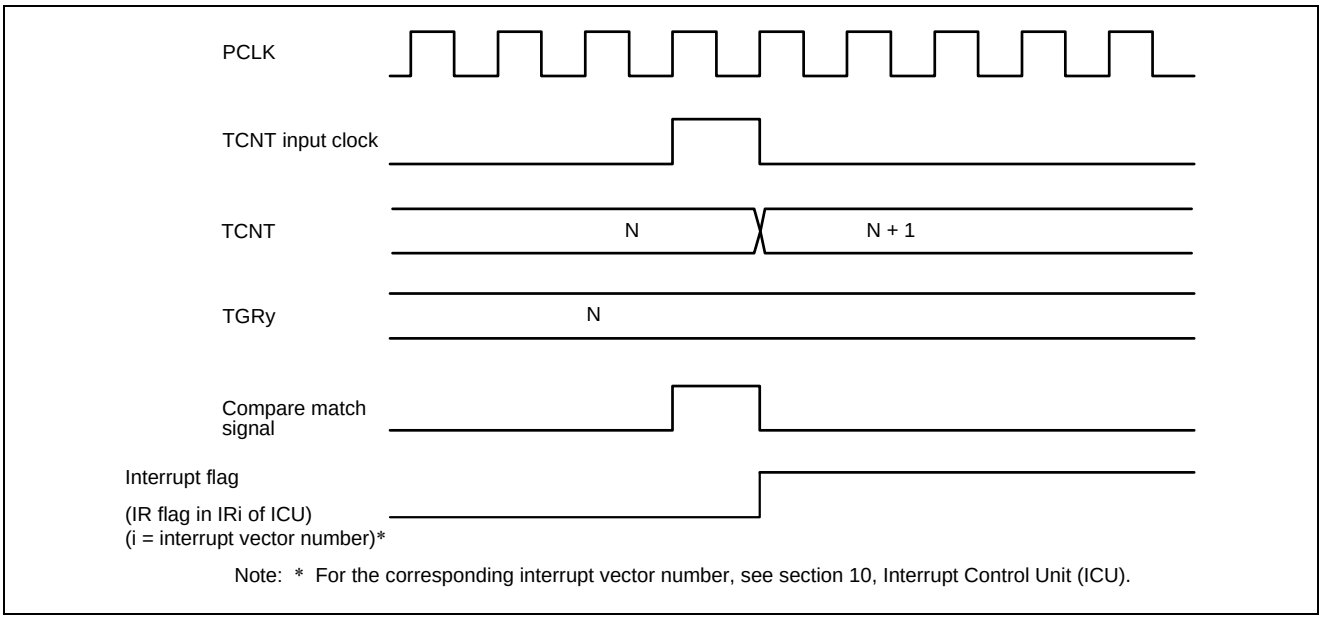



**Figure 15.39 TGImy Interrupt Timing (Compare Match)**

### (2) Interrupt Flag Setting to 1 in Case of Input Capture

Figure 15.40 shows the timing for setting the interrupt flag by input capture occurrence.

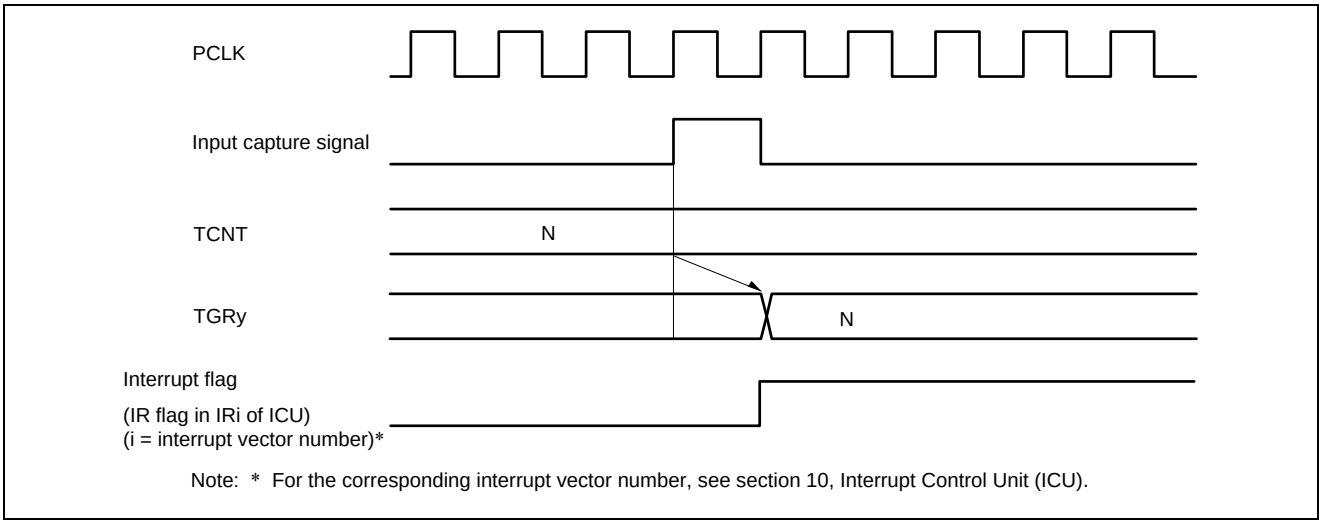



**Figure 15.40 TGImy Interrupt Timing (Input Capture)**



Downloaded from Arrow.com.

### (3) TCImV/TCImU Interrupt Flag Setting to 1

Figure 15.41 shows the timing for generating the TCImV interrupt request signal by overflow occurrence.

Figure 15.42 shows the timing for generating the TCImU interrupt request signal by underflow occurrence.

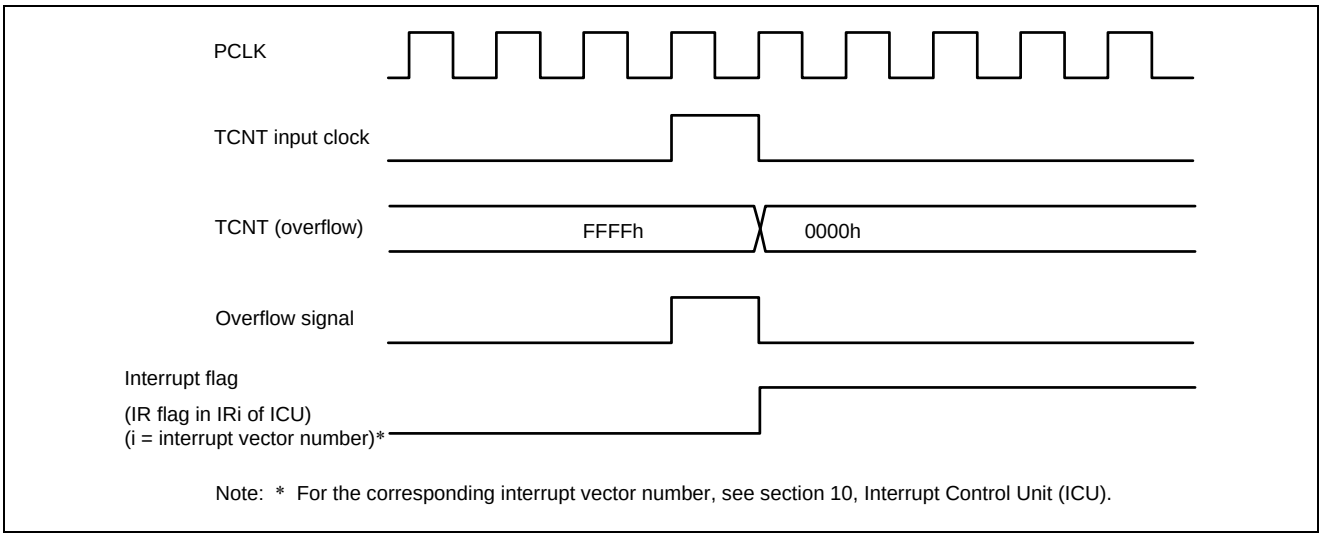



**Figure 15.41 TCImV Interrupt Setting Timing**

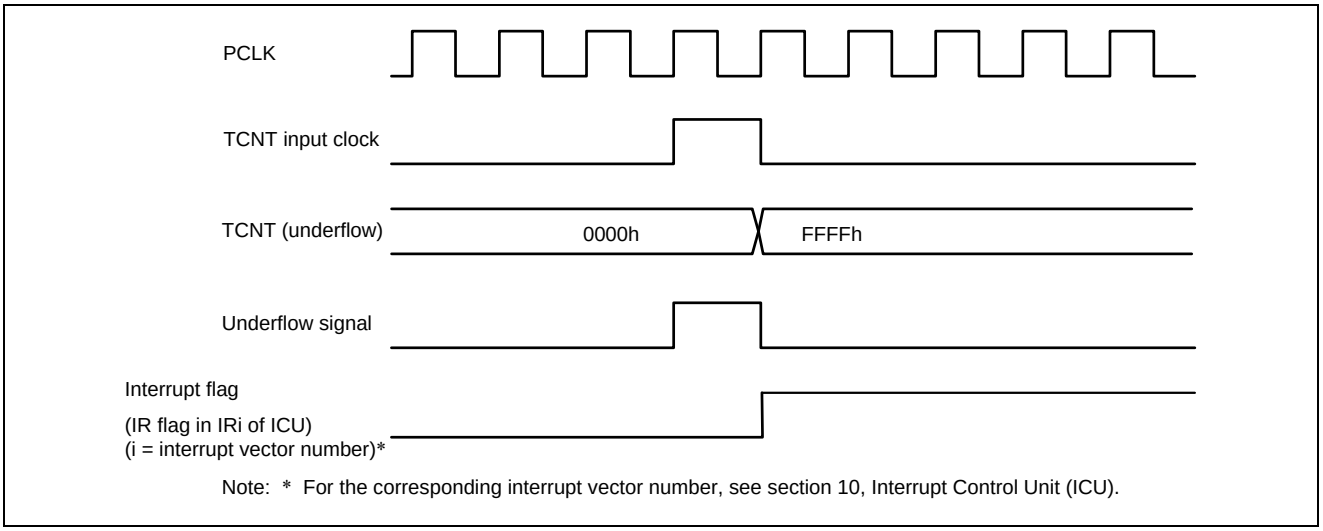



**Figure 15.42 TCImU Interrupt Setting Timing**



Downloaded from Arrow.com.

## 15.9 Usage Notes

### 15.9.1 Module Stop Function Setting

Operation of the TPU can be disabled or enabled using the module stop control register. The TPU does not operate with the initial setting. Register access is enabled by clearing module stop state. For details, see section 8, Low Power Consumption.

### 15.9.2 Input Clock Restrictions

The input clock pulse width must be at least 1.5 states in the case of single-edge detection, and at least 2.5 states in the case of both-edge detection. The TPU will not operate properly 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 15.43 shows the input clock conditions in phase counting mode.

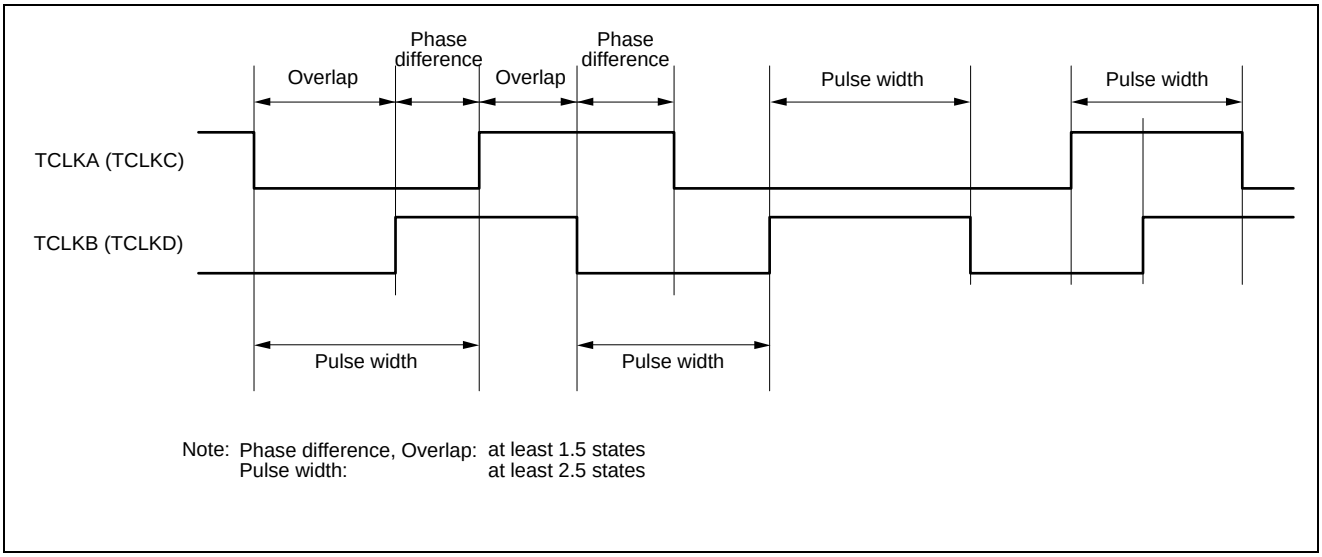



**Figure 15.43 Phase Difference, Overlap, and Pulse Width in Phase Counting Mode**



Downloaded from Arrow.com.

### 15.9.3 Caution on Cycle Setting

When counter clearing by compare match is set, TPUm.TCNT is cleared in the final state in which it matches the TPUm.TGRy value (the point at which the count value matched by TCNT is updated). Consequently, the actual counter frequency is given by the following formula:

$$f = \frac{PCLK}{(N + 1)}$$

f: Counter frequency
PCLK: Operating frequency
N: TGRy set value

### 15.9.4 Conflict between TPUm.TCNT Write and Clear Operations

If the counter clearing signal is generated in a TCNT write cycle, TCNT clearing takes precedence and the TCNT write is not performed. Figure 15.44 shows the timing in this case.

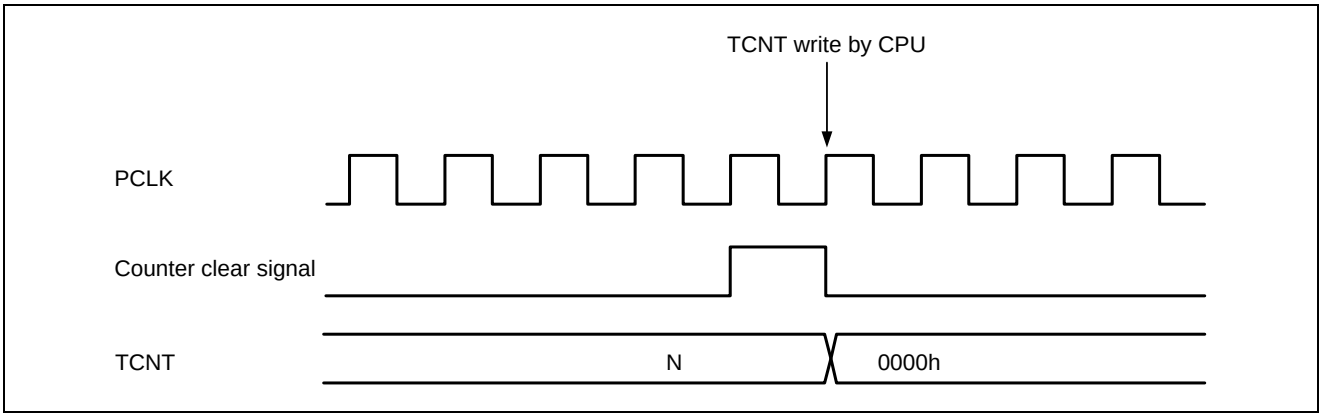



**Figure 15.44 Conflict between TPUm.TCNT Write and Clear Operations**

### 15.9.5 Conflict between TPUm.TCNT Write and Increment Operations

If incrementing occurs in a TCNT write cycle, the TCNT write takes precedence and TCNT is not incremented. Figure 15.45 shows the timing in this case.

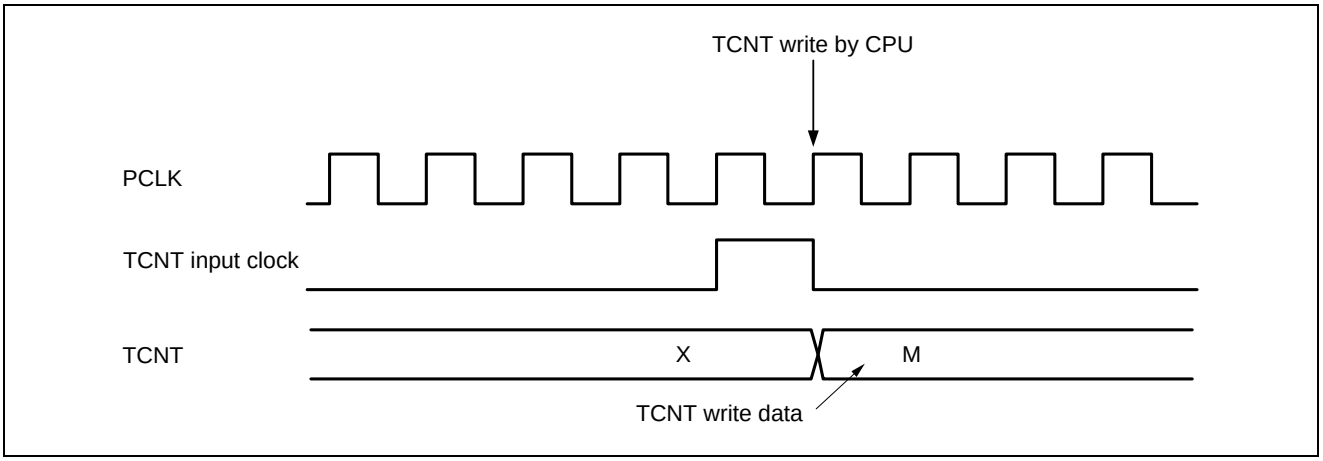



**Figure 15.45 Conflict between TPUm.TCNT Write and Increment Operations**



Downloaded from Arrow.com.

### 15.9.6 Conflict between TPUm.TGRy Write and Compare Match

If a compare match occurs in a TGRy write cycle, the TGRy write takes precedence and the compare match signal is disabled. A compare match also does not occur when the same value as before is written.

Figure 15.46 shows the timing in this case.

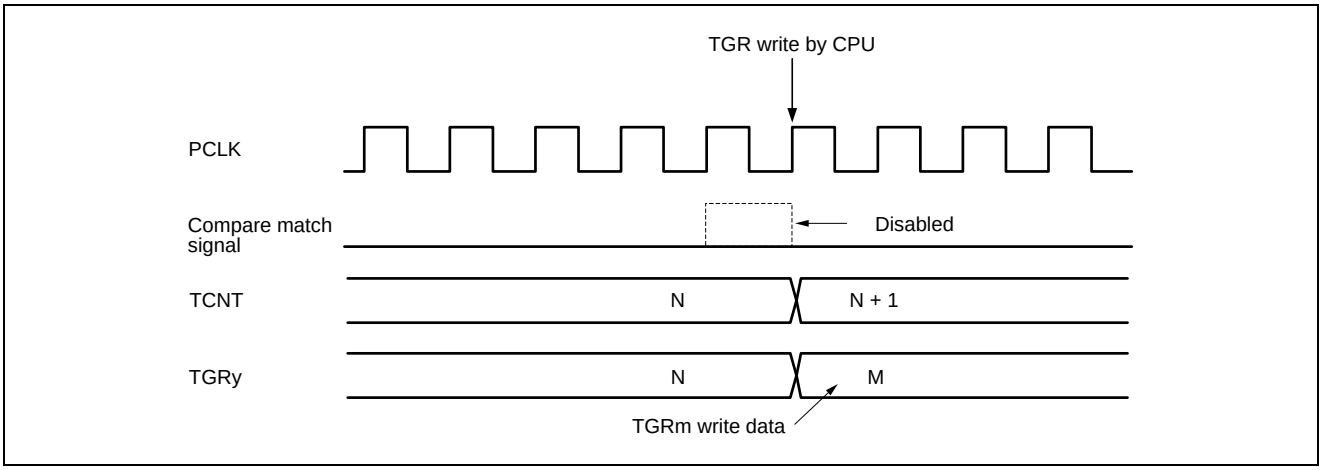



**Figure 15.46 Conflict between TPUm.TGRy Write and Compare Match**

### 15.9.7 Conflict between Buffer Register Write and Compare Match

If a compare match occurs in a TPUm.TGRy write cycle, the data transferred to TGRy by the buffer operation will be the data before writing.

Figure 15.47 shows the timing in this case.

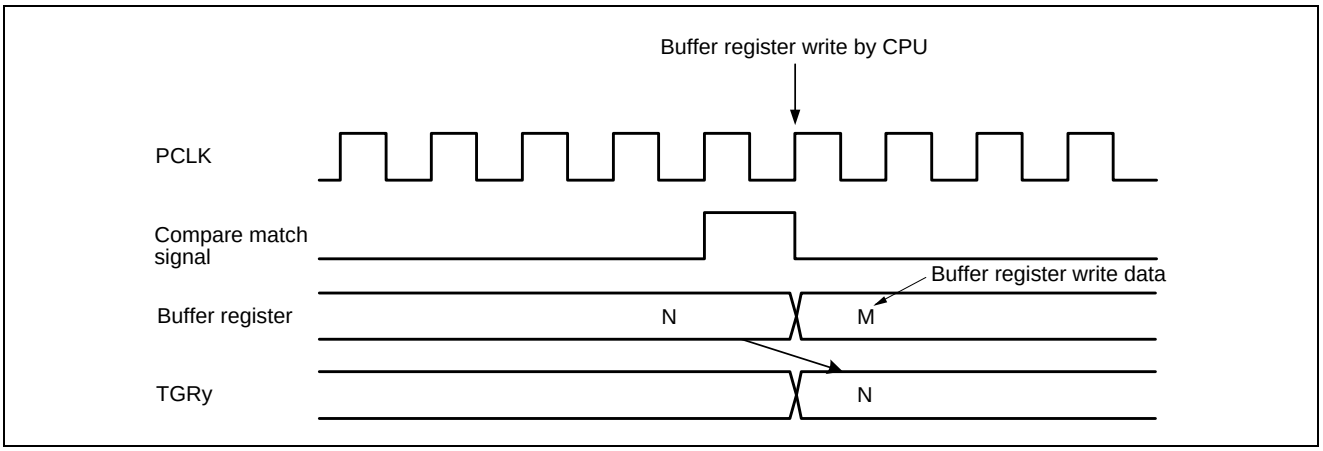



**Figure 15.47 Conflict between Buffer Register Write and Compare Match**



Downloaded from Arrow.com.

### 15.9.8 Conflict between TPUm.TGRy Read and Input Capture

If the input capture signal is generated in a TGRy read cycle, the data that is read will be the data before input capture transfer.

Figure 15.48 shows the timing in this case.

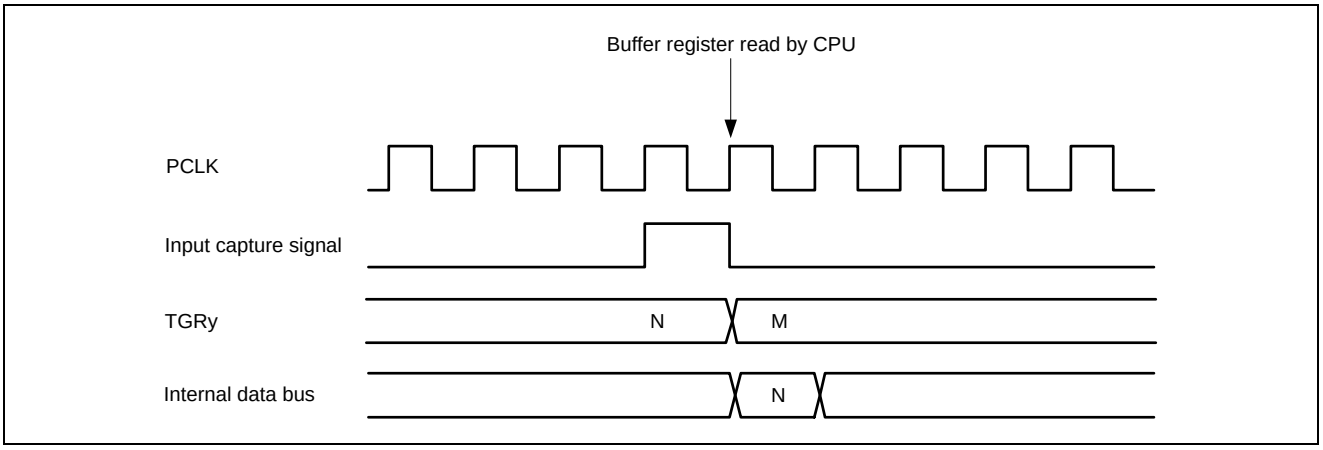



**Figure 15.48 Conflict between TPUm.TGRy Read and Input Capture**

### 15.9.9 Conflict between TPUm.TGRy Write and Input Capture

If the input capture signal is generated in a TGRy write cycle, the input capture operation takes precedence and the write to TGRy is not performed. Figure 15.49 shows the timing in this case.

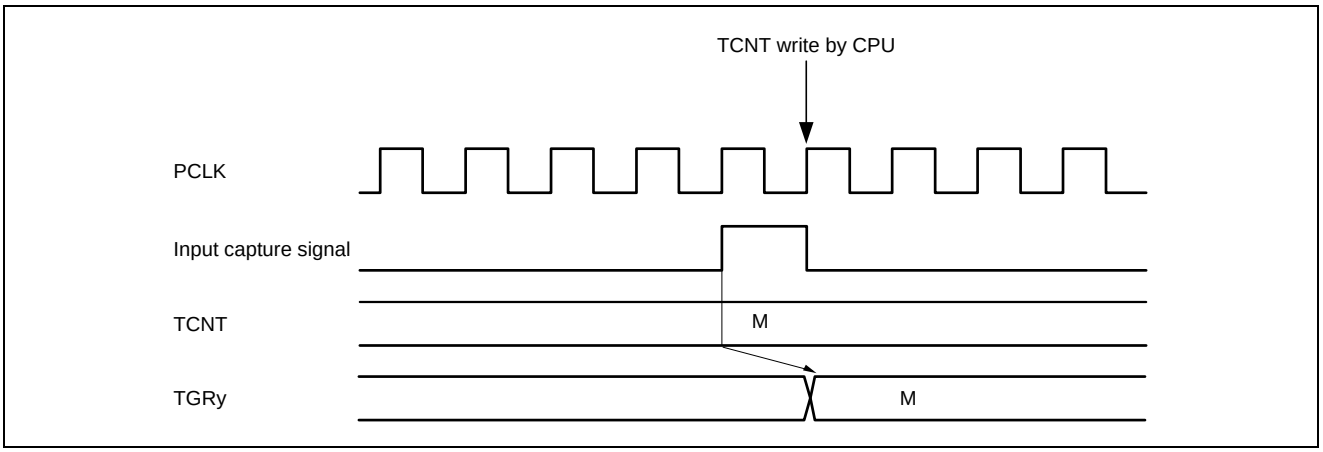



**Figure 15.49 Conflict between TPUm.TGRy Write and Input Capture**



Downloaded from Arrow.com.

### 15.9.10 Conflict between Buffer Register Write and Input Capture

If the input capture signal is generated in a buffer register write cycle, the buffer operation takes precedence and the write to the buffer register is not performed. Figure 15.50 shows the timing in this case.

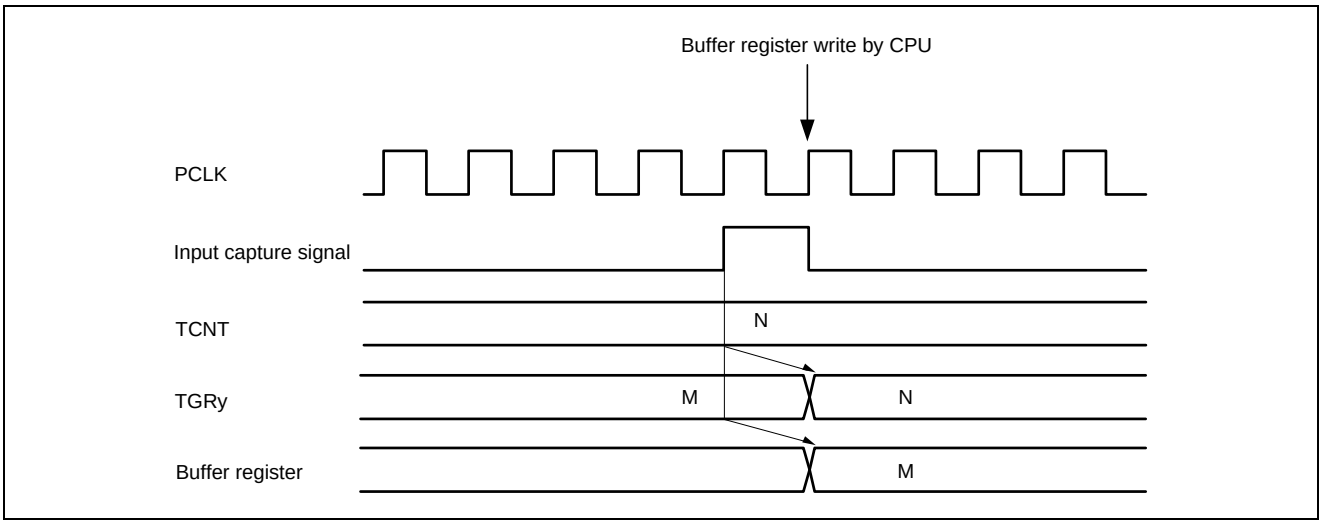



**Figure 15.50 Conflict between Buffer Register Write and Input Capture**

### 15.9.11 Conflict between Overflow/Underflow and Counter Clearing

If overflow/underflow and counter clearing occur simultaneously, TPUm.TCNT counter is cleared with the generation of the compare match interrupt and an overflow interrupt is generated.

Figure 15.51 shows the operation timing when a TPUm.TGRy compare match is specified as the clearing source and FFFFh is set in TGRy.

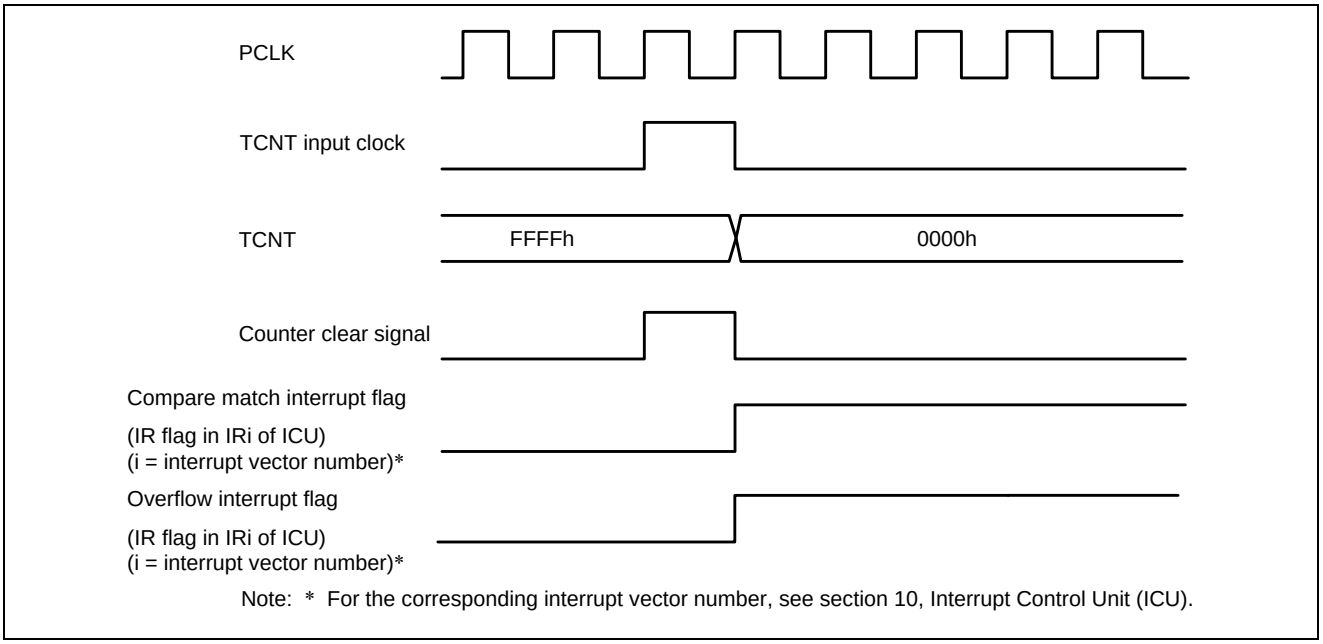



**Figure 15.51 Conflict between Overflow and Counter Clearing**



Downloaded from Arrow.com.

### 15.9.12 Conflict between TPUm.TCNT Write and Overflow/Underflow

If an overflow/underflow occurs due to increment/decrement in a TCNT write cycle, the TCNT write takes precedence.

Figure 15.52 shows the operation timing when there is conflict between TCNT write and overflow.

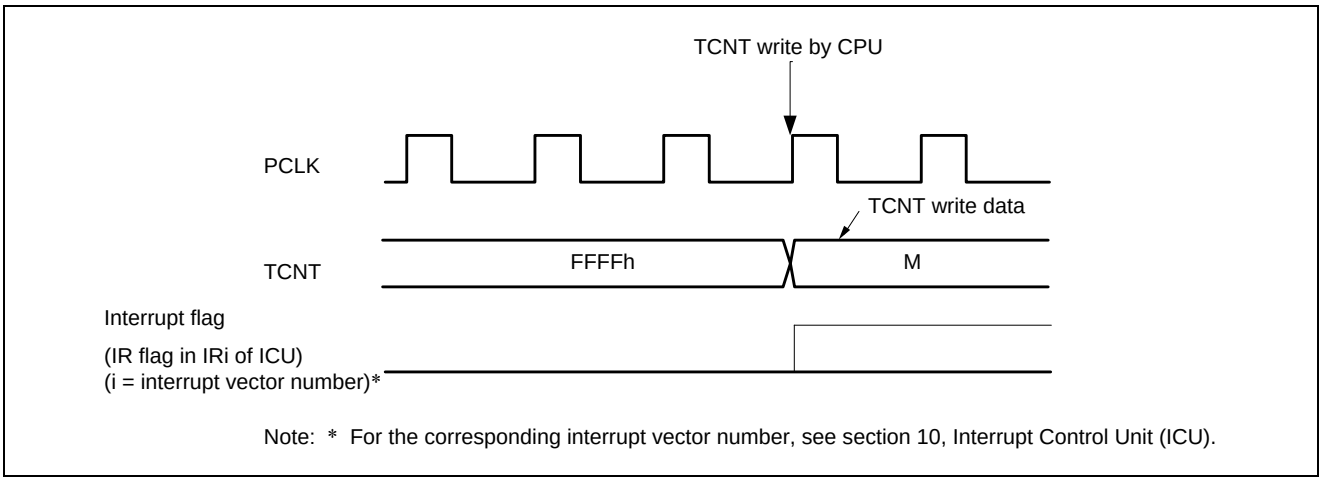



**Figure 15.52 Conflict between TPUm.TCNT Write and Overflow**

### 15.9.13 Multiplexing of I/O Pins

In the RX610 Group, the TCLKA-A input pin is multiplexed with the TIOCC0 I/O pin, the TCLKB-A input pin with the TIOCD0 I/O pin, the TCLKC-A input pin with the TIOCB1 I/O pin, and the TCLKD-A 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.



Downloaded from Arrow.com.

# 16. Programmable Pulse Generator (PPG)

The programmable pulse generator (PPG) generates pulse outputs by using the 16-bit timer pulse unit (TPU) as a time base.

The RX610 Group has two PPG units, each of which controls up to 16 pulse output pins. The pulse outputs from the PPGs are divided into 4-bit groups that can operate all simultaneously and independently.

## 16.1 Overview

Table 16.1 lists the specifications of the PPG and table 16.2 lists PPG functions.

Figures 16.1 and 16.2 show block diagrams of the PPGs.

**Table 16.1 Specifications of PPG**

| Item | Specifications |
|---|---|
| Number of output bits | Up to 32 bits |
| Pulse output | • Two units, each capable of output through four pin groups<br>• Output trigger signals are selectable.<br>• Non-overlapping operation is possible.<br>• Inverted output is selectable. |
| Output data transfer | Can operate together with the DTC and DMAC (When TPU interrupt is in use) |
| Power-down function | Module stop state can be set for each unit. |

**Table 16.2 List of PPG Functions**

| Item | | | PPG0 | PPG1 |
|---|---|---|---|---|
| PPG output trigger | TPU (unit 0) channels 0 to 3 (TPU0 to TPU3) | Compare match | √ | √ |
| | | Input capture | √ | √ |
| | TPU (unit 1) channels 6 to 9 (TPU6 to TPU9) | Compare match | — | √ |
| | | Input capture | — | √ |
| Non-overlapping operation | | | √ | √ |
| Output data transfer | DTC | | √ | √ |
| | DMAC | | √ | √ |
| Selecting inverted output | | | √ | √ |
| Setting the module stop state* | | | The MSTPA11 bit in MSTPCRA | The MSTPA10 bit in MSTPCRA |

[Legend] √: Possible
—: Not possible

Note: * For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

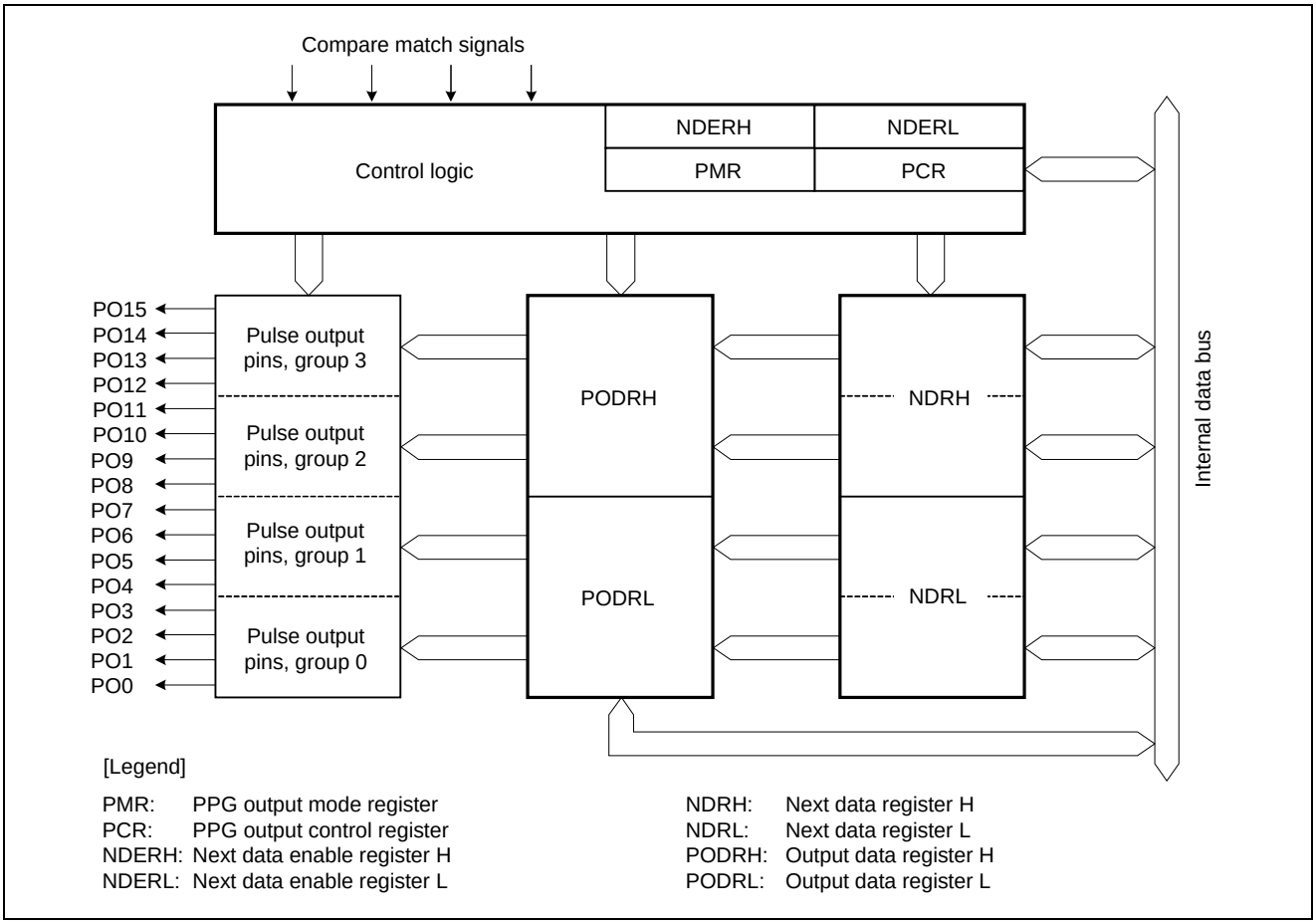



**Figure 16.1 Block Diagram of PPG (Unit 0)**



Downloaded from Arrow.com.

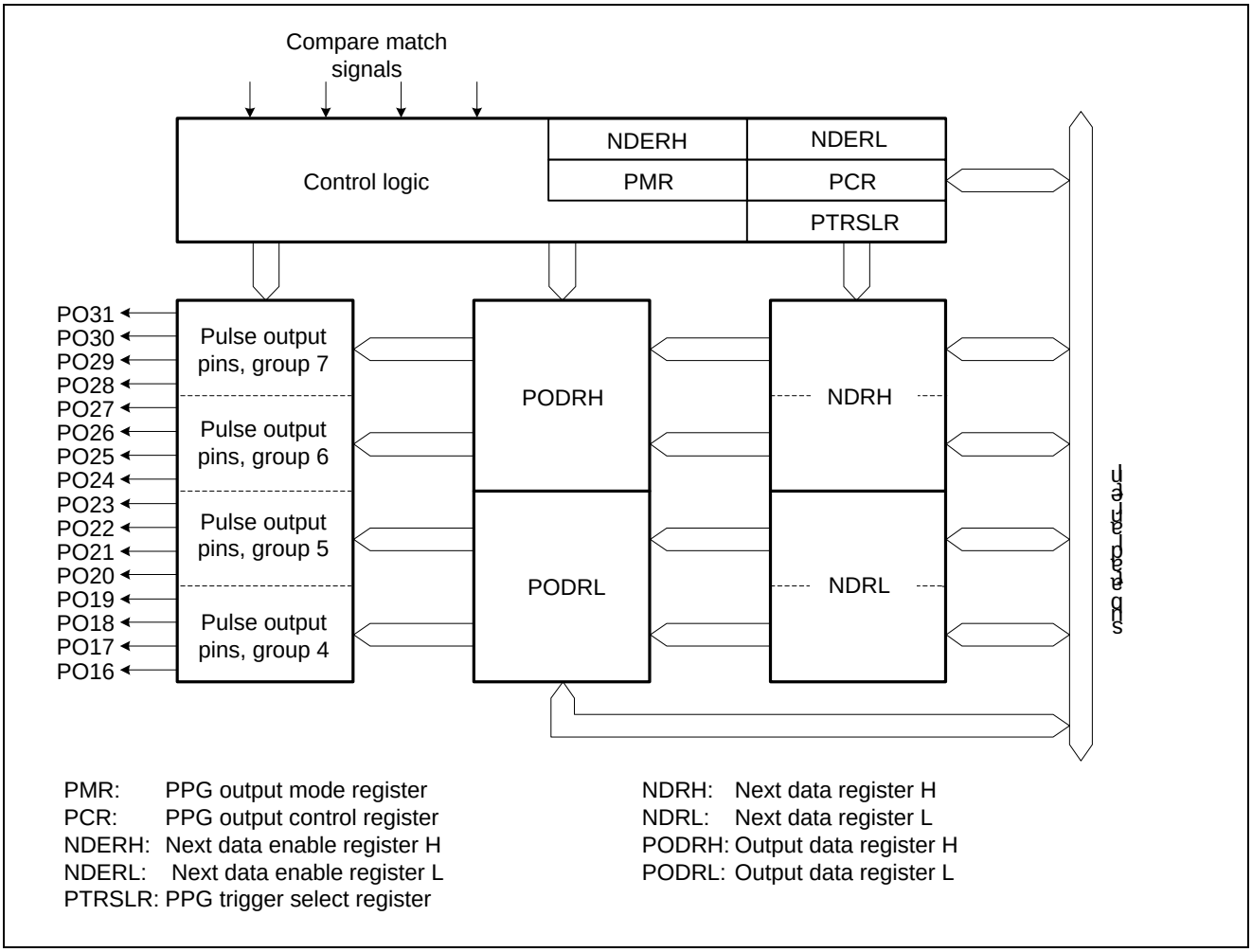



**Figure 16.2 Block Diagram of PPG (Unit 1)**



Downloaded from Arrow.com.

Table 16.3 lists the pin configuration of the PPG.

**Table 16.3 Pin Configuration of PPG**

| Unit | Pin Name | I/O | Function |
|---|---|---|---|
| PPG0 | PO0 | Output | Group 0 pulse output |
| | PO1 | Output | |
| | PO2 | Output | |
| | PO3 | Output | |
| | PO4 | Output | Group 1 pulse output |
| | PO5 | Output | |
| | PO6 | Output | |
| | PO7 | Output | |
| | PO8 | Output | Group 2 pulse output |
| | PO9 | Output | |
| | PO10 | Output | |
| | PO11 | Output | |
| | PO12 | Output | Group 3 pulse output |
| | PO13 | Output | |
| | PO14 | Output | |
| | PO15 | Output | |
| PPG1 | PO16 | Output | Group 4 pulse output |
| | PO17 | Output | |
| | PO18 | Output | |
| | PO19 | Output | |
| | PO20 | Output | Group 5 pulse output |
| | PO21 | Output | |
| | PO22 | Output | |
| | PO23 | Output | |
| | PO24 | Output | Group 6 pulse output |
| | PO25 | Output | |
| | PO26 | Output | |
| | PO27 | Output | |
| | PO28 | Output | Group 7 pulse output |
| | PO29 | Output | |
| | PO30 | Output | |
| | PO31 | Output | |



Downloaded from Arrow.com.

## 16.2 Register Descriptions

Table 16.4 lists the registers of the PPG.

**Table 16.4 Registers of PPG**

| Unit | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| PPG0 | PPG output control register | PCR | FFh | 0008 81E6h | 8 |
| | PPG output mode register | PMR | F0h | 0008 81E7h | 8 |
| | Next data enable register H | NDERH | 00h | 0008 81E8h | 8 |
| | Next data enable register L | NDERL | 00h | 0008 81E9h | 8 |
| | Output data register H | PODRH | 00h | 0008 81EAh | 8 |
| | Output data register L | PODRL | 00h | 0008 81EBh | 8 |
| | Next data register H | NDRH | 00h | 0008 81ECh*1 | 8 |
| | Next data register L | NDRL | 00h | 0008 81EDh*2 | 8 |
| | Next data register H | NDRH | 00h | 0008 81EEh*1 | 8 |
| | Next data register L | NDRL | 00h | 0008 81EFh*2 | 8 |
| PPG1 | PPG trigger select register | PTRSLR | 01h | 0008 81F0h | 8 |
| | PPG output control register | PCR | FFh | 0008 81F6h | 8 |
| | PPG output mode register | PMR | F0h | 0008 81F7h | 8 |
| | Next data enable register H | NDERH | 00h | 0008 81F8h | 8 |
| | Next data enable register L | NDERL | 00h | 0008 81F9h | 8 |
| | Output data register H | PODRH | 00h | 0008 81FAh | 8 |
| | Output data register L | PODRL | 00h | 0008 81FBh | 8 |
| | Next data register H | NDRH | 00h | 0008 81FCh*3 | 8 |
| | Next data register L | NDRL | 00h | 0008 81FDh*4 | 8 |
| | Next data register H | NDRH | 00h | 0008 81FEh*3 | 8 |
| | Next data register L | NDRL | 00h | 0008 81FFh*4 | 8 |

Notes:
1. When pulse output groups 2 and 3 have the same output trigger by PPG0.PCR settings, the PPG0.NDRH address is 0008 81ECh. When they have different output triggers, the PPG0.NDRH addresses corresponding to groups 2 and 3 are 0008 81EEh and 0008 81ECh, respectively.
2. When pulse output groups 0 and 1 have the same output trigger by PPG0.PCR settings, the PPG0.NDRL address is 0008 81EDh. When they have different output triggers, the PPG0.NDRL addresses corresponding to groups 0 and 1 are 0008 81EFh and 0008 81EDh, respectively.
3. When pulse output groups 6 and 7 have the same output trigger by PPG1.PCR settings, the PPG1.NDRH address is 0008 81FCh. When they have different output triggers, the PPG1.NDRH addresses corresponding to groups 6 and 7 are 0008 81FEh and 0008 81FCh, respectively.
4. When pulse output groups 4 and 5 have the same output trigger by PPG1.PCR settings, the PPG1.NDRL address is 0008 81FDh. When they have different output triggers, the PPG1.NDRL addresses corresponding to groups 4 and 5 are 0008 81FFh and 0008 81FDh, respectively.



Downloaded from Arrow.com.

## 16.2.1 PPG Trigger Select Register (PTRSLR)

Address: 0008 81F0h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | PTRSL |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

- PPG1.PTRSLR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | PTRSL | PPG Trigger Select | 0: Selects the set of TPU0 toTPU3 as the trigger channels for PPG1.<br>1: Selects the set of TPU6 to TPU9 as the trigger channels for PPG1. | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

PPG1.PTRSLR selects a set of trigger channels.

**PTRSL Bit (PPG Trigger Select)**

This bit selects either TPU0 to TPU3 or TPU6 to TPU9 as a set of trigger channels for PPG1.

When this bit is set to 0, TPU0 to TPU3 are selected as a set of trigger channels for PPG1.When it is set to 1, TPU6 to TPU9 are selected as a set of trigger channels for PPG1.

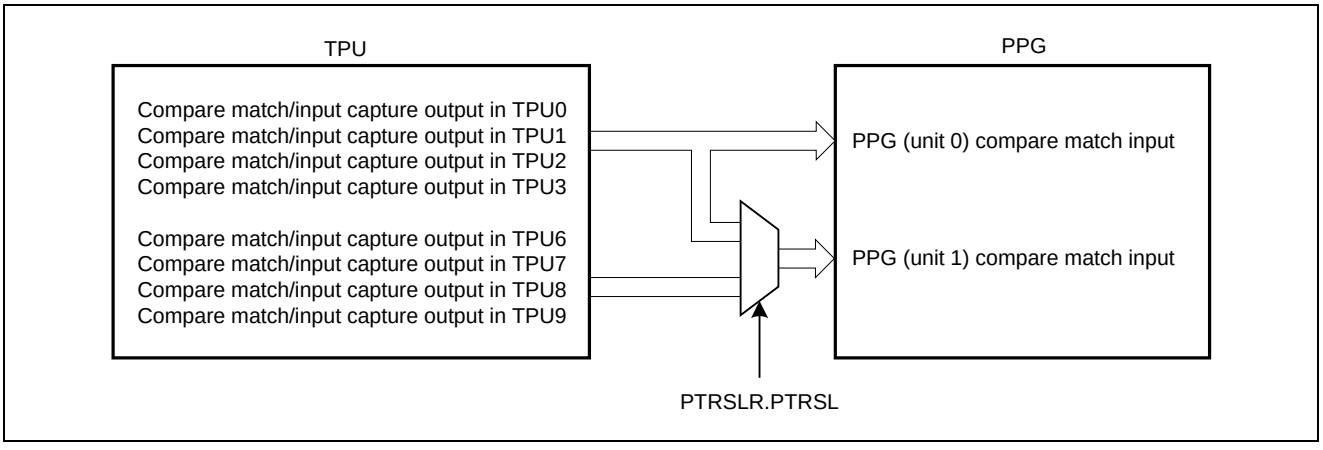



**Figure 16.3 Block Diagram of PPG Trigger Selection**



Downloaded from Arrow.com.

## 16.2.2 Next Data Enable Registers H and L (NDERH, NDERL)

Address: 0008 81E8h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.NDERH | NDER15 | NDER14 | NDER13 | NDER12 | NDER11 | NDER10 | NDER9 | NDER8 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0008 81E9h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.NDERL | NDER7 | NDER6 | NDER5 | NDER4 | NDER3 | NDER2 | NDER1 | NDER0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG0.NDERH

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDER8 | Next Data Transfer Enable | 0: Data transfer is disabled. | R/W |
| b1 | NDER9 | Next Data Transfer Enable | 1: Data transfer is enabled. | R/W |
| b2 | NDER10 | Next Data Transfer Enable | | R/W |
| b3 | NDER11 | Next Data Transfer Enable | | R/W |
| b4 | NDER12 | Next Data Transfer Enable | | R/W |
| b5 | NDER13 | Next Data Transfer Enable | | R/W |
| b6 | NDER14 | Next Data Transfer Enable | | R/W |
| b7 | NDER15 | Next Data Transfer Enable | | R/W |

PPG0.NDERH selects the pins (PO8 to PO15) for outputs of pulse from the PPG on a bit-by-bit basis.

**NDERj Bits (Next Data Transfer Enable) (j = 8 to 15)**

When these bits are set to 1, the output trigger specified by PTRSLR transfers data from the corresponding bit in PPG0.NDRH to the bit in PPG0.PODRH. When these bits are set to 0, the data are not transferred from PPG0.NDRH to PPG0.PODRH.



Downloaded from Arrow.com.

- PPG0.NDERL

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDER0 | Next Data Transfer Enable | 0: Data transfer is disabled. | R/W |
| b1 | NDER1 | Next Data Transfer Enable | 1: Data transfer is enabled. | R/W |
| b2 | NDER2 | Next Data Transfer Enable | | R/W |
| b3 | NDER3 | Next Data Transfer Enable | | R/W |
| b4 | NDER4 | Next Data Transfer Enable | | R/W |
| b5 | NDER5 | Next Data Transfer Enable | | R/W |
| b6 | NDER6 | Next Data Transfer Enable | | R/W |
| b7 | NDER7 | Next Data Transfer Enable | | R/W |

PPG0.NDERL selects the pins (PO0 to PO7) for outputs of pulse from the PPG on a bit-by-bit basis.

**NDERj Bits (Next Data Transfer Enable) (j = 0 to 7)**

When these bits are set to 1, the output trigger specified by PTRSLR transfers data from the corresponding bit in PPG0.NDRL to the bit in PPG0.PODRL. When these bits are set to 0, the data are not transferred from PPG0.NDRL to PPG0.PODRL.



Downloaded from Arrow.com.

Address: 0008 81F8h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.NDERH | NDER31 | NDER30 | NDER29 | NDER28 | NDER27 | NDER26 | NDER25 | NDER24 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0008 81F9h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.NDERL | NDER23 | NDER22 | NDER21 | NDER20 | NDER19 | NDER18 | NDER17 | NDER16 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG1.NDERH

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDER24 | Next Data Transfer Enable | 0: Data transfer is disabled. | R/W |
| b1 | NDER25 | Next Data Transfer Enable | 1: Data transfer is enabled. | R/W |
| b2 | NDER26 | Next Data Transfer Enable | | R/W |
| b3 | NDER27 | Next Data Transfer Enable | | R/W |
| b4 | NDER28 | Next Data Transfer Enable | | R/W |
| b5 | NDER29 | Next Data Transfer Enable | | R/W |
| b6 | NDER30 | Next Data Transfer Enable | | R/W |
| b7 | NDER31 | Next Data Transfer Enable | | R/W |

PPG1.NDERH selects the pins (PO24 to PO31) for outputs of pulse from the PPG on a bit-by-bit basis.

**NDERj Bits (Next Data Transfer Enable) (j = 24 to 31)**

When these bits are set to 1, the output trigger specified by PTRSLR transfers data from the corresponding bit in PPG1.NDRH to the bit in PPG1.PODRH. When these bits are set to 0, the data are not transferred from PPG1.NDRH to PPG1.PODRH.



Downloaded from Arrow.com.

- PPG1.NDERL

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDER16 | Next Data Transfer Enable | 0: Data transfer is disabled. | R/W |
| b1 | NDER17 | Next Data Transfer Enable | 1: Data transfer is enabled. | R/W |
| b2 | NDER18 | Next Data Transfer Enable | | R/W |
| b3 | NDER19 | Next Data Transfer Enable | | R/W |
| b4 | NDER20 | Next Data Transfer Enable | | R/W |
| b5 | NDER21 | Next Data Transfer Enable | | R/W |
| b6 | NDER22 | Next Data Transfer Enable | | R/W |
| b7 | NDER23 | Next Data Transfer Enable | | R/W |

PPG1.NDERL selects the pins (PO16 to PO23) for outputs of pulse from the PPG on a bit-by-bit basis.

**NDERj Bits (Next Data Transfer Enable) (j = 16 to 23)**

When these bits are set to 1, the output trigger specified by PTRSLR transfers data from the corresponding bit in PPG1.NDRL to the bit in PPG1.PODRL. When these bits are set to 0, the data are not transferred from PPG1.NDRL to PPG1.PODRL.



Downloaded from Arrow.com.

### 16.2.3 Output Data Registers H and L (PODRH, PODRL)

Address: 0008 81EAh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.PODRH | POD15 | POD14 | POD13 | POD12 | POD11 | POD10 | POD9 | POD8 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0008 81EBh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.PODRL | POD7 | POD6 | POD5 | POD4 | POD3 | POD2 | POD1 | POD0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG0.PODRH

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | POD8 | Output Data Register | For bits corresponding to pins that have been set for pulse output by PPG0.NDERH, the output trigger transfers the values in PPG0.NDRH to this register during PPG operation. If any of the NDERj (j = 8 to 15) bits in PPG0.NDERH is set to 1, the CPU cannot write to this register. Initial output values for the pins can be set while PPG0.NDERH is clear (00h). | R/W |
| b1 | POD9 | Output Data Register | | R/W |
| b2 | POD10 | Output Data Register | | R/W |
| b3 | POD11 | Output Data Register | | R/W |
| b4 | POD12 | Output Data Register | | R/W |
| b5 | POD13 | Output Data Register | | R/W |
| b6 | POD14 | Output Data Register | | R/W |
| b7 | POD15 | Output Data Register | | R/W |

PPG0.PODRH stores pulse output values. For bits corresponding to pins that have been set for pulse output by PPG0.NDERH, the output trigger transfers the values in PPG0.NDRH to this register.

- PPG0.PODRL

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | POD0 | Output Data Register | For bits corresponding to pins that have been set for pulse output by PPG0.NDERL, the output trigger transfers the values in PPG0.NDRL to this register during PPG operation. If any of the NDERj (j = 0 to 7) bits in PPG0.NDERL is set to 1, the CPU cannot write to this register. Initial output values for the pins can be set while PPG0.NDERL is clear (00h). | R/W |
| b1 | POD1 | Output Data Register | | R/W |
| b2 | POD2 | Output Data Register | | R/W |
| b3 | POD3 | Output Data Register | | R/W |
| b4 | POD4 | Output Data Register | | R/W |
| b5 | POD5 | Output Data Register | | R/W |
| b6 | POD6 | Output Data Register | | R/W |
| b7 | POD7 | Output Data Register | | R/W |

PPG0.PODRL stores pulse output values. For bits corresponding to pins that have been set for pulse output by PPG0.NDERL, the output trigger transfers the values in PPG0.NDRL to this register.



Downloaded from Arrow.com.

Address: 0008 81FAh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.PODRH | POD31 | POD30 | POD29 | POD28 | POD27 | POD26 | POD25 | POD24 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0008 81FBh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.PODRL | POD23 | POD22 | POD21 | POD20 | POD19 | POD18 | POD17 | POD16 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG1.PODRH

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | POD24 | Output Data Register | For bits corresponding to pins that have been set for pulse output by PPG1.NDERH, the output trigger transfers the values in PPG1.NDRH to this register during PPG operation. If any of the NDERj (j = 24 to 31) bits in PPG1.NDERH is set to 1, the CPU cannot write to this register. Initial output values for the pins can be set while PPG1.NDERH is clear (00h). | R/W |
| b1 | POD25 | Output Data Register | | R/W |
| b2 | POD26 | Output Data Register | | R/W |
| b3 | POD27 | Output Data Register | | R/W |
| b4 | POD28 | Output Data Register | | R/W |
| b5 | POD29 | Output Data Register | | R/W |
| b6 | POD30 | Output Data Register | | R/W |
| b7 | POD31 | Output Data Register | | R/W |

PPG1.PODRH stores pulse output values. For bits corresponding to pins that have been set for pulse output by PPG1.NDERH, the output trigger transfers the values in PPG1.NDRH to this register.

- PPG1.PODRL

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | POD16 | Output Data Register | For bits corresponding to pins that have been set for pulse output by PPG1.NDERL, the output trigger transfers the values in PPG1.NDRL to this register during PPG operation. If any of the NDERj (j = 16 to 23) bits in PPG1.NDERL is set to 1, the CPU cannot write to this register. Initial output values for the pins can be set while PPG1.NDERL is clear (00h). | R/W |
| b1 | POD17 | Output Data Register | | R/W |
| b2 | POD18 | Output Data Register | | R/W |
| b3 | POD19 | Output Data Register | | R/W |
| b4 | POD20 | Output Data Register | | R/W |
| b5 | POD21 | Output Data Register | | R/W |
| b6 | POD22 | Output Data Register | | R/W |
| b7 | POD23 | Output Data Register | | R/W |

PPG1.PODRL stores pulse output values. For bits corresponding to pins that have been set for pulse output by PPG1.NDERL, the output trigger transfers the values in PPG1.NDRL to this register.



Downloaded from Arrow.com.

## 16.2.4 Next Data Registers H and L (NDRH, NDRL)

Addresses: 0008 81ECh, 0008 81EEh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.NDRH | NDR15 | NDR14 | NDR13 | NDR12 | NDR11 | NDR10 | NDR9 | NDR8 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Addresses: 0008 81EDh, 0008 81EFh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG0.NDRL | NDR7 | NDR6 | NDR5 | NDR4 | NDR3 | NDR2 | NDR1 | NDR0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG0.NDRH

PPG0.NDRH stores the next data for pulse output. The PPG0.NDRH address differs depending on whether pulse output groups have the same output trigger or different output triggers.

(1) When pulse output groups 2 and 3 have the same output trigger

If pulse output groups 2 and 3 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time.

(0008 81ECh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR8 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRH. | R/W |
| b1 | NDR9 | Next Data Register | | R/W |
| b2 | NDR10 | Next Data Register | | R/W |
| b3 | NDR11 | Next Data Register | | R/W |
| b4 | NDR12 | Next Data Register | | R/W |
| b5 | NDR13 | Next Data Register | | R/W |
| b6 | NDR14 | Next Data Register | | R/W |
| b7 | NDR15 | Next Data Register | | R/W |

(2) When pulse output groups 2 and 3 have different output triggers

If pulse output groups 2 and 3 have different output triggers, upper 4 bits and lower 4 bits are mapped to the different addresses.

(Pulse output group 3: 0008 81ECh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b4 | NDR12 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRH. | R/W |
| b5 | NDR13 | Next Data Register | | R/W |
| b6 | NDR14 | Next Data Register | | R/W |
| b7 | NDR15 | Next Data Register | | R/W |



Downloaded from Arrow.com.

(Pulse output group 2: 0008 81EEh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR8 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRH. | R/W |
| b1 | NDR9 | Next Data Register | | R/W |
| b2 | NDR10 | Next Data Register | | R/W |
| b3 | NDR11 | Next Data Register | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 1 and cannot be modified. | R |

- PPG0.NDRL

PPG0.NDRL stores the next data for pulse output. The PPG0.NDRL address differs depending on whether pulse output groups have the same output trigger or different output triggers.

(1) When pulse output groups 0 and 1 have the same output trigger

If pulse output groups 0 and 1 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time.

(0008 81EDh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR0 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRL. | R/W |
| b1 | NDR1 | Next Data Register | | R/W |
| b2 | NDR2 | Next Data Register | | R/W |
| b3 | NDR3 | Next Data Register | | R/W |
| b4 | NDR4 | Next Data Register | | R/W |
| b5 | NDR5 | Next Data Register | | R/W |
| b6 | NDR6 | Next Data Register | | R/W |
| b7 | NDR7 | Next Data Register | | R/W |

(2) When pulse output groups 0 and 1 have different output triggers

If pulse output groups 0 and 1 have different output triggers, upper 4 bits and lower 4 bits are mapped to the different addresses.

(Pulse output group 1: 0008 81EDh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b4 | NDR4 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRL. | R/W |
| b5 | NDR5 | Next Data Register | | R/W |
| b6 | NDR6 | Next Data Register | | R/W |
| b7 | NDR7 | Next Data Register | | R/W |



Downloaded from Arrow.com.

(Pulse output group 0: 0008 81EFh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR0 | Next Data Register | The output trigger specified by PPG0.PCR transfers the values in this register to the corresponding bits in PPG0.PODRL. | R/W |
| b1 | NDR1 | Next Data Register | | R/W |
| b2 | NDR2 | Next Data Register | | R/W |
| b3 | NDR3 | Next Data Register | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |

Addresses: 0008 81FCh, 0008 81FEh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.NDRH | NDR31 | NDR30 | NDR29 | NDR28 | NDR27 | NDR26 | NDR25 | NDR24 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Addresses: 0008 81FDh, 0008 81FFh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| PPG1.NDRL | NDR23 | NDR22 | NDR21 | NDR20 | NDR19 | NDR18 | NDR17 | NDR16 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- PPG1.NDRH

PPG1.NDRH stores the next data for pulse output. The PPG1.NDRH address differs depending on whether pulse output groups have the same output trigger or different output triggers.

(1) When pulse output groups 6 and 7 have the same output trigger

If pulse output groups 6 and 7 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time.

(0008 81FCh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR24 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRH. | R/W |
| b1 | NDR25 | Next Data Register | | R/W |
| b2 | NDR26 | Next Data Register | | R/W |
| b3 | NDR27 | Next Data Register | | R/W |
| b4 | NDR28 | Next Data Register | | R/W |
| b5 | NDR29 | Next Data Register | | R/W |
| b6 | NDR30 | Next Data Register | | R/W |
| b7 | NDR31 | Next Data Register | | R/W |



Downloaded from Arrow.com.

(2) When pulse output groups 6 and 7 have different output triggers

If pulse output groups 6 and 7 have different output triggers, upper 4 bits and lower 4 bits are mapped to the different addresses.

(Pulse output group 7: 0008 81FCh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b4 | NDR28 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRH. | R/W |
| b5 | NDR29 | Next Data Register | | R/W |
| b6 | NDR30 | Next Data Register | | R/W |
| b7 | NDR31 | Next Data Register | | R/W |

(Pulse output group 6: 0008 81FEh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR24 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRH. | R/W |
| b1 | NDR25 | Next Data Register | | R/W |
| b2 | NDR26 | Next Data Register | | R/W |
| b3 | NDR27 | Next Data Register | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |

- PPG1.NDRL

PPG1.NDRL stores the next data for pulse output. The PPG1.NDRL address differs depending on whether pulse output groups have the same output trigger or different output triggers.

(1) When pulse output groups 4 and 5 have the same output trigger

If pulse output groups 4 and 5 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time.

(0008 81FDh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR16 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRL. | R/W |
| b1 | NDR17 | Next Data Register | | R/W |
| b2 | NDR18 | Next Data Register | | R/W |
| b3 | NDR19 | Next Data Register | | R/W |
| b4 | NDR20 | Next Data Register | | R/W |
| b5 | NDR21 | Next Data Register | | R/W |
| b6 | NDR22 | Next Data Register | | R/W |
| b7 | NDR23 | Next Data Register | | R/W |



Downloaded from Arrow.com.

(2) When pulse output groups 4 and 5 have different output triggers

If pulse output groups 4 and 5 have different output triggers, upper 4 bits and lower 4 bits are mapped to the different addresses.

(Pulse output group 5: 0008 81FDh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b4 | NDR20 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRL. | R/W |
| b5 | NDR21 | Next Data Register | | R/W |
| b6 | NDR22 | Next Data Register | | R/W |
| b7 | NDR23 | Next Data Register | | R/W |

(Pulse output group 4: 0008 81FFh)

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | NDR16 | Next Data Register | The output trigger specified by PPG1.PCR transfers the values in this register to the corresponding bits in PPG1.PODRL. | R/W |
| b1 | NDR17 | Next Data Register | | R/W |
| b2 | NDR18 | Next Data Register | | R/W |
| b3 | NDR19 | Next Data Register | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |



Downloaded from Arrow.com.

### 16.2.5 PPG Output Control Register (PCR)

Addresses: PPG0.PCR 0008 81E6h, PPG1.PCR 0008 81F6h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | G3CMS[1:0] | | G2CMS[1:0] | | G1CMS[1:0] | | G0CMS[1:0] | |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- PPG0.PCR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | G0CMS[1:0] | Group 0 Compare Match Select | b1 b0<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3 | R/W |
| b3, b2 | G1CMS[1:0] | Group 1 Compare Match Select | b3 b2<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3 | R/W |
| b5, b4 | G2CMS[1:0] | Group 2 Compare Match Select | b5 b4<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3 | R/W |
| b7, b6 | G3CMS[1:0] | Group 3 Compare Match Select | b7 b6<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3 | R/W |

- PPG1.PCR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | G0CMS[1:0] | Group 4 Compare Match Select | • When the PTRSL bit in PPG1.PTRSLR is set to 0.<br>b1 b0<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3<br>• When the PTRSL bit in PPG1.PTRSLR is set to 1.<br>b1 b0<br>0 0: Compare match in TPU6<br>0 1: Compare match in TPU7<br>1 0: Compare match in TPU8<br>1 1: Compare match in TPU9 | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3, b2 | G1CMS[1:0] | Group 5 Compare Match Select | • When the PTRSL bit in PPG1.PTRSLR is set to 0.<br>b3 b2<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3<br>• When the PTRSL bit in PPG1.PTRSLR is set to 1.<br>b3 b2<br>0 0: Compare match in TPU6<br>0 1: Compare match in TPU7<br>1 0: Compare match in TPU8<br>1 1: Compare match in TPU9 | R/W |
| b5, b4 | G2CMS[1:0] | Group 6 Compare Match Select | • When the PTRSL bit in PPG1.PTRSLR is set to 0.<br>b5 b4<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3<br>• When the PTRSL bit in PPG1.PTRSLR is set to 1.<br>b5 b4<br>0 0: Compare match in TPU6<br>0 1: Compare match in TPU7<br>1 0: Compare match in TPU8<br>1 1: Compare match in TPU9 | R/W |
| b7, b6 | G3CMS[1:0] | Group 7 Compare Match Select | • When the PTRSL bit in PPG1.PTRSLR is set to 0.<br>b7 b6<br>0 0: Compare match in TPU0<br>0 1: Compare match in TPU1<br>1 0: Compare match in TPU2<br>1 1: Compare match in TPU3<br>• When the PTRSL bit in PPG1.PTRSLR is set to 1.<br>b7 b6<br>0 0: Compare match in TPU6<br>0 1: Compare match in TPU7<br>1 0: Compare match in TPU8<br>1 1: Compare match in TPU9 | R/W |

PPGm.PCR (m = 0, 1) selects pulse output trigger signals on a group-by-group basis. For details on output trigger selection, see section 16.2.6, PPG Output Mode Register (PMR).

**GjCMS[1:0] Bits (Group k Compare Match Select) (j = 0 to 3, k = 0 to 7)**

Each bit selects an output trigger of pulse output group k.



Downloaded from Arrow.com.

### 16.2.6 PPG Output Mode Register (PMR)

Addresses: PPG0.PMR 0008 81E7h, PPG1.PMR 0008 81F7h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | G3INV | G2INV | G1INV | G0INV | G3NOV | G2NOV | G1NOV | G0NOV |
| Value after reset: | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

- PPG0.PMR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | G0NOV | Group 0 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3) | R/W |
| b1 | G1NOV | Group 1 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3) | R/W |
| b2 | G2NOV | Group 2 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3) | R/W |
| b3 | G3NOV | Group 3 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3) | R/W |
| b4 | G0INV | Group 0 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b5 | G1INV | Group 1 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b6 | G2INV | Group 2 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b7 | G3INV | Group 3 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |



Downloaded from Arrow.com.

- PPG1.PMR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | G0NOV | Group 4 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3, 6 to 9) | R/W |
| b1 | G1NOV | Group 5 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3, 6 to 9) | R/W |
| b2 | G2NOV | Group 6 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3, 6 to 9) | R/W |
| b3 | G3NOV | Group 7 Non-Overlap | 0: Normal operation<br>(Output values updated on compare match A in the selected TPUm)<br>1: Non-overlapping operation<br>(Output values updated on compare match A or B in the selected TPUm)<br>(m = 0 to 3, 6 to 9) | R/W |
| b4 | G0INV | Group 4 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b5 | G1INV | Group 5 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b6 | G2INV | Group 6 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |
| b7 | G3INV | Group 7 Output Polarity Invert | 0: Inverted output<br>1: Direct output | R/W |

PPGm.PCR (m = 0, 1) selects the pulse output mode of the PPG on a group-by-group basis.

While inverted output is selected, a low-level pulse is output when the values in PPGm.PODRH and PPGm.PODRL are 1, and a high-level pulse is output when the values in PPGm.PODRH and PPGm.PODRL are 0.

In addition, when non-overlapping operation is selected, the PPG updates its output values on compare match A or B in a TPU channel that functions as an output trigger.

For details, see section 16.3.4, Non-Overlapping Pulse Output.



Downloaded from Arrow.com.

### GjNOV Bits (Group k Non-Overlap) (j = 0 to 3, k = 0 to 7)

Each bit selects normal operation or non-overlapping operation for pulse output group j.

### GjINV Bits (Group k Invert) (j = 0 to 3, k = 0 to 7)

Each bit selects direct output or inverted output for pulse output group k.



Downloaded from Arrow.com.

## 16.3 Operation

Figure 16.4 shows a schematic diagram of the PPG.

PPG pulse output is enabled when the corresponding bits in PPGm.NDERH and PPGm.NDERL (m = 0, 1) are set to 1 (data transfer is enabled).

An initial output value is determined by the initial settings in the corresponding PPGm.PODRH and PPGm.PODRL. When the compare match event selected in PPGm.PCR occurs, the output values are updated by transfer of the values in the corresponding PPGm.NDRH and PPGm.NDRL to PPGm.PODRH and PPGm.PODRL, respectively.

Consecutive output of up to 16 bits of data is possible by writing new output data to PPGm.NDRH and PPGm.NDRL before the next compare match.

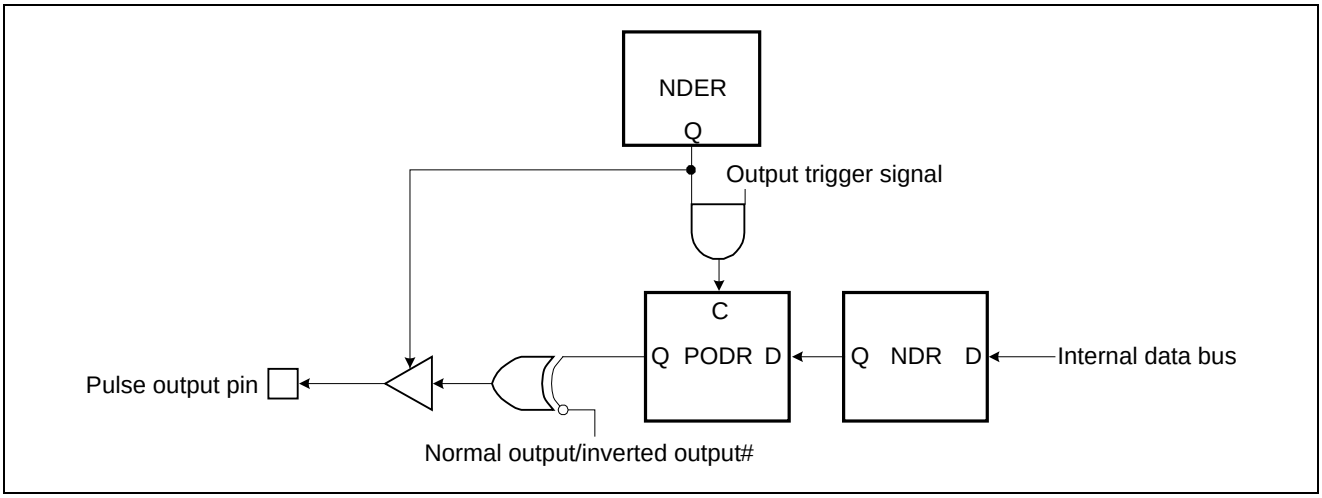



**Figure 16.4 Schematic Diagram of PPG**



Downloaded from Arrow.com.

### 16.3.1 Output Timing

When the selected compare match event occurs while pulse output is enabled, the values in PPGm.NDRH and PPGm.NDRL (m = 0, 1) are transferred to PPGm.PODRH and PPGm.PODRL, respectively, and then output on the corresponding pins.

Figure 16.5 shows the timing of the above operation. In this case, the timing when compare match A triggers normal output from groups 2 and 3 is shown.

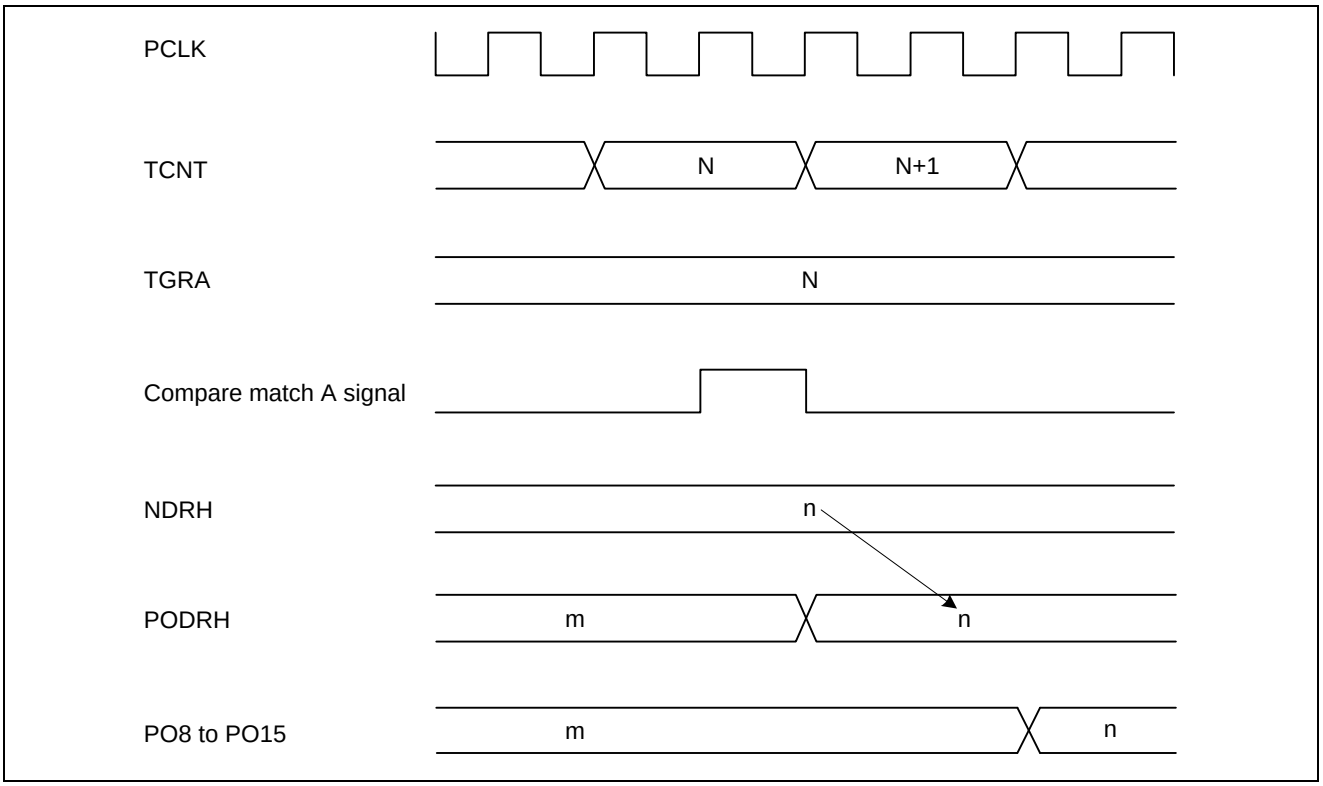



**Figure 16.5 Timing of Transfer and Output of the Values in NDR (Example)**



Downloaded from Arrow.com.

## 16.3.2 Sample Setup Procedure for Normal Pulse Output

Figures 16.6 and 16.7 show sample procedures for setting normal pulse output.

### (1) PPG0 Setting

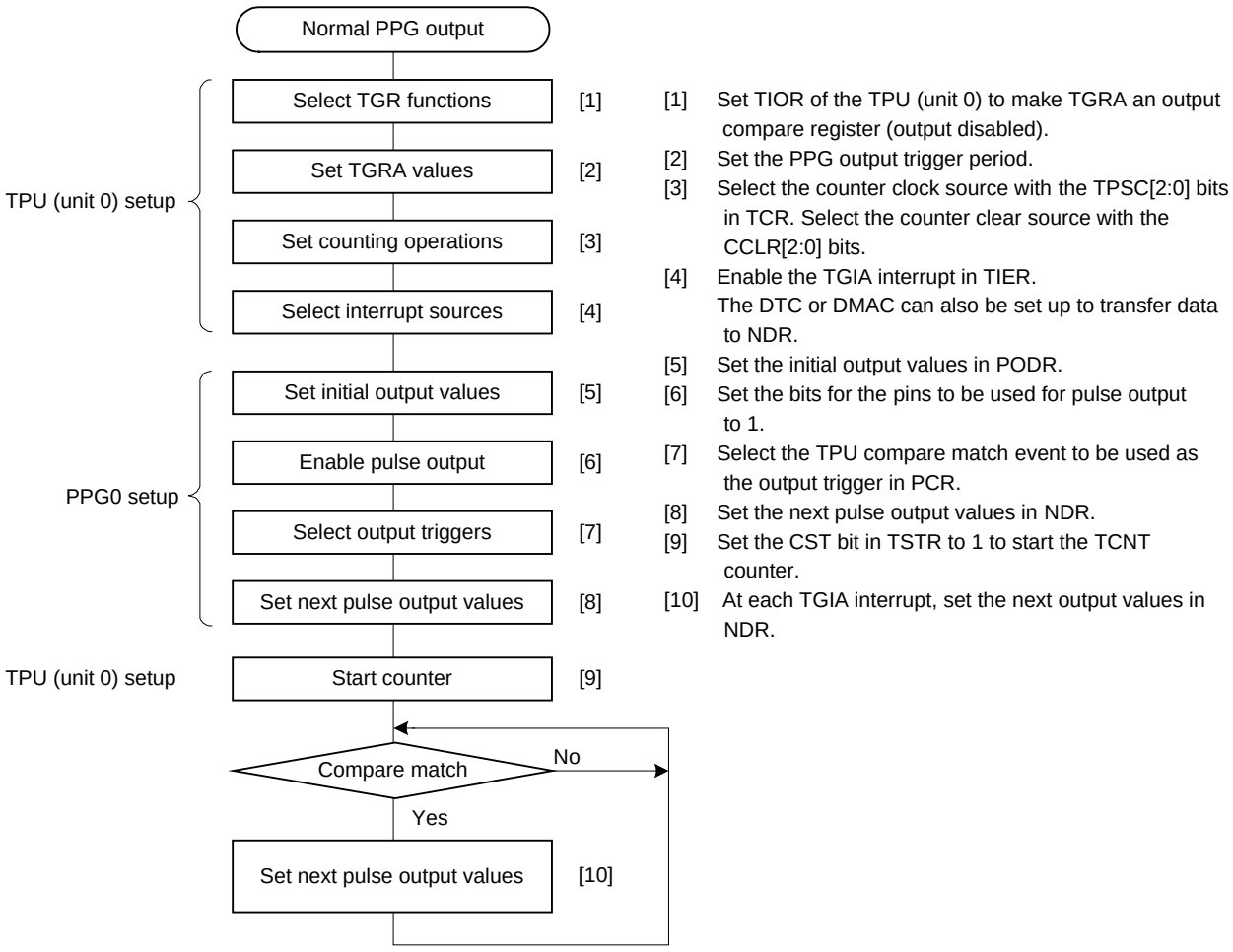



**Figure 16.6 Sample Setup Procedure for Normal Pulse Output (PPG0 Setting)**



Downloaded from Arrow.com.

## (2) PPG1 Setting

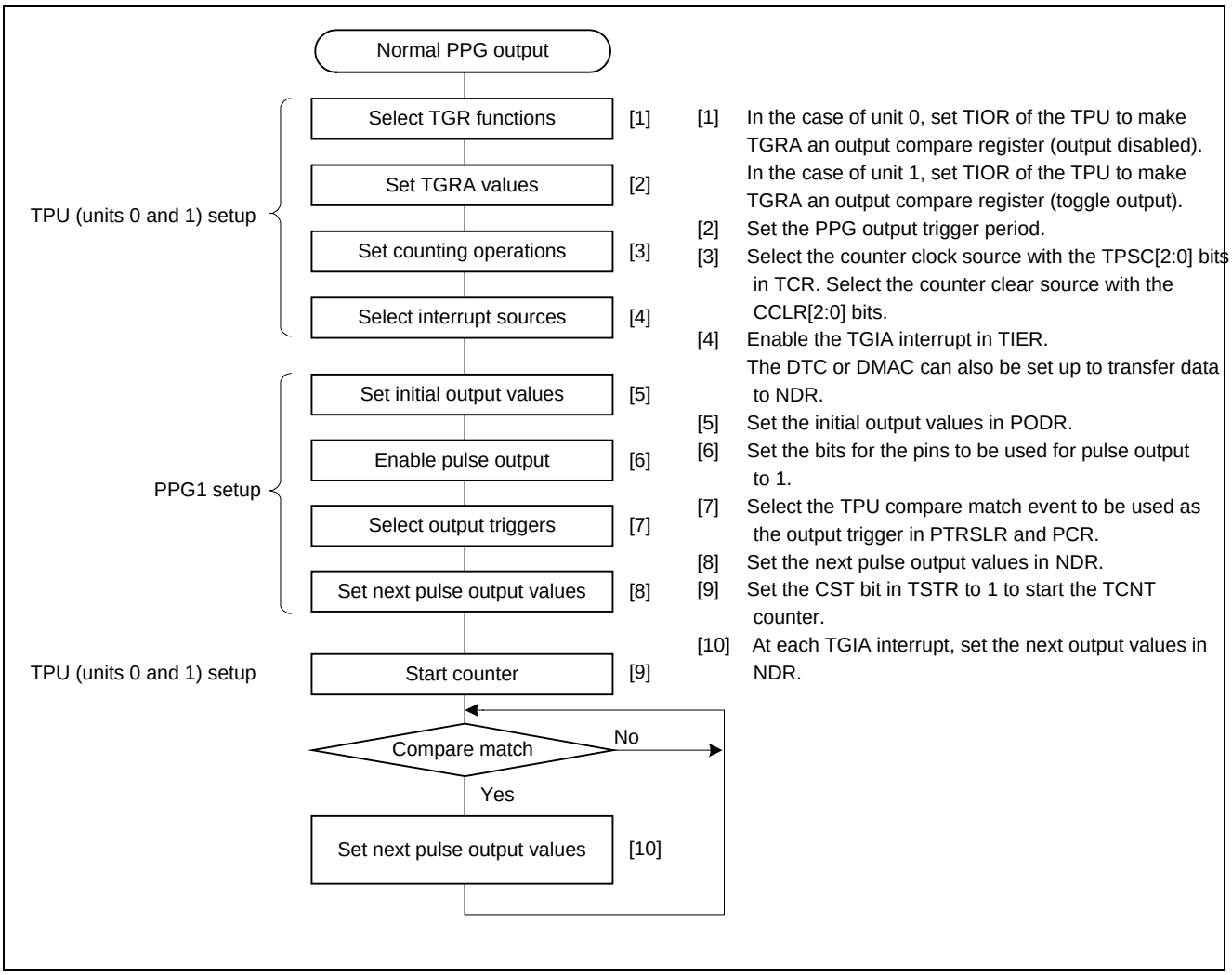



**Figure 16.7 Sample Setup Procedure for Normal Pulse Output (PPG1 Setting)**



Downloaded from Arrow.com.

### 16.3.3 Example of Normal Pulse Output (Example of Five-Phase Pulse Output)

Figure 16.8 shows an example in which pulse output from the PPG0 is used for cyclic five-phase pulse output.

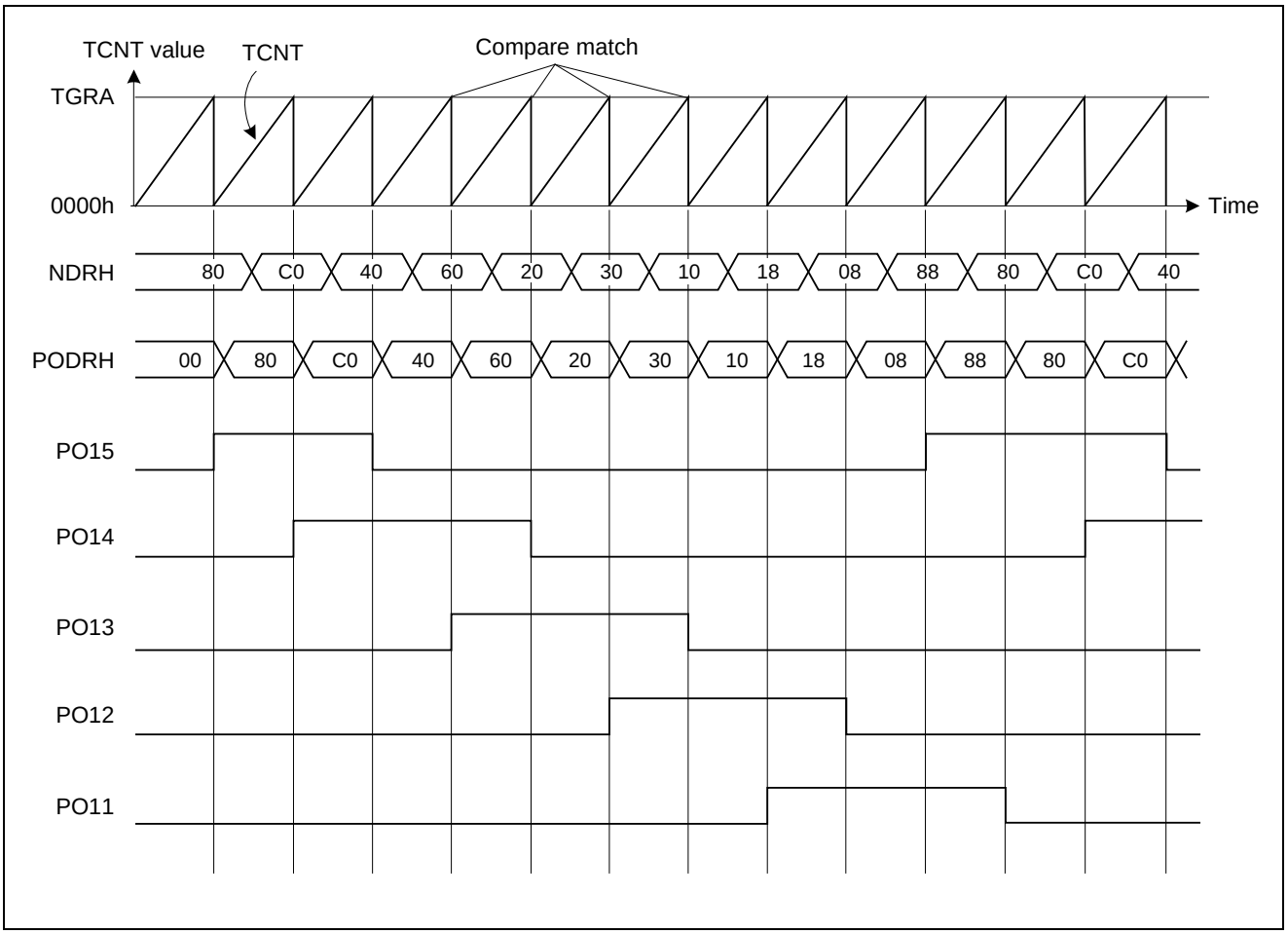



**Figure 16.8 Example of Normal Pulse Output (Example of Five-Phase Pulse Output)**

1. Set an output compare register of the TPU, i.e. TPUm.TGRA (m = 0 to 3) so that the corresponding compare match signal is the output trigger. Set a cycle in TGRA so that the counter will be cleared by compare match A. Set the TGIEA bit in TPUm.TIER to 1 to enable the compare match/input capture A (TGImA) interrupt.
2. Write F8h to PPG0.NDERH, and set the G3CMS[1:0] and G2CMS[1:0] bits in PPG0.PCR to select the respective compare matches in the TPUm selected in the previous step to be the output triggers. Write output data 80h to PPG0.NDRH.
3. The timer counter in the TPUm starts. When compare match A occurs, the values in PPG0.NDRH are transferred to PPG0.PODRH and output. The TGImA interrupt handling routine writes the next output data C0h to PPG0.NDRH.
4. Five-phase pulse output (one or two phases active at a time) can be obtained subsequently by writing 40h, 60h, 20h, 30h, 10h, 18h, 08h, 88h... at successive TGImA interrupts.
   If the DTC or DMAC is set for activation by the TGImA interrupt, pulse output can be obtained without imposing a load on the CPU.



Downloaded from Arrow.com.

### 16.3.4 Non-Overlapping Pulse Output

During non-overlapping operation, data transfer from PPGm.NDRH and PPGm.NDRL (m = 0, 1) to PPGm.PODRH and PPGm. PODRL is performed as follows.

- On compare match A, the values in PPGm.NDRH and PPGm.NDRL are always transferred to PPGm.PODRH and PPGm. PODRL.
- On compare match B, data transfer proceeds for bits in PPGm.NDRH and PPGm.NDRL that have the value 0. It does not proceed for bits having the value 1.

Figure 16.9 shows the non-overlapping pulse output operation.

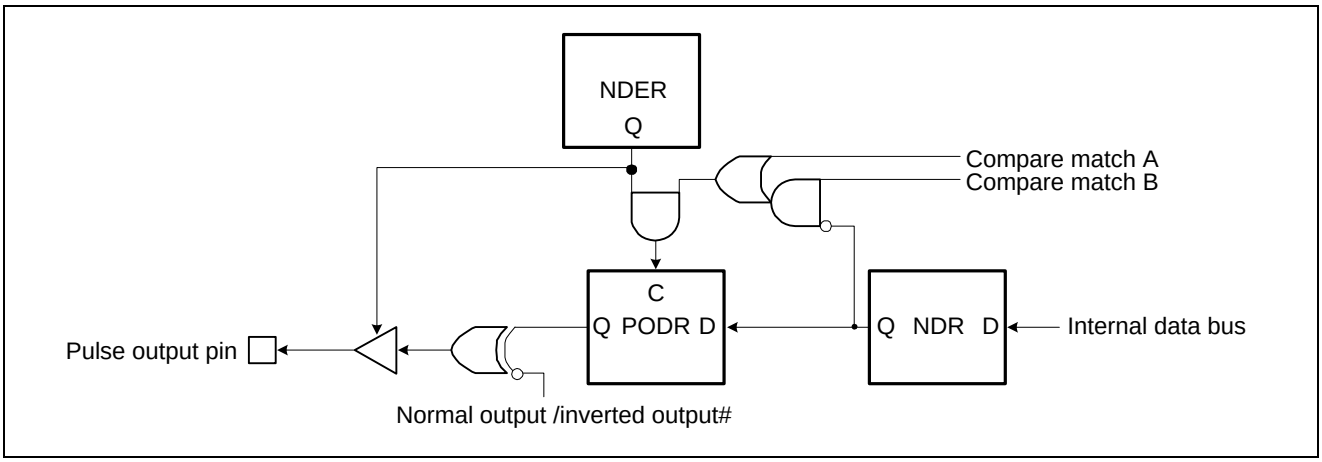



**Figure 16.9 Non-Overlapping Pulse Output**

Therefore, compare match B before compare match A allows 0-valued data to be transferred in advance of 1-valued data.

Do not change the values in PPGm.NDRH and PPGm.NDRL during the interval from compare match B to compare match A (the non-overlap margin).

To transfer 0-valued data in advance of 1-valued data, write the next data to PPGm.NDRH and PPGm.NDRL from within the TGIA interrupt handling routine or by using a TGIA interrupt to activate transfer by the DTC or DMAC. In any case, the next data must be written before the next compare match B occurs.

Figure 16.10 shows the timing of the above procedure.



Downloaded from Arrow.com.

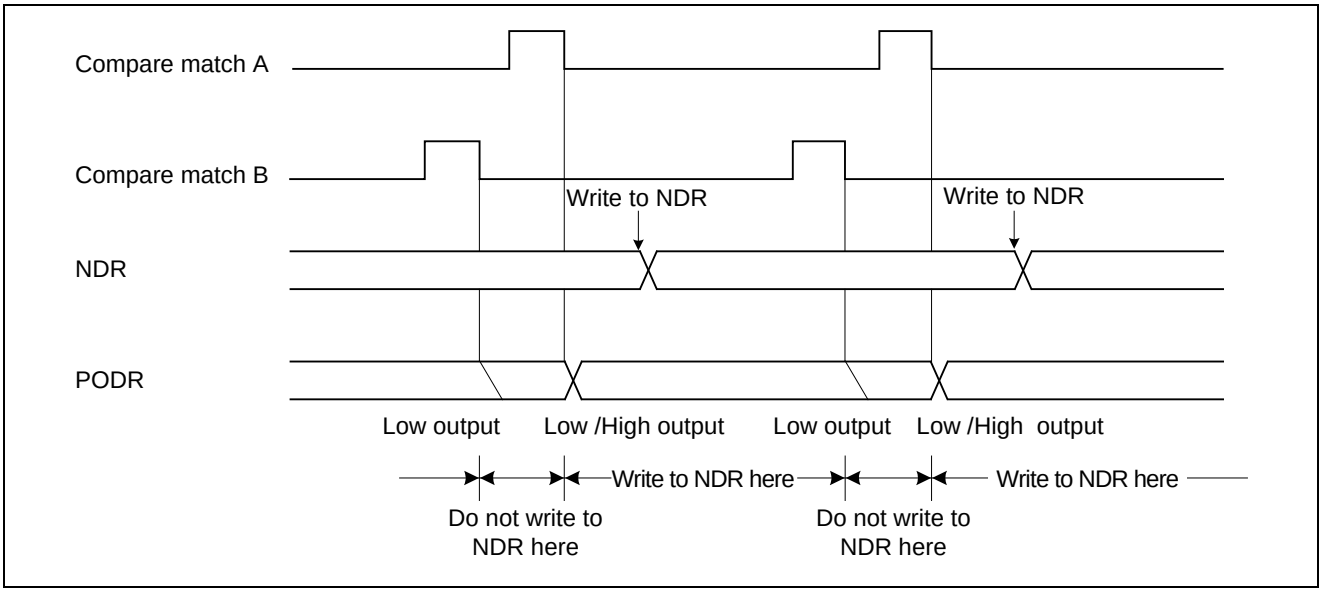



**Figure 16.10 Non-Overlapping Operation and Write Timing to PPGm.NDRH and PPGm.NDRL**



Downloaded from Arrow.com.

### 16.3.5 Sample Setup Procedure for Non-Overlapping Pulse Output

Figures 16.11 and 16.12 show sample procedures for setting up non-overlapping pulse outputs.

(1) PPG0 Setting

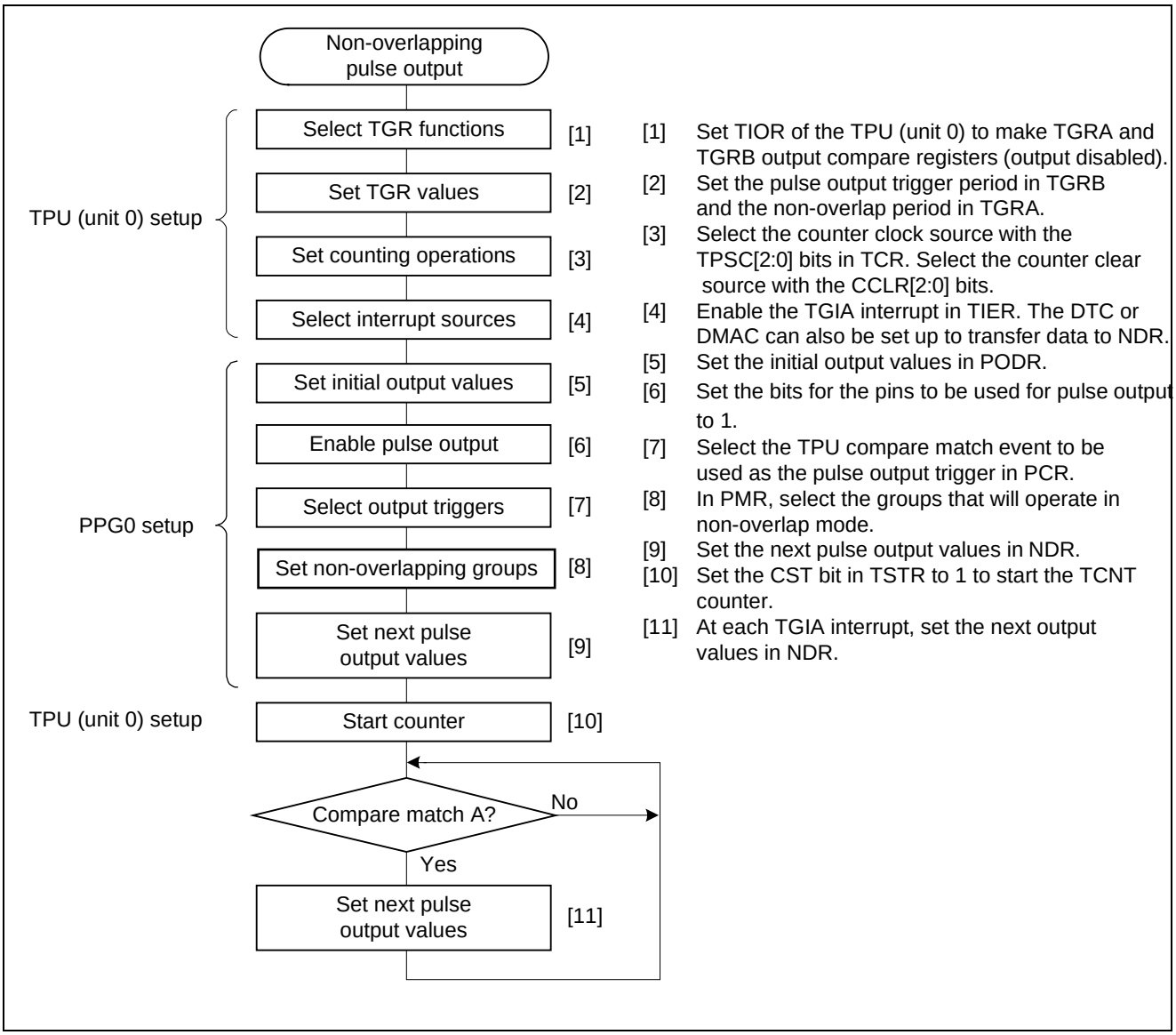
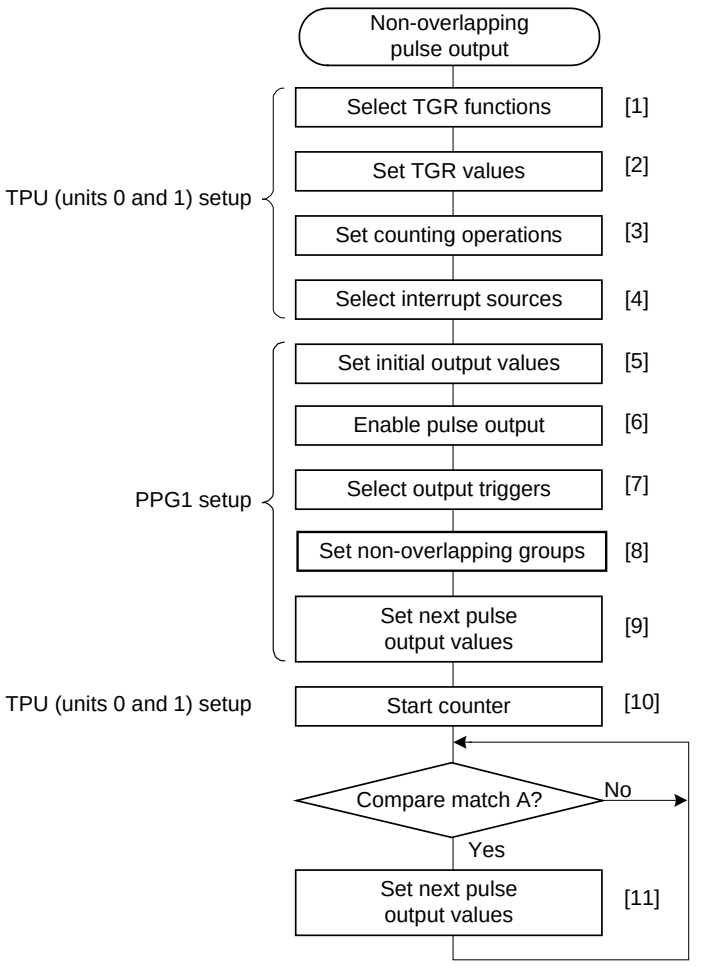



**Figure 16.11 Sample Setup Procedure for Non-Overlapping Pulse Output (PPG0 Setting)**



Downloaded from Arrow.com.

## (2) PPG1 Setting




[1] In the case of unit 0, set TIOR of the TPU to make TGRA and TGRB output compare registers (output disabled).
In the case of unit 1, set TIOR of the TPU to make TGRA and TGRB output compare registers (toggle output).
[2] Set the pulse output trigger period in TGRB and the non-overlap period in TGRA.
[3] Select the counter clock source with the TPSC[2:0] bits in TCR.
Select the counter clear source with the CCLR[2:0] bits.
[4] Enable the TGIA interrupt in TIER. The DTC or DMAC can also be set up to transfer data to NDR.
[5] Set the initial output values in PODR.
[6] Set the bits for the pins to be used for pulse output to 1.
[7] Select the TPU compare match event to be used as the pulse output trigger in PTRSLR and PCR.
[8] In PMR, select the groups that will operate in non-overlap mode.
[9] Set the next pulse output values in NDR.
[10] Set the CST bit in TSTR to 1 to start the TCNT counter.
[11] At each TGIA interrupt, set the next output values in NDR.

**Figure 16.12 Sample Setup Procedure for Non-Overlapping Pulse Output (PPG1 Setting)**



Downloaded from Arrow.com.

### 16.3.6 Example of Non-Overlapping Pulse Output (Example of Four-Phase Complementary Non-Overlapping Output)

Figure 16.13 shows an example in which pulse output from the PPG0 is used for four-phase complementary non-overlapping pulse output.

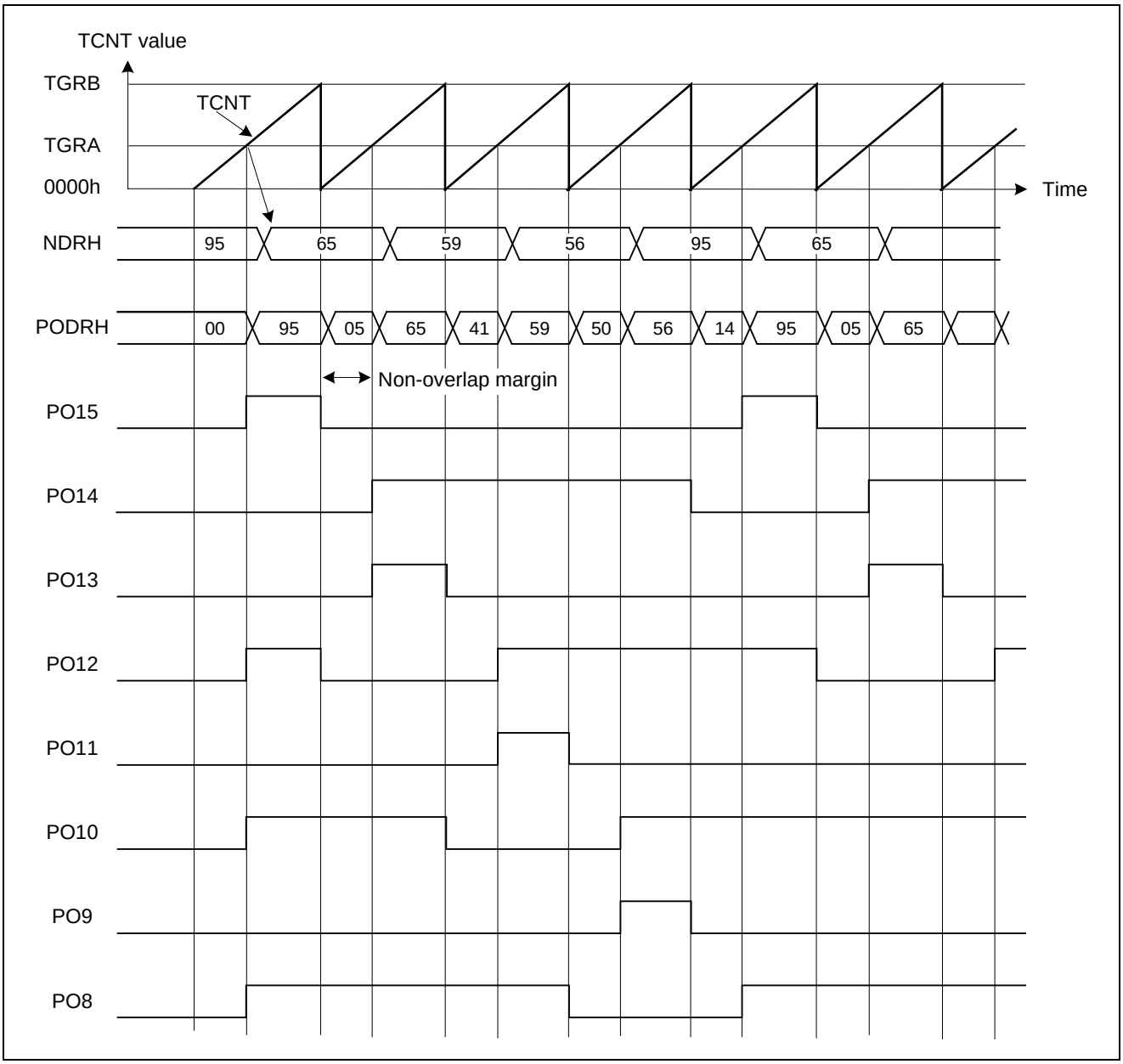



**Figure 16.13 Example of Non-Overlapping Pulse Output (Example of Four-Phase Complementary Non-Overlapping Output)**



Downloaded from Arrow.com.

1. Set output compare registers of the TPU, i.e. TPUm.TGRA and TPUm.TGRB (m = 0 to 3) so that the corresponding compare match signals are the output triggers. Set the trigger period in TGRB and the non-overlap margin in TGRA, and set the counter to be cleared by compare match B. Set the TGIEA bit in TPUm.TIER to 1 to enable the compare match/input capture A (TGImA) interrupt.
2. Write FFh in PPG0.NDERH, and set the G3CMS[1:0] and G2CMS[1:0] bits in PPG0.PCR to select the respective compare matches in the TPUm selected in the previous step to be the output triggers.
   Set the G3NOV and G2NOV bits in PPG0.PMR to 1 to select non-overlapping outputs. Write output data 95h in PPG0.NDRH.
3. The timer counter in the TPUm starts. When a compare match with TGRB occurs, outputs change from high to low. When a compare match with TGRA occurs, outputs change from low to high (the change from low to high is delayed by the value set in TGRA).
   The TGImA interrupt handling routine writes the next output data 65h in PPG0.NDRH.
4. Four-phase complementary non-overlapping pulse output can be obtained subsequently by writing 59h, 56h, 95h... at successive TGImA interrupts.
   If the DTC or DMAC is set for activation by the TGImA interrupt, pulse output can be obtained without imposing a load on the CPU.



Downloaded from Arrow.com.

### 16.3.7 Inverted Pulse Output

When the G3INV, G2INV, G1INV, and G0INV bits in PPG0.PMR are cleared to 0, the values that are the inverse of the respective values in PPG0.PODRH and PPG0.PODRL can be output.

Figure 16.14 shows the outputs when the G3INV and G2INV bits are cleared to 0 in addition to the settings in figure 16.13.

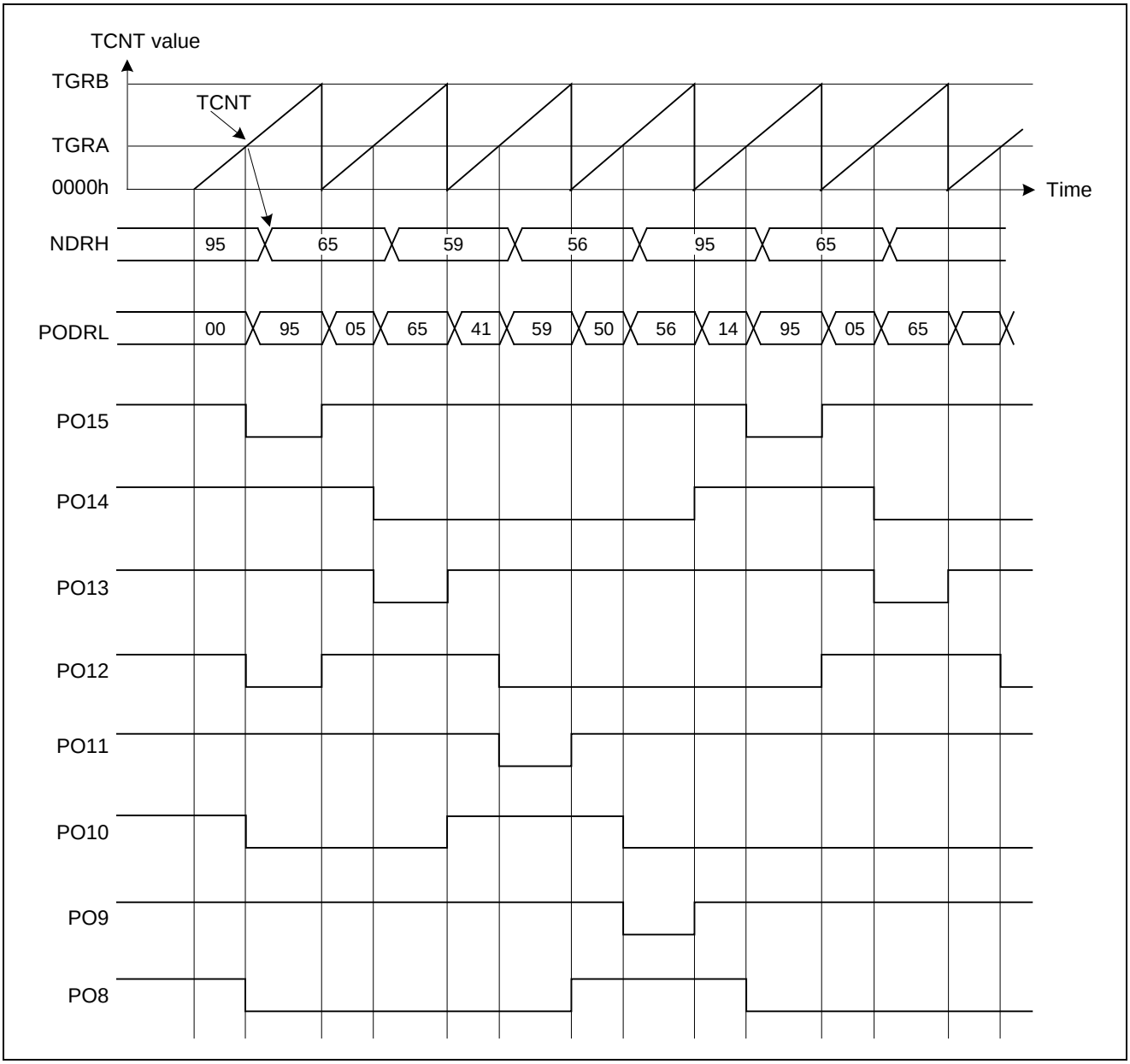



**Figure 16.14 Inverted Pulse Output (Example)**



Downloaded from Arrow.com.

### 16.3.8 Pulse Output Triggered by Input Capture

Pulse output from the PPG0 can be triggered by the TPU (unit 0) input capture as well as by compare match. When a TPUm.TGRA (m = 0 to 3) functions as an input capture register in the TPU (unit 0) channel selected by PPG0.PCR, pulse output is triggered by the input capture signal.

Figure 16.15 shows the timing of pulse output triggered by input capture.

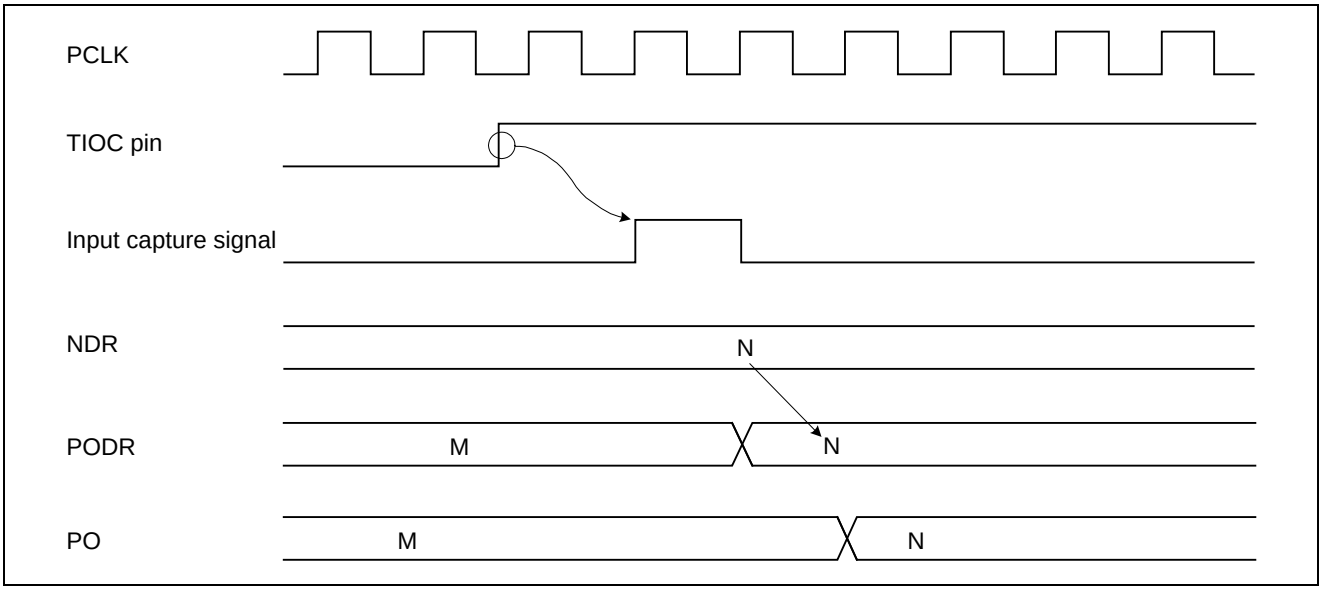



**Figure 16.15 Timing of Pulse Output Triggered by Input Capture (Example)**



Downloaded from Arrow.com.

## 16.4 Usage Note

### 16.4.1 Module Stop Function Setting

Operation of the PPG can be disabled or enabled by the module stop control register. The initial setting is for operation of the PPG to be halted. Register access is enabled by clearing module stop state. For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

# 17. 8-Bit Timer (TMR)

The RX610 Group has two units (unit 0, unit 1) of an on-chip 8-bit timer (TMR) module that comprise two 8-bit counter channels, totaling four channels. The 8-bit timer module can be used to count external events and also be used as a multifunction timer in a variety of applications, such as generation of counter reset, interrupt requests, and pulse output with a desired duty cycle using a compare-match signal with two registers.

Unit 0 and unit 1 can generate a baud rate clock signal for the SCI and have the same functions.

## 17.1 Overview

Table 17.1 shows the specifications of the TMR.

Figure 17.1 shows a block diagram of the 8-bit timer module (unit 0), and figure 17.2 shows that of the 8-bit timer module (unit 1).

**Table 17.1 Specifications of TMR**

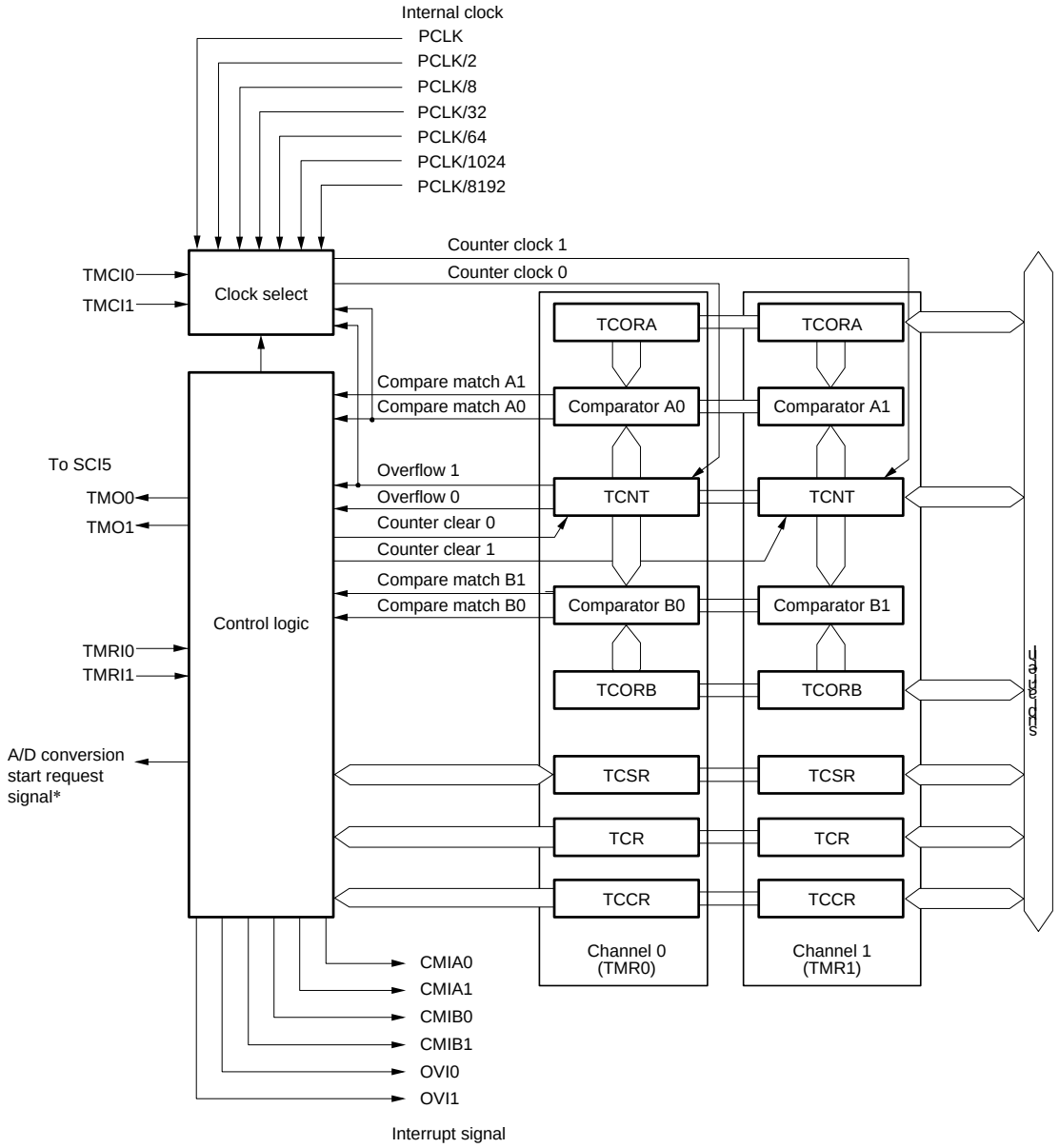
| Item | Description |
|---|---|
| Count clock | • Internal clock: PCLK, PCLK/2, PCLK/8, PCLK/32, PCLK/64, PCLK/1024, PCLK/8192<br>• External clock |
| Number of channels | (8 bits x 2 channels) x 2 units |
| Compare match | • 8-bit mode (compare match A, compare match B)<br>• 16-bit mode (compare match A, compare match B) |
| Counter clear | Selected by compare match A or B, or an external reset signal. |
| Timer output | Output pulses with a desired duty cycle or PWM output |
| Cascading of two channels | • 16-bit count mode<br>16-bit timer using TMR0 for the upper 8 bits and TMR1 for the lower 8 bits (TMR2 for the upper 8 bits and TMR3 for the lower 8 bits)<br>• Compare match count mode<br>TMR1 can be used to count TMR0 compare matches (TMR3 can be used to count TMR2 compare matches). |
| Interrupt sources | Compare match A, compare match B, and overflow |
| DTC activation | DTC can be activated by compare match A interrupts or compare match B interrupts. |
| Generation of trigger to start A/D converter conversion | Compare match A of TMR0 and TMR2*1 |
| Capable of generating baud rate clock for SCI | Generates baud rate clock for SCI5 and SCI6.*2 |
| Power-down function | Each unit can be placed in a module stop state. |

Notes: 1. For details, see section 23, A/D Converter.

2. For details, see section 20, Serial Communication Interface (SCI).



Downloaded from Arrow.com.




[Legend]

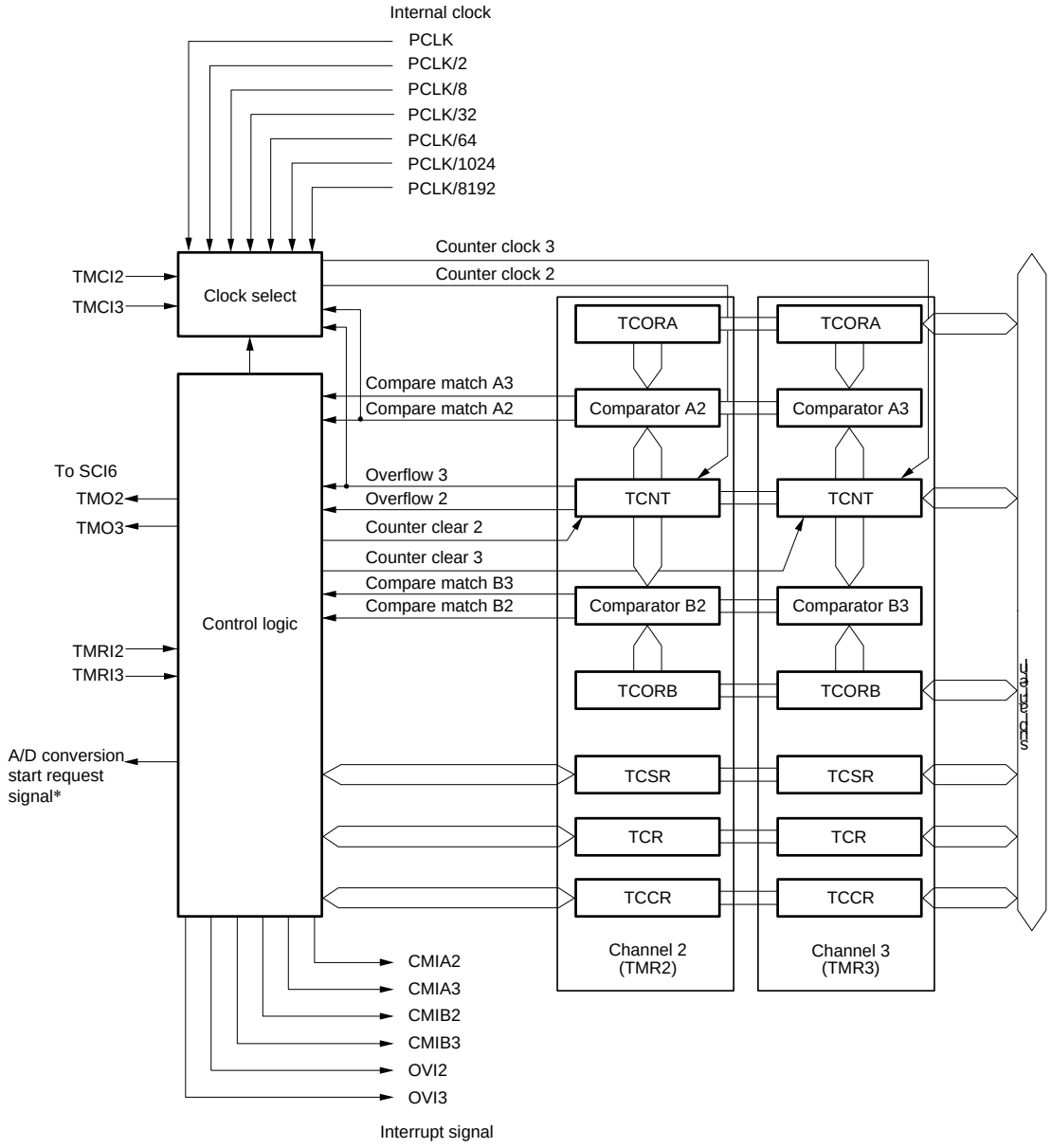
| | |
|---|---|
| TCORA: | Time constant register A |
| TCNT: | Timer counter |
| TCORB: | Time constant register B |
| TCSR: | Timer control/status register |
| TCR: | Timer control register |
| TCCR: | Timer counter control register |

Note: * For the corresponding A/D converter channels, see section 23, A/D Converter.

**Figure 17.1 Block Diagram of TMR (Unit 0)**



Downloaded from Arrow.com.




[Legend]

| | |
|---|---|
| TCORA: | Time constant register A |
| TCNT: | Timer counter |
| TCORB: | Time constant register B |
| TCSR: | Timer control/status register |
| TCR: | Timer control register |
| TCCR: | Timer counter control register |

Note: * For he corresponding A/D converter channels, see sec ion 23, A/D Converter.

**Figure 17.2 Block Diagram of TMR (Unit 1)**



Downloaded from Arrow.com.

Table 17.2 lists the input/output pins of the TMR.

**Table 17.2 Pin Configuration of TMR**

| Unit | Channel | Pin Name | I/O | Description |
|---|---|---|---|---|
| 0 | TMR0 | TMO0 | Output | Outputs compare match |
| | | TMCI0 | Input | Inputs external clock for counter |
| | | TMRI0 | Input | Inputs external reset to counter |
| | TMR1 | TMO1 | Output | Outputs compare match |
| | | TMCI1 | Input | Inputs external clock for counter |
| | | TMRI1 | Input | Inputs external reset to counter |
| 1 | TMR2 | TMO2 | Output | Outputs compare match |
| | | TMCI2 | Input | Inputs external clock for counter |
| | | TMRI2 | Input | Inputs external reset to counter |
| | TMR3 | TMO3 | Output | Outputs compare match |
| | | TMCI3 | Input | Inputs external clock for counter |
| | | TMRI3 | Input | Inputs external reset to counter |



Downloaded from Arrow.com.

## 17.2 Register Descriptions

Table 17.3 shows the registers of the TMR.

**Table 17.3 Registers of TMR**

| Unit | Channel | Register Name | Symbol | Value after Reset | Address* | Access Size |
|---|---|---|---|---|---|---|
| 0 | TMR0 | Timer counter | TCNT | 00h | 0008 8208h | 8 or 16 |
| | | Time constant register A | TCORA | FFh | 0008 8204h | 8 or 16 |
| | | Time constant register B | TCORB | FFh | 0008 8206h | 8 or 16 |
| | | Timer control register | TCR | 00h | 0008 8200h | 8 |
| | | Timer counter control register | TCCR | 00h | 0008 820Ah | 8 or 16 |
| | | Timer control/status register | TCSR | x0h | 0008 8202h | 8 |
| | TMR1 | Timer counter | TCNT | 00h | 0008 8209h | 8 or 16* |
| | | Time constant register A | TCORA | FFh | 0008 8205h | 8 or 16* |
| | | Time constant register B | TCORB | FFh | 0008 8207h | 8 or 16* |
| | | Timer control register | TCR | 00h | 0008 8201h | 8 |
| | | Timer counter control register | TCCR | 00h | 0008 820Bh | 8 or 16* |
| | | Timer control/status register | TCSR | x0h | 0008 8203h | 8 |
| 1 | TMR2 | Timer counter | TCNT | 00h | 0008 8218h | 8 or 16 |
| | | Time constant register A | TCORA | FFh | 0008 8214h | 8 or 16 |
| | | Time constant register B | TCORB | FFh | 0008 8216h | 8 or 16 |
| | | Timer control register | TCR | 00h | 0008 8210h | 8 |
| | | Timer counter control register | TCCR | 00h | 0008 821Ah | 8 or 16 |
| | | Timer control/status register | TCSR | x0h | 0008 8212h | 8 |
| | TMR3 | Timer counter | TCNT | 00h | 0008 8219h | 8 or 16* |
| | | Time constant register A | TCORA | FFh | 0008 8215h | 8 or 16* |
| | | Time constant register B | TCORB | FFh | 0008 8217h | 8 or 16* |
| | | Timer control register | TCR | 00h | 0008 8211h | 8 |
| | | Timer counter control register | TCCR | 00h | 0008 821Bh | 8 or 16* |
| | | Timer control/status register | TCSR | x0h | 0008 8213h | 8 |

Note: * Odd addresses should not be accessed in 16-bit units. When accessing a register in 16-bit units, access the address of the TMR0 or TMR2 register. Table 17.4 lists register allocation for 16-bit access.

**Table 17.4 Register Allocation for 16-Bit Access**

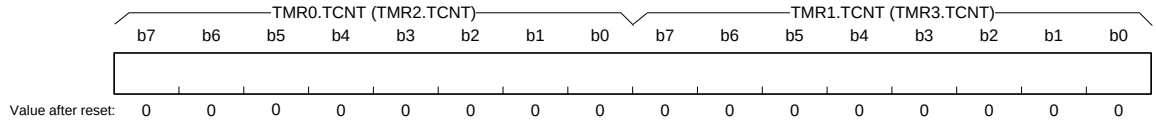
| Address | Upper 8 Bits | Lower 8 Bits |
|---|---|---|
| 0008 8208h | TMR0.TCNT | TMR1.TCNT |
| 0008 8204h | TMR0.TCORA | TMR1.TCORA |
| 0008 8206h | TMR0.TCORB | TMR1.TCORB |
| 0008 820Ah | TMR0.TCCR | TMR1.TCCR |
| 0008 8218h | TMR2.TCNT | TMR3.TCNT |
| 0008 8214h | TMR2.TCORA | TMR3.TCORA |
| 0008 8216h | TMR2.TCORB | TMR3.TCORB |
| 0008 821Ah | TMR2.TCCR | TMR3.TCCR |



Downloaded from Arrow.com.

### 17.2.1 Timer Counter (TCNT)

Addresses: TMR0.TCNT 0008 8208h, TMR1.TCNT 0008 8209h
TMR2.TCNT 0008 8218h, TMR3.TCNT 0008 8219h




TCNT is an 8-bit readable/writable up-counter.

TMR0.TCNT and TMR1.TCNT (TMR2.TCNT and TMR3.TCNT) comprise a single 16-bit counter so they can be accessed together by a word transfer instruction.

The TCCR.CSS[1:0] and CKS[2:0] bits are used to select a clock.

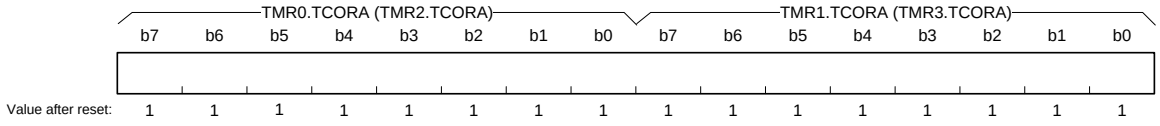
TCNT can be cleared by an external reset input signal, compare match A signal, or compare match B signal. Which signal to be used for clearing is selected by the TCR.CCLR[1:0] bits.

When TCNT overflows from FFh to 00h, the interrupt flag is set to 1.

For details on the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU), and table 17.6, TMR Interrupt Sources.

### 17.2.2 Time Constant Register A (TCORA)

Addresses: TMR0.TCORA: 0008 8204h, TMR1.TCORA: 0008 8205h
TMR2.TCORA: 0008 8214h, TMR3.TCORA: 0008 8215h




TCORA is an 8-bit readable/writable register.

TMR0.TCORA and TMR1.TCORA (TMR2.TCORA and TMR3.TCORA) comprise a single 16-bit register so they can be accessed together by a word transfer instruction.

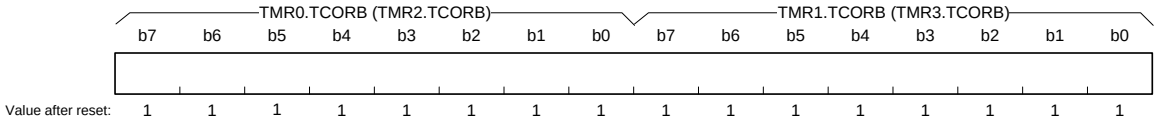
The value in TCORA is continually compared with the value in TCNT. When a match is detected, the corresponding compare match A signal is set to high. Note however that comparison is not performed during writing to TCORA. The timer output from the TMOn pin can be freely controlled by this compare match A signal and the settings of the TCSR.OSA[1:0] bits.



Downloaded from Arrow.com.

### 17.2.3 Time Constant Register B (TCORB)

Addresses: TMR0.TCORB 0008 8206h, TMR1.TCORB 0008 8207h
TMR2.TCORB 0008 8216h, TMR3.TCORB 0008 8217h




TCORB is an 8-bit readable/writable register.

TMR0.TCORB and TMR1.TCORB (TMR2.TCORB and TMR3.TCORB) comprise a single 16-bit register so they can be accessed together by a word transfer instruction.

The value in TCORB is continually compared with the value in TCNT. When a match is detected, the corresponding compare match B signal is set to high. Note however that comparison is not performed during writing to TCORB. The timer output from the TMOn pin can be freely controlled by this compare match B signal and the settings of the TCSR.OSB[1:0] bits.



Downloaded from Arrow.com.

## 17.2.4 Timer Control Register (TCR)

Addresses: TMR0.TCR 0008 8200h, TMR1.TCR 0008 8201h
TMR2.TCR 0008 8210h, TMR3.TCR 0008 8211h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CMIEB | CMIEA | OVIE | CCLR[1:0] | | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4, b3 | CCLR[1:0] | Counter Clear* | b4 b3<br>0 0: Clearing is disabled<br>0 1: Cleared by compare match A<br>1 0: Cleared by compare match B<br>1 1: Cleared by the external reset input<br>(Select edge or level by the TMRIS bit in TCCR.) | R/W |
| b5 | OVIE | Timer Overflow Interrupt Enable | 0: Overflow interrupt requests (OVIm) are disabled<br>1: Overflow interrupt requests (OVIm) are enabled | R/W |
| b6 | CMIEA | Compare Match Interrupt Enable A | 0: Compare match A interrupt requests (CMIAm) are disabled<br>1: Compare match A interrupt requests (CMIAm) are enabled | R/W |
| b7 | CMIEB | Compare Match Interrupt Enable B | 0: Compare match B interrupt requests (CMIBm) are disabled<br>1: Compare match B interrupt requests (CMIBm) are enabled | R/W |

Note: * To use an external reset, set the Pn.DDR.Bi bit for the corresponding pin to "0" and the Pn.ICR.Bi bit to "1".

TCR specifies the condition for clearing TCNT.

**CCLR[1:0] Bits (Counter Clear)**

Select the method by which TCNT is cleared.

**OVIE Bit (Timer Overflow Interrupt Enable)**

Selects whether overflow interrupt requests (OVIm) issued by TCNT are enabled or disabled.

**CMIEA Bit (Compare Match Interrupt Enable A)**

Selects whether compare match A interrupt requests (CMIAm) that are issued when the value of TCORA corresponds to that of TCNT are enabled or disabled.

**CMIEB Bit (Compare Match Interrupt Enable B)**

Selects whether compare match B interrupt requests (CMIBm) that are issued when the value of TCORB corresponds to that of TCNT are enabled or disabled.



Downloaded from Arrow.com.

## 17.2.5 Timer Counter Control Register (TCCR)

Addresses: TMR0.TCCR 0008 820Ah, TMR1.TCCR 0008 820Bh
TMR2.TCCR 0008 821Ah, TMR3.TCCR 0008 821Bh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TMRIS | — | — | CSS[1:0] | | CKS[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | CKS[2:0] | Clock Select* | See table 17.5. | R/W |
| b4, b3 | CSS[1:0] | Clock Source Select | See table 17.5. | R/W |
| b6, b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | TMRIS | Timer Reset Detection Condition Select | 0: Cleared at rising edge of the external reset<br>1: Cleared when the external reset is high | R/W |

Note * To use an external clock, set the Pn.DDR.Bi bit for the corresponding pin to "0" and the PnICR.Bi bit to "1". For details, see section 14, I/O Ports.

TCCR selects an internal clock source for TCNT and the condition for detecting external reset.

**CKS[2:0] Bits (Clock Select)**
**CSS[1:0] Bits (Clock Source Select)**

The CKS[2:0] and CSS[1:0] bits select a clock. For details, see table 17.5.

**TMRIS Bit (Timer Reset Detection Condition Select)**

This bit is enabled when the TCR.CCLR[1:0] bits are 11b (cleared by external reset input) and selects the condition for detecting external reset (level or edge).



Downloaded from Arrow.com.

**Table 17.5 Clock Input to TCNT and Count Condition**

| | TCCR Register | | | | | |
|---|---|---|---|---|---|---|
| | CSS[1:0] | | CKS[2:0] | | | |
| Channel | b4 | b3 | b2 | b1 | b0 | Description |
| TMR0 (TMR2) | 0 | 0 | — | 0 | 0 | Clock input prohibited |
| | | | | | 1 | Uses external clock. Counts at rising edge*1. |
| | | | | 1 | 0 | Uses external clock. Counts at falling edge*1. |
| | | | | | 1 | Uses external clock. Counts at both rising and falling edges*1. |
| | 0 | 1 | 0 | 0 | 0 | Uses internal clock. Counts at PCLK. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/2. |
| | | | | 1 | 0 | Uses internal clock. Counts at PCLK/8. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/32. |
| | | | 1 | 0 | 0 | Uses internal clock. Counts at PCLK/64. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/1024. |
| | | | | 1 | 0 | Uses internal clock. Counts at PCLK/8192. |
| | | | | | 1 | Clock input prohibited |
| | 1 | 0 | — | — | — | Setting prohibited |
| | 1 | 1 | — | — | — | Counts at TMR1.TCNT (TMR3.TCNT) overflow signal*2. |
| TMR1 (TMR3) | 0 | 0 | — | 0 | 0 | Clock input prohibited |
| | | | | | 1 | Uses external clock. Counts at rising edge*1. |
| | | | | 1 | 0 | Uses external clock. Counts at falling edge*1. |
| | | | | | 1 | Uses external clock. Counts at both rising and falling edges*1. |
| | 0 | 1 | 0 | 0 | 0 | Uses internal clock. Counts at PCLK. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/2. |
| | | | | 1 | 0 | Uses internal clock. Counts at PCLK/8. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/32. |
| | | | 1 | 0 | 0 | Uses internal clock. Counts at PCLK/64. |
| | | | | | 1 | Uses internal clock. Counts at PCLK/1024. |
| | | | | 1 | 0 | Uses internal clock. Counts at PCLK/8192. |
| | | | | | 1 | Clock input prohibited |
| | 1 | 0 | — | — | — | Setting prohibited |
| | 1 | 1 | — | — | — | Counts at TMR0.TCNT (TMR2.TCNT) compare match A*2. |

Notes: 1. To use an external clock, set the Pn.DDR.Bi bit for the corresponding pin to "0" and the Pn.ICR.Bi bit to "1". For details, see section 14, I/O Ports.

2. If the clock input of TMR0 (TMR2) is the overflow signal of the TMR1.TCNT (TMR3.TCNT) counter and that of TMR1 (TMR3) is the compare match signal of the TMR0.TCNT (TMR2.TCNT) counter, no incrementing clock is generated. Do not use this setting.



Downloaded from Arrow.com.

## 17.2.6 Timer Control/Status Register (TCSR)

Addresses: TMR0.TCSR 0008 8202h, TMR2.TCSR 0008 8212h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | ADTE | OSB[1:0] | | OSA[1:0] | |
| Value after reset: | x | x | x | 0 | 0 | 0 | 0 | 0 |

Addresses: TMR1.TCSR 0008 8203h, TMR3.TCSR 0008 8213h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | OSB[1:0] | | OSA[1:0] | |
| Value after reset: | x | x | x | 1 | 0 | 0 | 0 | 0 |

[Legend] x: Undefined

- TMR0.TCSR, TMR2.TCSR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | OSA[1:0] | Output Select A*[1] | b1 b0<br>0 0: No change when compare match A occurs<br>0 1: Low is output when compare match A occurs<br>1 0: High is output when compare match A occurs<br>1 1: Output is inverted when compare match A occurs (toggle output) | R/W |
| b3, b2 | OSB[1:0] | Output Select B*[1] | b3 b2<br>0 0: No change when compare match B occurs<br>0 1: Low is output when compare match B occurs<br>1 0: High is output when compare match B occurs<br>1 1: Output is inverted when compare match B occurs (toggle output) | R/W |
| b4 | ADTE | A/D Trigger Enable*[2] | 0: A/D converter start requests by compare match A are disabled<br>1: A/D converter start requests by compare match A are enabled | R/W |
| b7 to b5 | — | Reserved | These bits are always read as an undefined value. The write value should always be 1. | R/W |

Notes: 1. Timer output is disabled when the OSB[1:0] and OSA[1:0] bits are all 0. Timer output is 0 until the first compare match occurs after a reset.

2. For the corresponding A/D converter channels, see section 23, A/D Converter.



Downloaded from Arrow.com.

- TMR1.TCSR, TMR3.TCSR

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | OSA[1:0] | Output Select A* | b1 b0<br>0 0: No change when compare match A occurs<br>0 1: Low is output when compare match A occurs<br>1 0: High is output when compare match A occurs<br>1 1: Output is inverted when compare match A occurs (toggle output) | R/W |
| b3, b2 | OSB[1:0] | Output Select B* | b3 b2<br>0 0: No change when compare match B occurs<br>0 1: Low is output when compare match B occurs<br>1 0: High is output when compare match B occurs<br>1 1: Output is inverted when compare match B occurs (toggle output) | R/W |
| b4 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b7 to b5 | — | Reserved | These bits are always read as an undefined value. The write value should always be 1. | R/W |

Note: * Timer output is disabled when the OSB[1:0] and OSA[1:0] bits are all 0. Timer output is 0 until the first compare match occurs after a reset.

TCSR controls compare match output.

**OSA[1:0] Bits (Output Select A)**

These bits select a method of TMOn pin output when compare match A of TCORA and TCNT occurs.

**OSB[1:0] Bits (Output Select B)**

These bits select a method of TMOn pin output when compare match B of TCORB and TCNT occurs.

**ADTE Bit (A/D Trigger Enable)**

Selects enabling or disabling of A/D converter start requests by compare match A.

This bit is reserved for TMR1.TCSR and TMR3.TCSR.



Downloaded from Arrow.com.

## 17.3 Operation

### 17.3.1 Pulse Output

Figure 17.3 shows an example of the 8-bit timer being used to generate a pulse output with a desired duty cycle.

1. Set the TCR.CCLR[1:0] bits to 01b (cleared by compare match A) so that TCNT is cleared at a compare match of TCORA.
2. Set the TCSR.OSA[1:0] bits to 10b (high-output) and TCSR.OSB[1:0] bits to 01b (low-output), causing the output to change to high at a compare match of TCORA and to low at a compare match of TCORB.

With these settings, the 8-bit timer provides pulses output at a cycle determined by TCORA with a pulse width determined by TCORB. No software intervention is required.

The timer output is low until the first compare match occurs after a reset.

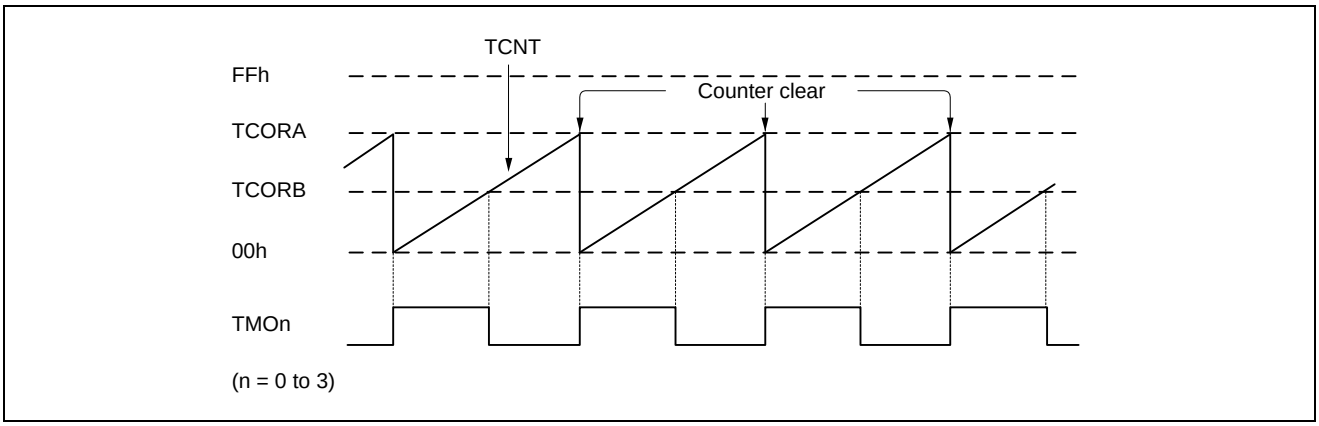



**Figure 17.3 Example of Pulse Output**



Downloaded from Arrow.com.

### 17.3.2 Reset Input

Figure 17.4 shows an example of the 8-bit timer being used to generate a pulse which is output after a desired delay time from a TMRIn input.

1. Set the TCR.CCLR[1:0] bits to 11b (cleared by external reset input) and set the TMRIS bit in TCCR to high (cleared when the external reset is high) so that TCNT is cleared at the high level input of the TMRIn signal.
2. Set the TCSR.OSA[1:0] bits to 10b (high-output) and the TCSR.OSB[1:0] bits to 01b (low-output), causing the output to change to high at a compare match of TCORA and to low at a compare match of TCORB.

With these settings, the 8-bit timer provides pulses output at a desired delay time from a TMRIn input determined by TCORA and with a pulse width determined by TCORB and TCORA.

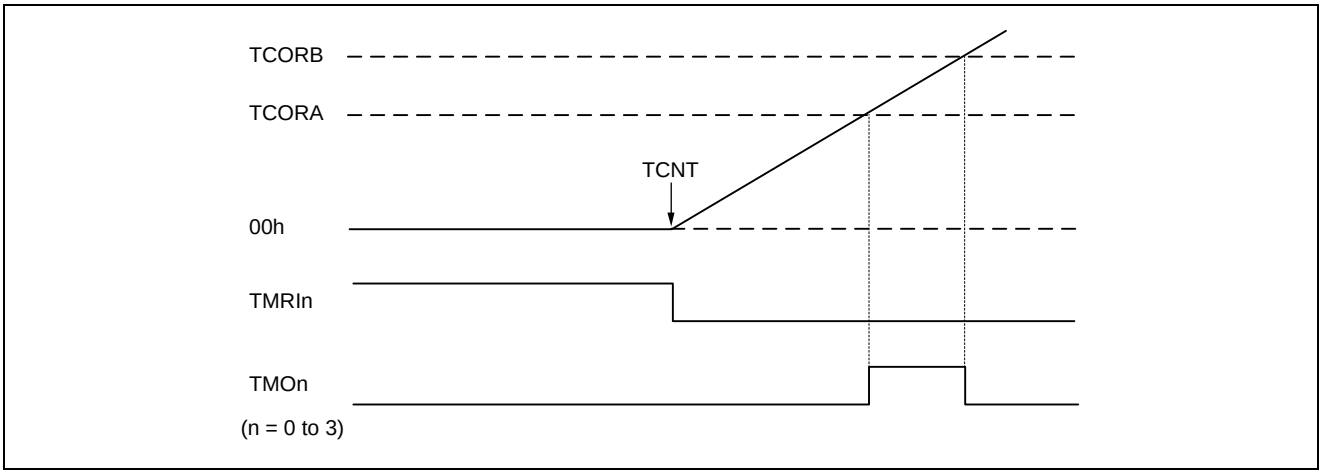



**Figure 17.4 Example of Reset Input**



Downloaded from Arrow.com.

## 17.4 Operation Timing

### 17.4.1 TCNT Count Timing

Figure 17.5 shows the count timing of TCNT for internal clock input. Figure 17.6 shows the count timing of TCNT for external clock input.

Note that the external clock pulse width must be at least 1.5 states for increment at a single edge, and at least 2.5 states for increment at both edges. The counter will not increment correctly if the pulse width is less than these values.

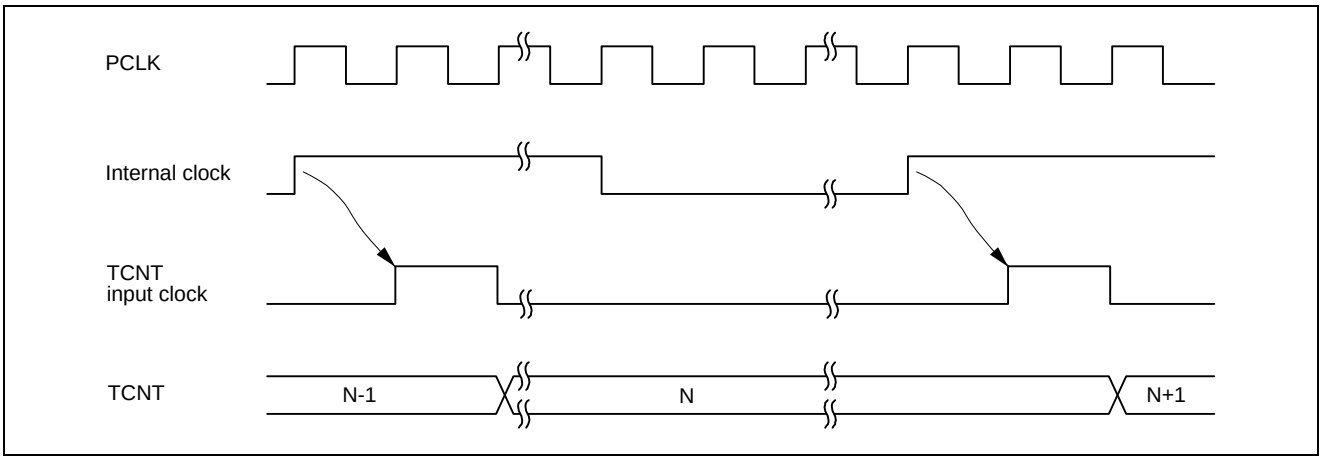



**Figure 17.5 Count Timing for Internal Clock Input**

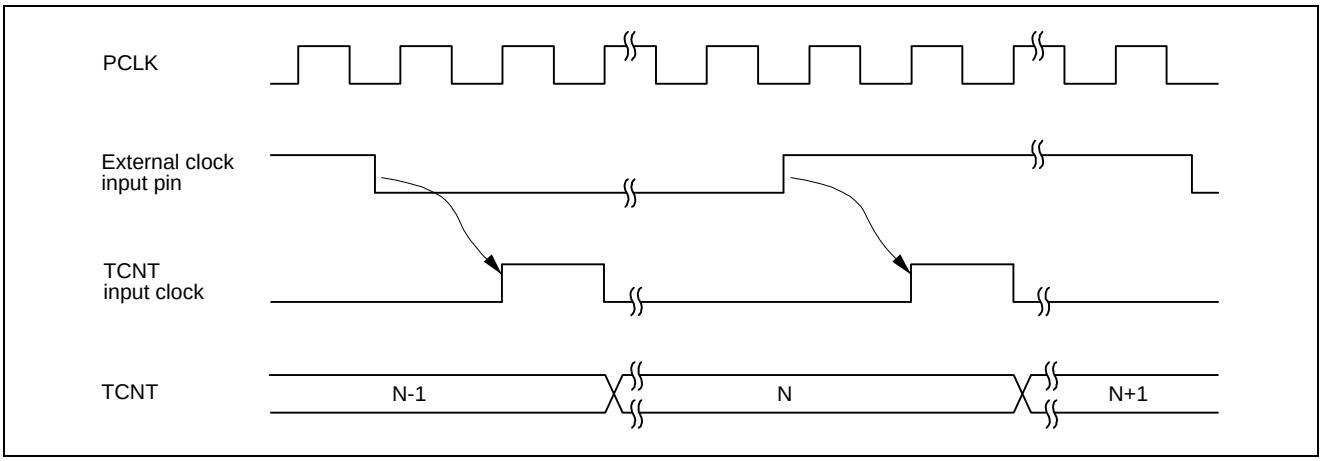



**Figure 17.6 Count Timing for External Clock Input**



Downloaded from Arrow.com.

### 17.4.2 Timing of Interrupt Flag Setting to 1 at Compare Match

The interrupt flag is set to 1 by a compare match signal generated when values of TCORA and TCORB and that of TCNT match.

The compare match signal is generated at the last state in which the match is true, just before the timer counter is updated. Therefore, when values of TCORA and TCORB and that of TCNT match, the compare match signal is not generated until the next TCNT clock input.

Figure 17.7 shows this timing. For details on the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU) and table 17.6, TMR Interrupt Sources.

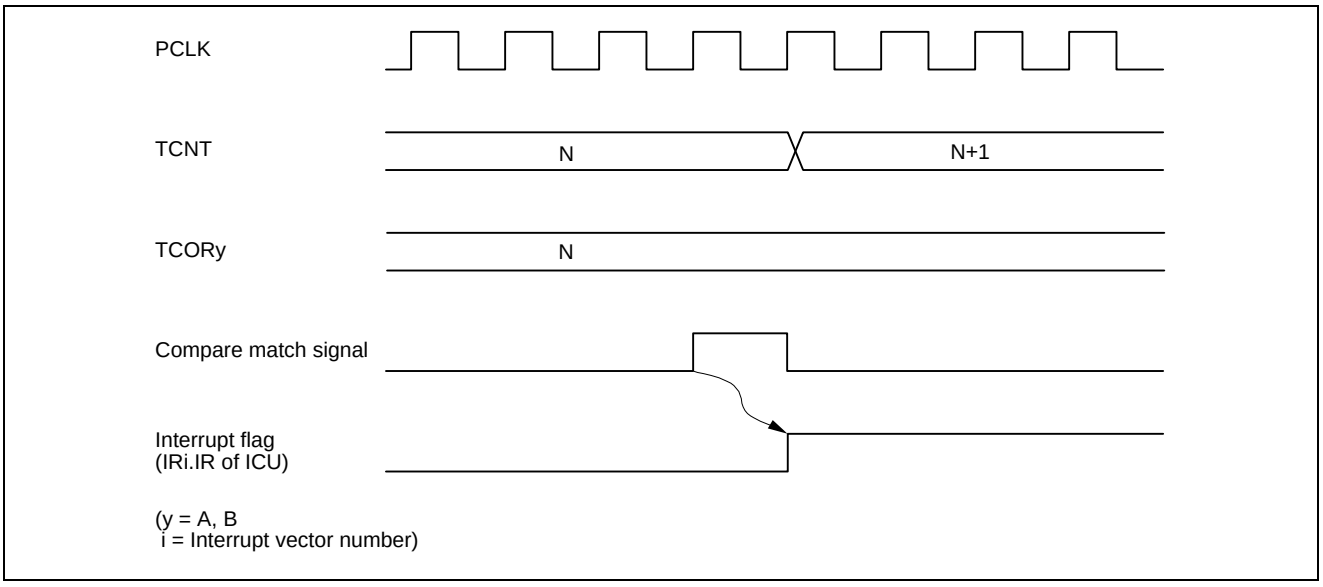



**Figure 17.7 Timing of Interrupt Flag Setting at Compare Match**



Downloaded from Arrow.com.

### 17.4.3 Timing of Timer Output at Compare Match

When a compare match signal is generated, the timer output changes as specified by the TCSR.OSA[1:0] and OSB[1:0] bits.

Figure 17.8 shows the timing when the timer output is toggled by the compare match A signal.

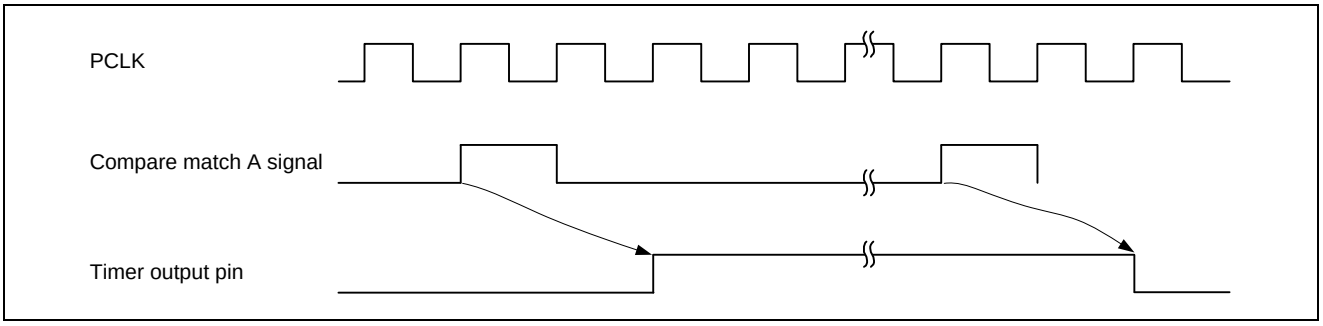



**Figure 17.8 Timing of Timer Output at Compare Match A**

### 17.4.4 Timing of Counter Clear by Compare Match

TCNT is cleared when compare match A or B occurs, depending on the settings of the TCR.CCLR[1:0] bits.

Figure 17.9 shows the timing of this operation.

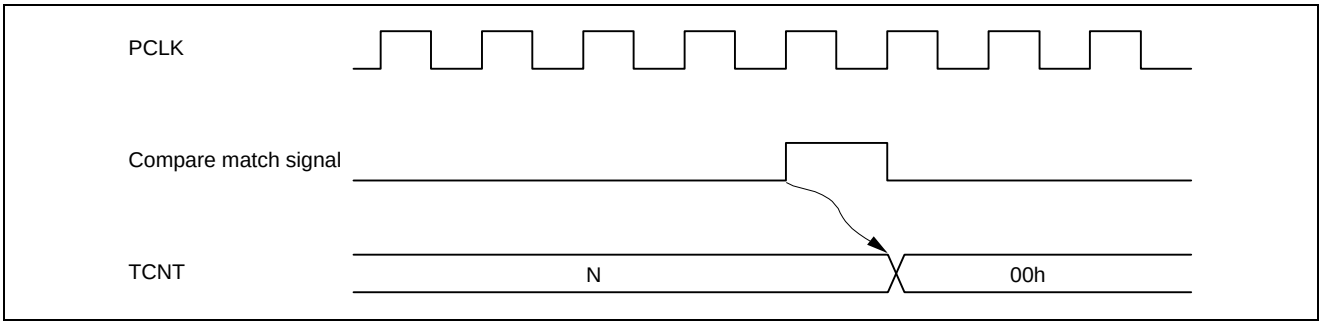



**Figure 17.9 Timing of Counter Clear by Compare Match**



Downloaded from Arrow.com.

### 17.4.5 Timing of the External Reset for TCNT

TCNT is cleared at the rising edge or high level of an external reset input, depending on the settings of the TCR.CCLR[1:0] bits. At least 2 states are required from an external reset input to clearing of TCNT.

Figures 17.10 and 17.11 show the timing of this operation.

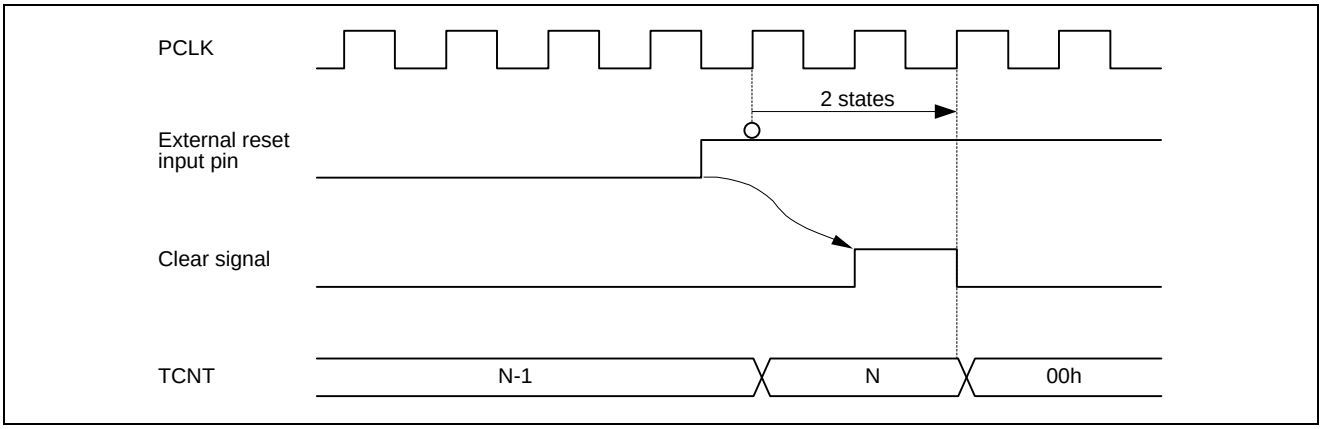



**Figure 17.10 Timing of Clearance by External Reset (Rising Edge)**

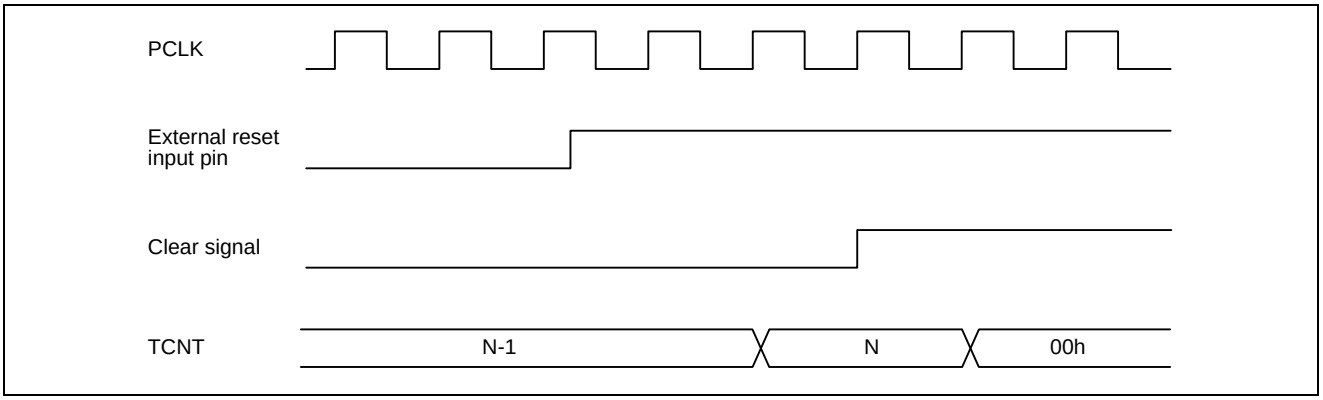



**Figure 17.11 Timing of Clearance by External Reset (High Level)**



Downloaded from Arrow.com.

### 17.4.6 Timing of Overflow Interrupt Flag Setting to 1

The interrupt flag is set to 1 by an overflow signal outputted when TCNT overflows (changes from FFh to 00h).

Figure 17.12 shows the timing of this operation.

For details on the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU) and table 17.6, TMR Interrupt Sources.

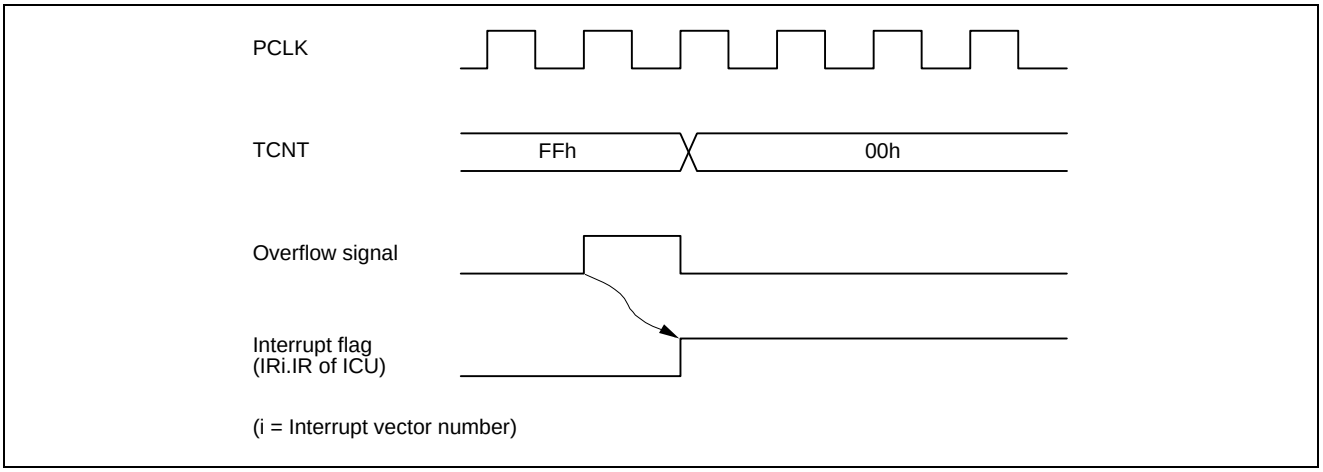



**Figure 17.12 Timing of Overflow Interrupt Flag Setting**



Downloaded from Arrow.com.

## 17.5 Operation with Cascaded Connection

If the CSS[1:0] bits in either TMR0.TCCR or TMR1.TCCR are set to 11b, the TMR of the two channels are cascaded. With this configuration, a single 16-bit timer could be used (16-bit count mode) or compare matches of TMR0 could be counted by TMR1 (compare match count mode).

This section describes unit 0. The operation of unit 1 with cascaded connection is the same as unit 0.

### 17.5.1 16-Bit Count Mode

When the TMR0.TCCR.CSS[1:0] bits are set to 11b, the timer functions as a single 16-bit timer with TMR0 occupying the upper 8 bits and TMR1 occupying the lower 8 bits. Only in this mode, registers whose Access Size column in table 17.3 is 8 or 16 can be accessed in 16-bit units.

#### (1) Counter Clear Specification

- The settings of the TMR0.TCR.CCLR[1:0] bits become effective for the 16-bit counter. If the TMR0.TCR.CCLR[1:0] bits have been set for counter clear at compare match, the 16-bit counter (TMR0.TCNT and TMR1.TCNT together) is cleared when a 16-bit compare match event occurs. The 16-bit counter (TMR0.TCNT and TMR1.TCNT together) is cleared even if counter clear by the TMRI0 pin has been set.
- The settings of the TMR1.TCR.CCLR[1:0] bits are ignored.

#### (2) Pin Output

- Control of output from the TMO0 pin by the TMR0.TCSR.OSA[1:0] and OSB[1:0] bits is in accordance with the 16-bit compare match conditions.
- Control of output from the TMO1 pin by the TMR1.TCSR.OSA[1:0] and OSB[1:0] bits is in accordance with the lower 8-bit compare match conditions.

### 17.5.2 Compare Match Count Mode

When the TMR1.TCCR.CSS[1:0] bits are set to 11b, TMR1.TCNT counts the number of occurrences of compare match A for TMR0. TMR0 and TMR1 are controlled independently. Conditions such as generation of interrupts, output from the TMOn (n = 0, 1) pin, and counter clear are in accordance with the settings for each channel.



Downloaded from Arrow.com.

## 17.6 Interrupt Sources

### 17.6.1 Interrupt Sources and DTC Activation

There are three interrupt sources for TMRn: CMIAm, CMIBm, and OVIm. Their interrupt sources and priorities are listed in table 17.6.

It is also possible to activate the DTC by means of CMIAm and CMIBm interrupts. The DMAC cannot be activated by the interrupt sources for TMRn.

**Table 17.6 TMR Interrupt Sources**

| Name | Interrupt Sources | Interrupt Status Flag* | DTC Activation | Priority |
|---|---|---|---|---|
| CMIA0 | TMR0.TCORA compare match | IR174.IR | Possible | High |
| CMIB0 | TMR0.TCORB compare match | IR175.IR | Possible | |
| OVI0 | TMR0.TCNT overflow | IR176.IR | Not possible | |
| CMIA1 | TMR1.TCORA compare match | IR177.IR | Possible | |
| CMIB1 | TMR1.TCORB compare match | IR178.IR | Possible | |
| OVI1 | TMR1.TCNT overflow | IR179.IR | Not possible | |
| CMIA2 | TMR2.TCORA compare match | IR180.IR | Possible | |
| CMIB2 | TMR2.TCORB compare match | IR181.IR | Possible | |
| OVI2 | TMR2.TCNT overflow | IR182.IR | Not possible | |
| CMIA3 | TMR3.TCORA compare match | IR183.IR | Possible | |
| CMIB3 | TMR3.TCORB compare match | IR184.IR | Possible | |
| OVI3 | TMR3.TCNT overflow | IR185.IR | Not possible | Low |

Note: * For details on the interrupt status flag, see section 10, Interrupt Control Unit (ICU).

### 17.6.2 A/D Converter Activation

The A/D converter can be activated by a compare match A for TMR0 or TMR2.

If the TMRn.TCSR.ADTE bit (n = 0, 2) is set to 1 (A/D converter start requests by compare match A are enabled), a request to start A/D conversion is sent to the A/D converter by the occurrence of a compare match A. If the 8-bit timer conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is started.

For the corresponding A/D converter units, see section 23, A/D Converter.



Downloaded from Arrow.com.

## 17.7 Usage Notes

### 17.7.1 Module Stop State Setting

Operation of the TMR can be disabled or enabled by using the module-stop control registers. The initial setting is for halting of TMR operation. Register access becomes possible after release from the module-stop state. For details, see section 8, Power-Down Modes.

### 17.7.2 Notes on Setting Cycle

If the compare match is selected for counter clear, TCNT is cleared at the last state in the cycle in which the value of TCNT matches with that of TCORA or TCORB. TCNT updates the counter value at this last state. Therefore, the counter frequency is obtained by the following formula (f: Counter frequency, ϕ: Operating frequency, N: TCORA and TCORB register setting value).

$$f = \phi / (N + 1)$$

### 17.7.3 Conflict between TCNT Write and Counter Clear

If a counter clear signal is generated concurrently with CPU write to TCNT, the clear takes priority and the write is not performed as shown in figure 17.13.

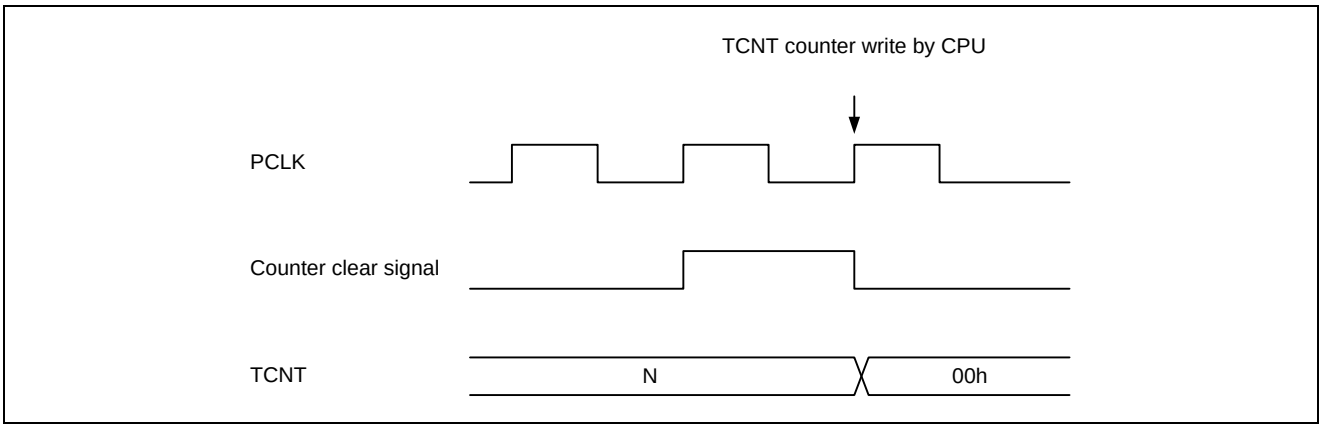



**Figure 17.13 Conflict between TCNT Write and Counter Clear**



Downloaded from Arrow.com.

### 17.7.4 Conflict between TCNT Write and Increment

Even if a counting-up signal is generated concurrently with CPU write to TCNT, the counting-up is not performed and the write takes priority as shown in figure 17.14.

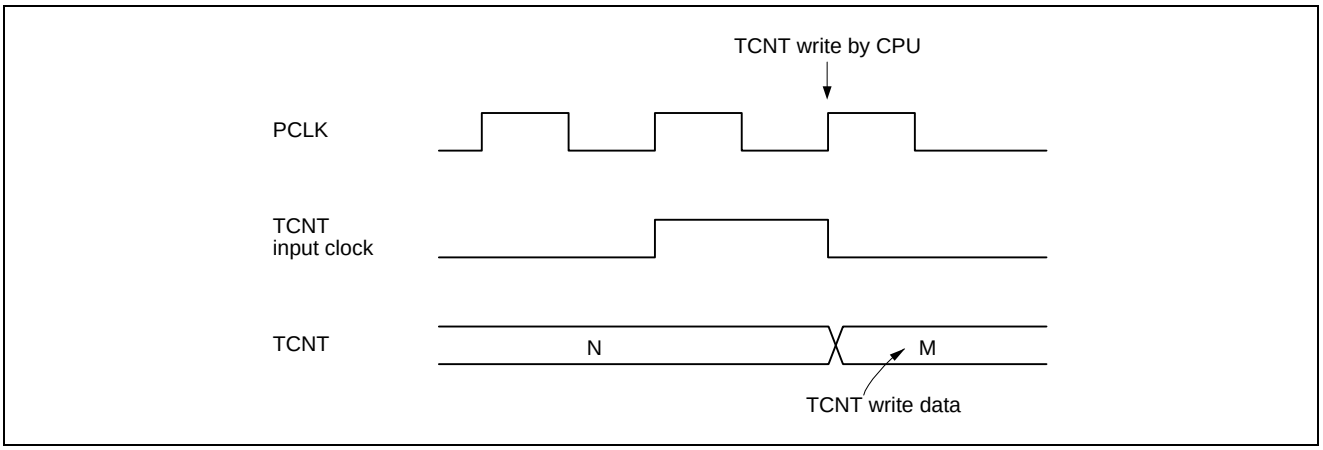



**Figure 17.14 Conflict between TCNT Write and Increment**



Downloaded from Arrow.com.

### 17.7.5 Conflict between TCORA or TCORB Write and Compare Match

Even if a compare match signal is generated simultaneously with CPU write to TCORA or TCORB, the write takes priority and the compare match signal is not set as shown in figure 17.15.

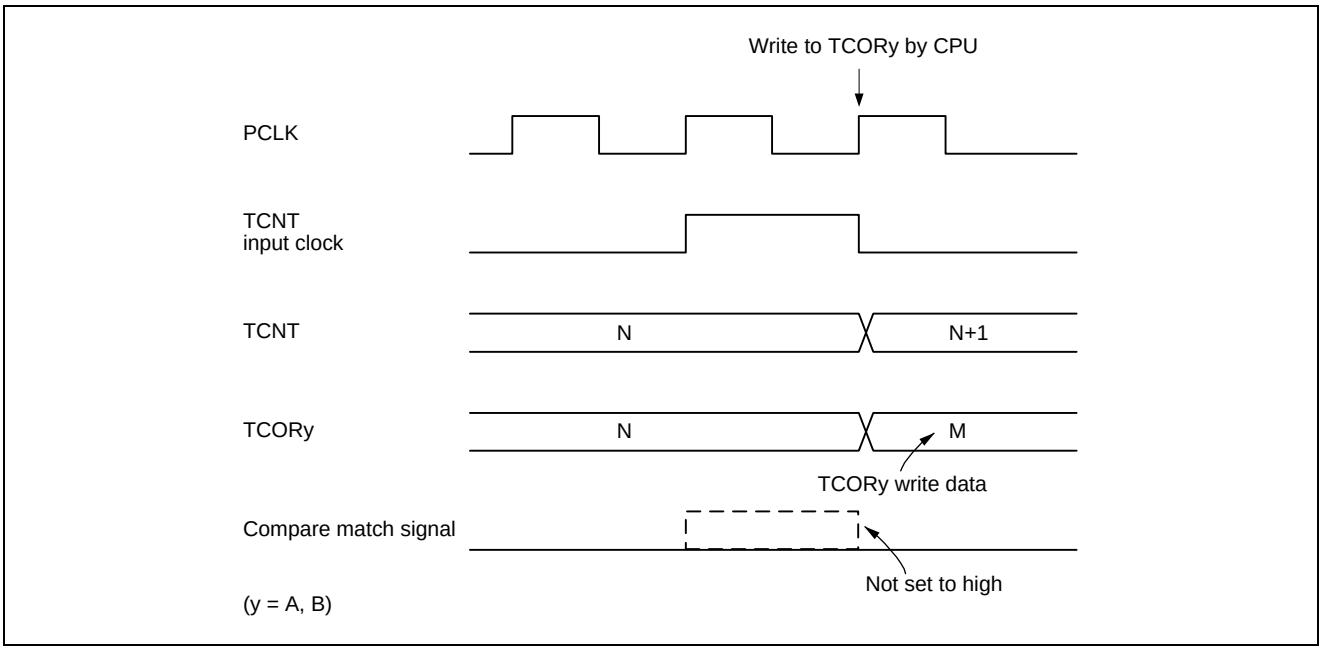



**Figure 17.15 Conflict between TCORA or TCORB Write and Compare Match**

### 17.7.6 Conflict between Compare Matches A and B

If compare match events A and B occur at the same time, the 8-bit timer operates in accordance with the priorities for the output statuses set for compare match A and compare match B, as shown in table 17.7.

**Table 17.7 Timer Output Priorities**

| Output Setting | Priority |
|---|---|
| Toggle output | High |
| High-output | ↑ |
| Low-output | |
| No change | Low |



Downloaded from Arrow.com.

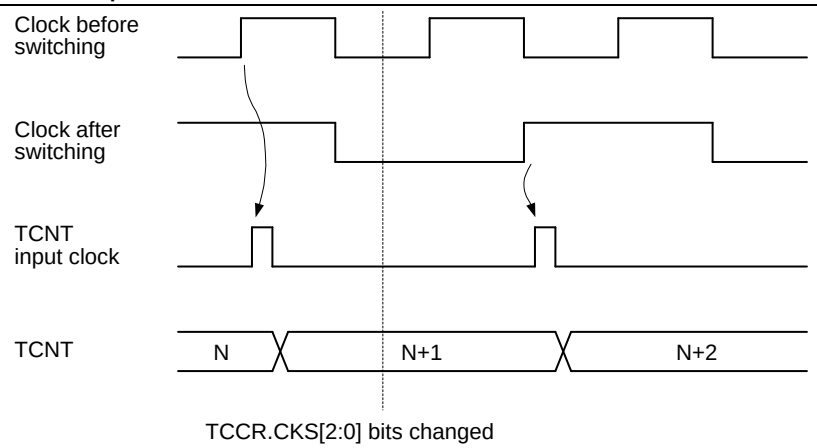
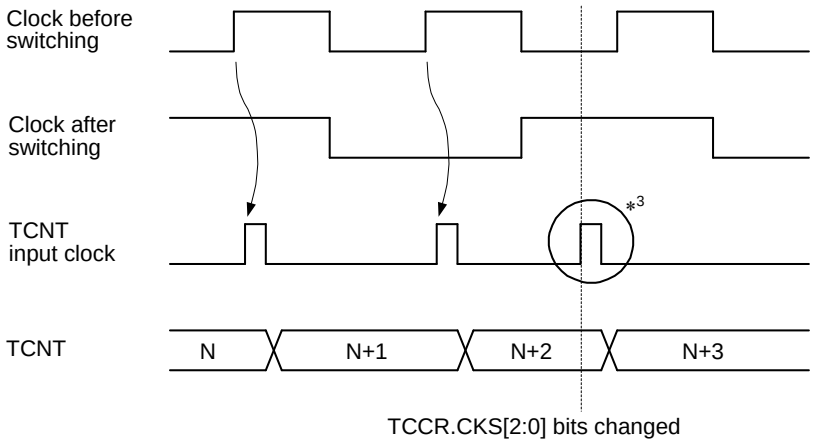
### 17.7.7 Switching of Internal Clocks and TCNT Operation

TCNT may be incremented erroneously depending on when the internal clock is switched. Table 17.8 shows the relationship between the timing at which the internal clock is switched (by writing to the TCCR.CKS[2:0] bits) and the operation of TCNT.

When TCNT clock is generated from an internal clock, the rising edge of the internal clock pulse are always monitored. If the signal levels of the clocks before and after switching change from low to high as shown in No. 2 in table 17.8, the change is considered as an edge. Therefore, a TCNT clock pulse is generated and TCNT is incremented.

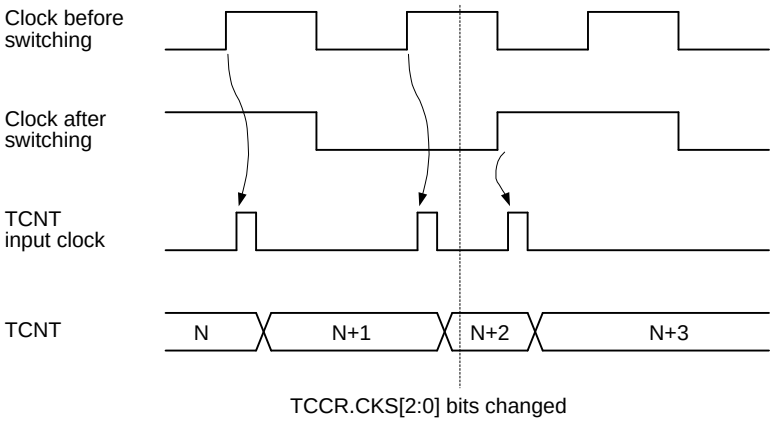
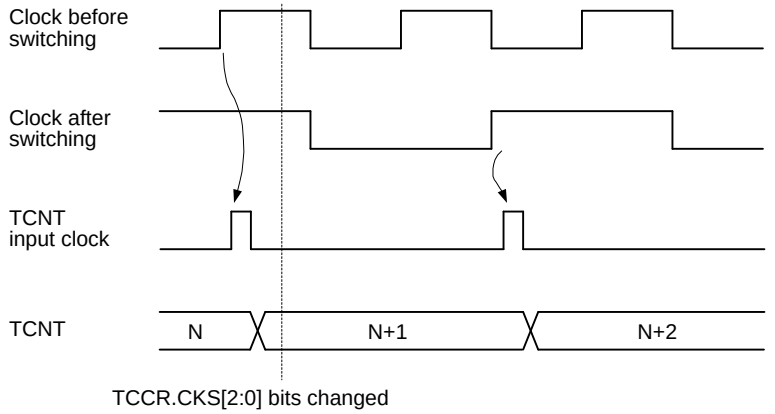
The erroneous increment of TCNT can also happen when switching between internal and external clocks.

**Table 17.8 Switching of Internal Clocks and TCNT Operation**

| Timing to Change the TCCR.CKS[2:0] Bits | TCNT Clock Operation |
|---|---|
| Switching from low to low*1 |  |
| Switching from low to high*2 |  |



Downloaded from Arrow.com.

| Timing to Change the TCCR.CKS[2:0] Bits | TCNT Clock Operation |
|---|---|
| Switching from high to low*4 |  |
| Switching from high to high |  |

Notes: 1. Includes switching from low to stop, and from stop to low.

2. Includes switching from stop to high.

3. Generated because the change of the signal levels is considered as an edge; TCNT is incremented.

4. Includes switching from high to stop.

## 17.7.8 Clock Source Setting with Cascaded Connection

If 16-bit count mode and compare match count mode are specified at the same time, input clocks for TMR0.TCNT and TMR1.TCNT (TMR2.TCNT and TMR3.TCNT) are not generated, and the counter stops. Do not specify 16-bit count mode and compare match count mode simultaneously.



Downloaded from Arrow.com.

# 18. Compare Match Timer (CMT)

The RX610 Group has two on-chip compare match timer (CMT) units (unit 0 and unit 1) each consisting of a two-channel 16-bit timer (i.e., a total of four channels). The CMT has a 16-bit counter, and can generate interrupts at set intervals.

## 18.1 Overview

Table 18.1 lists the specifications for the CMT.

Figure 18.1 shows a block diagram of the CMT (unit 0). A two-channel CMT constitutes a unit. Unit 0 and unit 1 are the same in terms of specifications.

**Table 18.1 Specifications of CMT**

| Item | Description |
|---|---|
| Count clock | • Four internal clocks<br>One clock from PCLK/8, PCLK/32, PCLK/128, and PCLK/512 can be selected individually for each channel. |
| Interrupt | A compare match interrupt can be requested individually for each channel. |
| Power-down function | Each unit can be placed in a module stop state. |

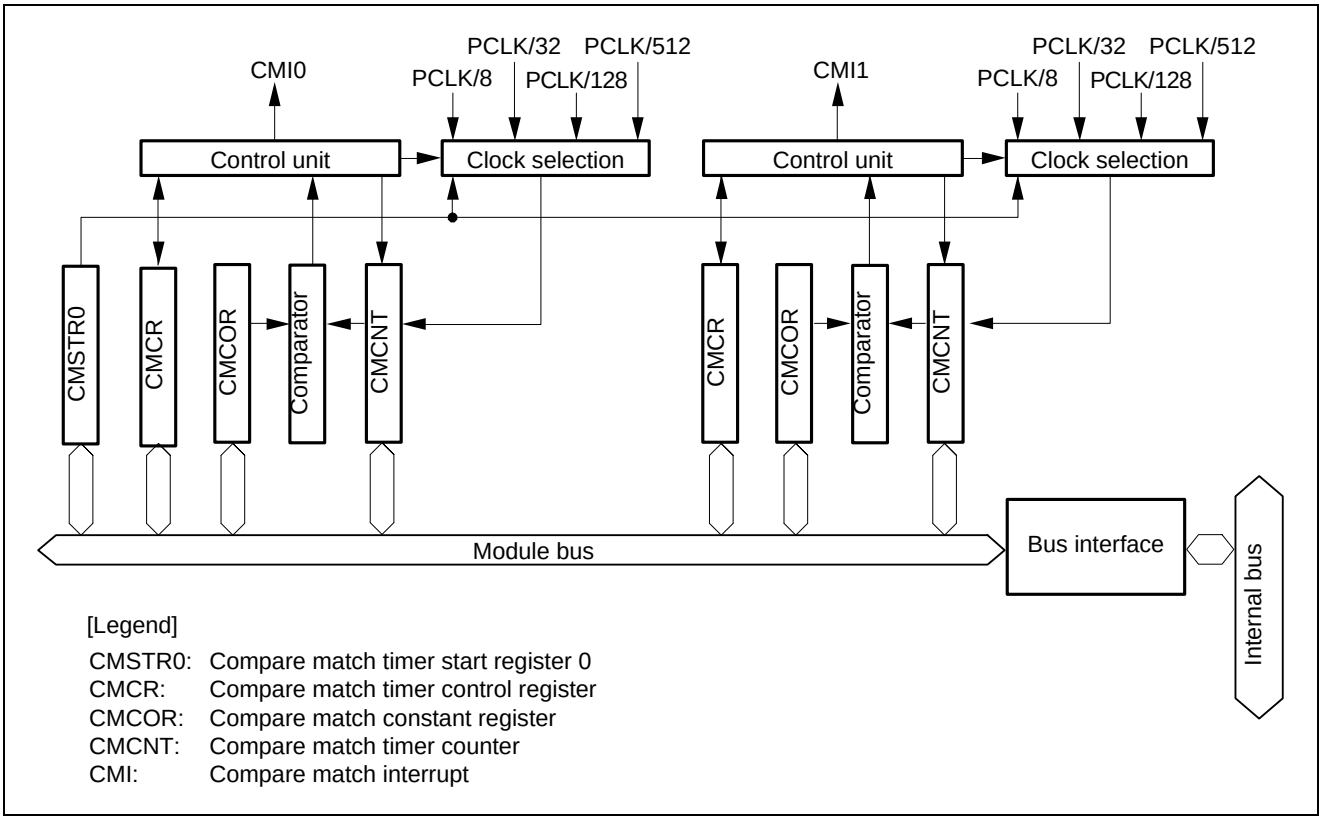



**Figure 18.1 Block Diagram of CMT (Unit 0)**



Downloaded from Arrow.com.

## 18.2 Register Descriptions

Table 18.2 lists the registers of the CMT.

**Table 18.2 List of CMT Registers**

| Unit | Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| Unit 0 | Common | Compare match timer start register 0 | CMSTR0 | 0000h | 0008 8000h | 16 |
| | CMT0 | Compare match timer control register | CMCR | 00x0h | 0008 8002h | 16 |
| | | Compare match timer counter | CMCNT | 0000h | 0008 8004h | 16 |
| | | Compare match timer constant register | CMCOR | FFFFh | 0008 8006h | 16 |
| | CMT1 | Compare match timer control register | CMCR | 00x0h | 0008 8008h | 16 |
| | | Compare match timer counter | CMCNT | 0000h | 0008 800Ah | 16 |
| | | Compare match timer constant register | CMCOR | FFFFh | 0008 800Ch | 16 |
| Unit 1 | Common | Compare match timer start register 1 | CMSTR1 | 0000h | 0008 8010h | 16 |
| | CMT2 | Compare match timer control register | CMCR | 00x0h | 0008 8012h | 16 |
| | | Compare match timer counter | CMCNT | 0000h | 0008 8014h | 16 |
| | | Compare match timer constant register | CMCOR | FFFFh | 0008 8016h | 16 |
| | CMT3 | Compare match timer control register | CMCR | 00x0h | 0008 8018h | 16 |
| | | Compare match timer counter | CMCNT | 0000h | 0008 801Ah | 16 |
| | | Compare match timer constant register | CMCOR | FFFFh | 0008 801Ch | 16 |



Downloaded from Arrow.com.

### 18.2.1 Compare Match Timer Start Register 0 (CMSTR0)

Address: 0008 8000h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | STR1 | STR0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | STR0 | Count Start 0 | 0: CMT0.CMCNT count is stopped<br>1: CMT0.CMCNT count is started | R/W |
| b1 | STR1 | Count Start 1 | 0: CMT1.CMCNT count is stopped<br>1: CMT1.CMCNT count is started | R/W |
| b15 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

CMSTR0 selects whether the CMT0.CMCNT or CMT1.CMCNT counter operates or is stopped.

**STR0 Bit (Count Start 0)**

The STR0 bit specifies whether CMT0.CMCNT operates or is stopped.

**STR1 Bit (Count Start 1)**

The STR1 bit specifies whether CMT1.CMCNT operates or is stopped.



Downloaded from Arrow.com.

## 18.2.2 Compare Match Timer Start Register 1 (CMSTR1)

Address: 0008 8010h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | STR3 | STR2 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | STR2 | Count Start 2 | 0: CMT2.CMCNT count is stopped<br>1: CMT2.CMCNT count is started | R/W |
| b1 | STR3 | Count Start 3 | 0: CMT3.CMCNT count is stopped<br>1: CMT3.CMCNT count is started | R/W |
| b15 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

CMSTR1 selects whether the CMT2.CMCNT or CMT3.CMCNT counter operates or is stopped.

**STR2 Bit (Count Start 2)**

The STR2 bit specifies whether CMT2.CMCNT operates or is stopped.

**STR3 Bit (Count Start 3)**

The STR3 bit specifies whether CMT3.CMCNT operates or is stopped.



Downloaded from Arrow.com.

### 18.2.3 Compare Match Timer Control Register (CMCR)

Addresses: CMT0.CMCR 0008 8002h, CMT1.CMCR 0008 8008h,
CMT2.CMCR 0008 8012h, CMT3.CMCR 0008 8018h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | CMIE | — | — | — | — | CKS[1:0] | |
| Value after reset: | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[Legend] x: Undefined

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | CKS[1:0] | Clock Select | b1 b0<br>0 0: PCLK/8<br>0 1: PCLK/32<br>1 0: PCLK/128<br>1 1: PCLK/512 | R/W |
| b5 to b2 | — | Reserved | These bits are always read as 0.<br>The write value should always be 0. | R/W |
| b6 | CMIE | Compare Match Interrupt Enable | 0: Compare match interrupt (CMIm) disabled<br>1: Compare match interrupt (CMIm) enabled | R/W |
| b7 | — | Reserved | This bit is always read as an undefined value. The write value should always be 1. | R/W |
| b15 to b8 | — | Reserved | These bits are always read as 0.<br>The write value should always be 0. | R/W |

CMCR sets the clock used for counting up.

#### CKS[1:0] Bits (Clock Select)

These bits select the clock to be input to CMCNT from four internal clocks obtained by dividing the peripheral clock (PCLK).

When the STRj (j = 0 to 3) bit in CMSTRy (y = 0 or 1) is set to 1, CMCNT starts counting up on the clock selected with bits CKS[1:0].

#### CMIE Bit (Compare Match Interrupt Enable)

The CMIE bit enables or disables compare match interrupt (CMIm) (m = 0 to 3) generation when CMCNT and CMCOR values match.



Downloaded from Arrow.com.

### 18.2.4 Compare Match Counter (CMCNT)

Addresses: CMT0.CMCNT 0008 8004h, CMT1.CMCNT 0008 800Ah,
CMT2.CMCNT 0008 8014h, CMT3.CMCNT 0008 801Ah

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CMCNT is a readable/writable up-counter to generate interrupt requests.

When an internal clock is selected by bits CKS[1:0] in CMCR and the STRj (j = 0 to 3) bit in CMSTRy (y = 0 or 1) is set to 1, CMCNT starts counting up using the selected clock.

When the value in CMCNT and the value in CMCOR match, CMCNT is cleared to 0000h. At the same time, a compare match interrupt (CMIm) (m = 0 to 3) is generated.

### 18.2.5 Compare Match Constant Register (CMCOR)

Addresses: CMT0.CMCOR 0008 8006h, CMT1.CMCOR 0008 800Ch,
CMT2.CMCOR 0008 8016h, CMT3.CMCOR 0008 801Ch

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

CMCOR sets the interval up to a compare match with CMCNT.



Downloaded from Arrow.com.

## 18.3 Operation

### 18.3.1 Periodic Count Operation

When an internal clock is selected by bits CKS[1:0] in CMCR and the STRj (j = 0 to 3) bit in CMSTRy (y = 0 or 1) is set to 1, CMCNT starts counting up using the selected clock.

When the value in CMCNT and the value in CMCOR match, CMCNT is cleared to 0000h. At the same time, a compare match interrupt (CMIm) (m = 0 to 3) is generated. CMCNT then starts counting up again from 0000h. Figure 18.2 shows the operation of the CMCNT counter.

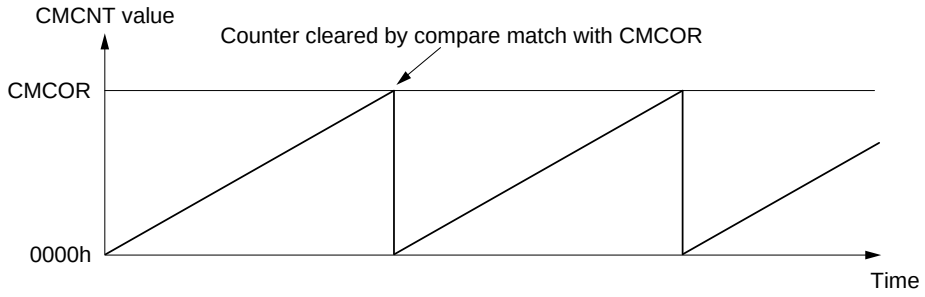



**Figure 18.2 CMCNT Counter Operation**

### 18.3.2 CMCNT Count Timing

One of four internal clocks (PCLK/8, PCLK/32, PCLK/128, and PCLK/512) obtained by dividing the peripheral clock (PCLK) can be selected with the CKS[1:0] bits in CMCSR. Figure 18.3 shows the timing of CMCNT.

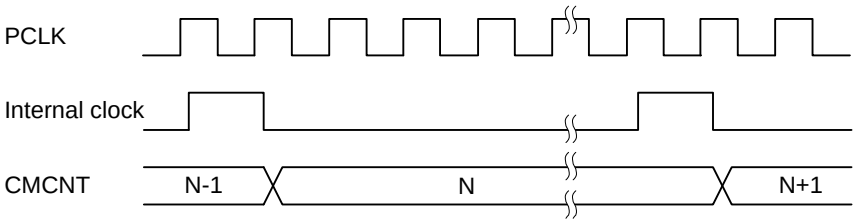



**Figure 18.3 CMCNT Count Timing**



Downloaded from Arrow.com.

## 18.4 Interrupts

### 18.4.1 Interrupt Sources

The CMT has channels and each of them to which a different vector address is allocated has a compare match interrupt (CMIm) (m = 0 to 3). When a compare match interrupt occurs, the corresponding interrupt request is output.

When the interrupt is used to activate a CPU interrupt, the priority of channels can be changed by the interrupt control unit settings. For details, see section 10, Interrupt Control Unit (ICU).

**Table 18.3 CMT Interrupt Sources**

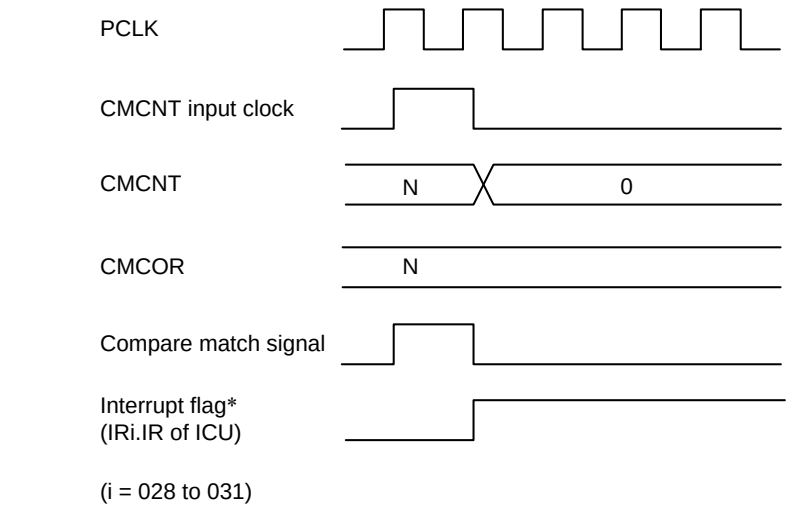
| Name | Interrupt Sources | Interrupt Status Flag | DTC Activation | DMAC Activation |
|---|---|---|---|---|
| CMI0 | Compare match between CMT0.CMCNT and CMT0.CMCOR | IR028.IR | Possible | Possible |
| CMI1 | Compare match between CMT1.CMCNT and CMT1.CMCOR | IR029.IR | Possible | Possible |
| CMI2 | Compare match between CMT2.CMCNT and CMT2.CMCOR | IR030.IR | Possible | Possible |
| CMI3 | Compare match between CMT3.CMCNT and CMT3.CMCOR | IR031.IR | Possible | Possible |

### 18.4.2 Timing of Compare Match Interrupt Generation

When CMCNT and CMCOR match, a compare match interrupt (CMIm) (m = 0 to 3) is generated.

A compare match signal is generated at the last state in which the values match (the timing when the CMCNT counter updates the matched count value). That is, after a match between CMCOR and CMCNT, the compare match signal is not generated until the next CMCNT counter clock input.

Figure 18.4 shows the timing of interrupt flag setting to 1.




Note: * For the corresponding interrupt vector numbers, see section 10, Interrupt Control Unit (ICU).

**Figure 18.4 Timing of Compare Match Interrupt Flag Setting to 1**



Downloaded from Arrow.com.

## 18.5 Usage Notes

### 18.5.1 Setting the Module Stop Function

The CMT can be enabled or disabled using the module stop control register. The CMT is disabled by default. The registers can be accessed by canceling the module stop state. For details, see section 8, Low Power Consumption.

### 18.5.2 Conflict between Write and Compare-Match Processes of CMCNT

When the compare match signal is generated while writing to CMCNT, clearing CMCNT has priority over writing to it. In this case, CMCNT is not written to. Figure 18.5 shows the timing to clear the CMCNT counter.

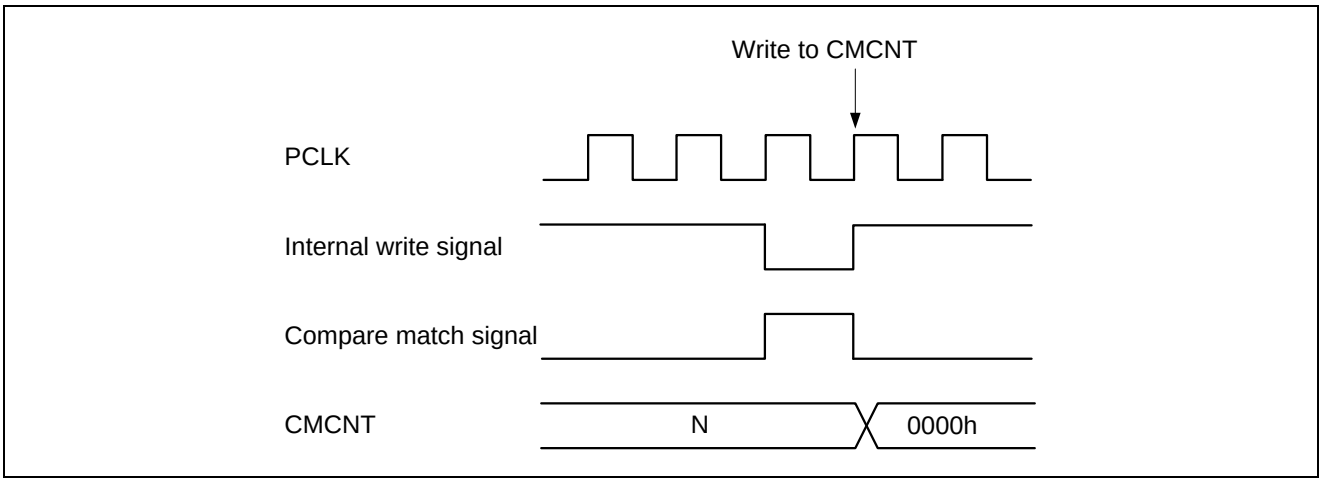



**Figure 18.5 Conflict between Write and Compare Match Processes of CMCNT**

### 18.5.3 Conflict between Write and Count-Up Processes of CMCNT

Even when the count-up occurs while writing to CMCNT, the writing has priority over the count-up. In this case, the count-up is not performed. Figure 18.6 shows the timing to write the CMCNT counter.

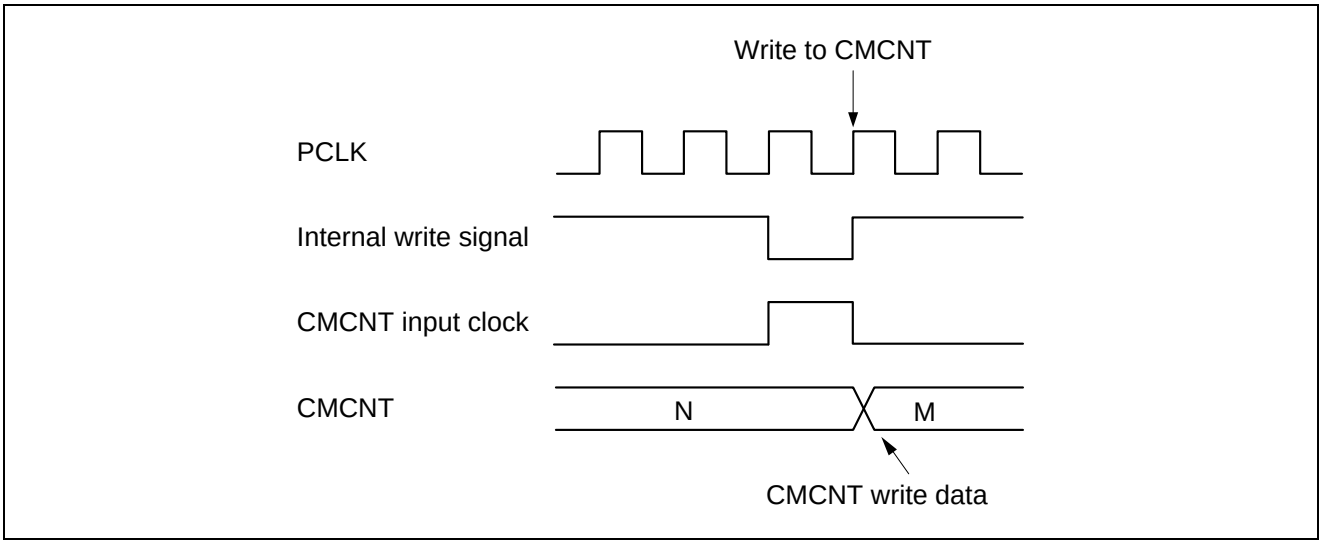



**Figure 18.6 Conflict between Write and Count-Up Processes of CMCNT**



Downloaded from Arrow.com.

### 18.5.4 Notes on Rewriting Compare Match Timer Control Register (CMCR)

When rewriting to the CMCR competes with generation of the compare match, writing to the CMCR is ignored. Therefore, it is necessary to check if data is written correctly by reading the CMCR after writing to the CMCR. If the data is not written correctly, once again the writing should be carried out to the CMCR. Because the read value of bit 7 in CMCR is not defined, care should to be taken while comparing it with the written data.

### 18.5.5 Notes on Compare Match Timer Counter (CMCNT) and Compare Match Constant Register (CMCOR)

Do not set the CMCNT and the CMCOR to the same value while the count operation of the CMCNT is stopped. If the CMCNT and the CMCOR are set to the same value with the CMCNT count operation halted, the compare match occurs irrespective of halted count operation. At this time, when the compare match interrupt enable bit (the CMIE bit in CMCR) has been enabled (set to 1), the compare match interrupt occurs. Despite of disabling/enabling of the compare match interrupt, if the compare match occurs due to matching of the value with the value of the CMCOR, the CMCNT is automatically cleared to 0000h.



Downloaded from Arrow.com.

# 19. Watchdog Timer (WDT)

The watchdog timer (WDT) is an 8-bit timer that outputs an overflow signal (WDTOVF#) if a system crash prevents the CPU from writing to the timer counter, thus allowing it to overflow. At the same time, the WDT can also generate an internal reset signal.

When this watchdog function is not needed, the WDT can be used as an interval timer. In interval timer operation, an interval timer interrupt is generated each time the counter overflows.

## 19.1 Overview

Table 19.1 lists the specifications of the WDT.

Figure 19.1 shows a block diagram of the WDT.

**Table 19.1 Specifications of WDT**

| Item | Specifications |
|---|---|
| Count clocks | PCLK/4, PCLK/64, PCLK/128, PCLK/512, PCLK/2048, PCLK/8192, PCLK/32768, and PCLK/131072 |
| Number of channels | 8 bits x 1 channel |
| Count clear | Write to TCNT |
| Operating modes | Switchable between watchdog timer mode and interval timer mode |
| Watchdog timer mode | Outputs a WDTOVF# signal when the counter overflows.<br>Selectable whether or not to internally reset the LSI at the same time. |
| Interval timer mode | Generates an interval timer interrupt (WOVI) when the counter overflows. |



Downloaded from Arrow.com.

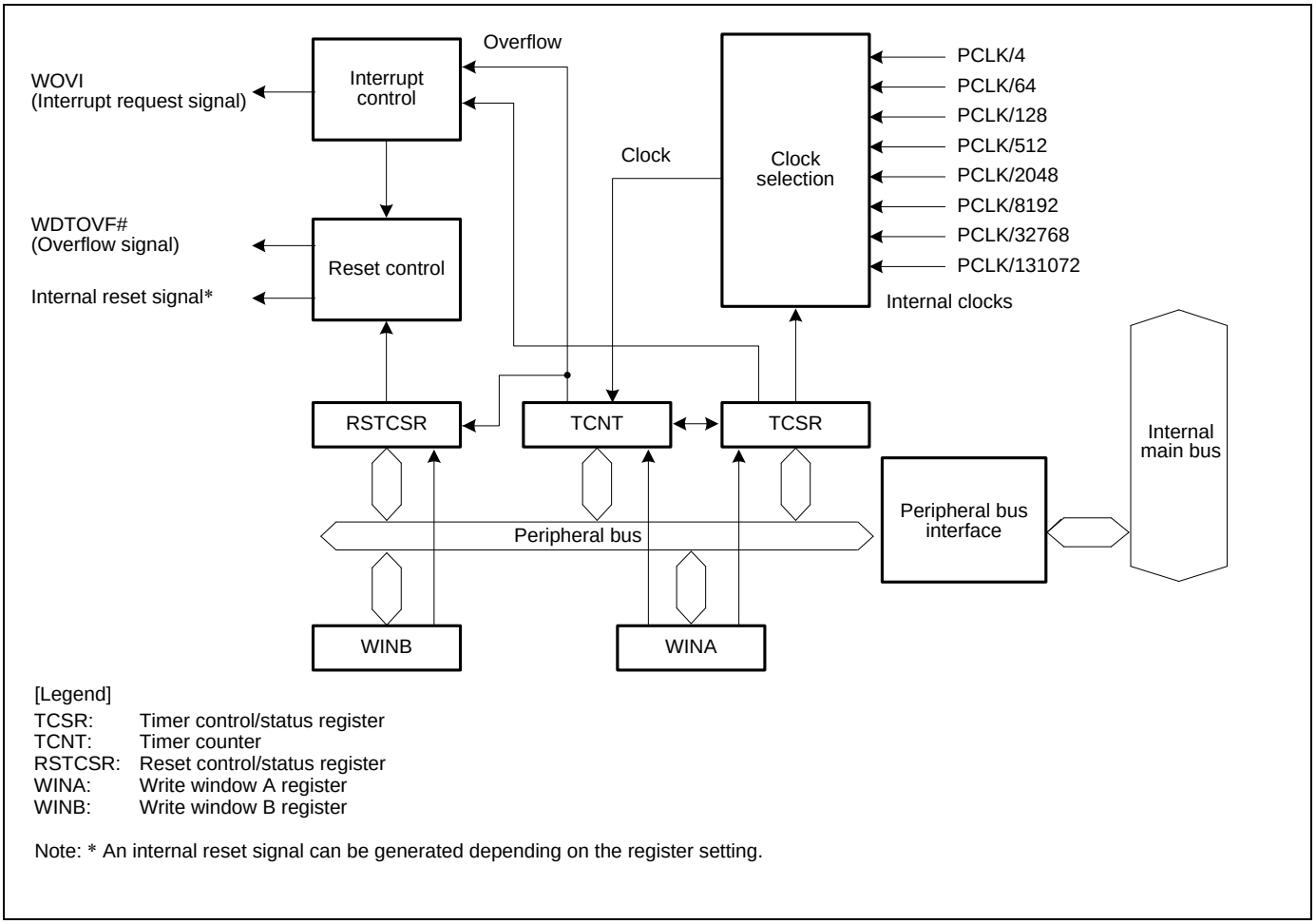



**Figure 19.1 Block Diagram of WDT**

Table 19.2 shows the input/output pin used for the WDT.

**Table 19.2 Pin Configuration**

| Pin Name | I/O | Description |
|---|---|---|
| WDTOVF# | Output | Outputs a counter overflow signal in watchdog timer mode. |



Downloaded from Arrow.com.

## 19.2 Register Descriptions

Table 19.3 lists the registers of the WDT.

**Table 19.3 WDT Registers**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Timer control/status register | TCSR | x8h | 0008 8028h*1 | 8 |
| Timer counter | TCNT | 00h | 0008 8029h*1 | 8 |
| Reset control/status register | RSTCSR | 1Fh | 0008 802Bh*1 | 8 |
| Write window A register | WINA | — | 0008 8028h*2 | 16 |
| Write window B register | WINB | — | 0008 802Ah*2 | 16 |

Notes: 1. Read-only register
2. Write-only register

### 19.2.1 Timer Counter (TCNT)

Address: 0008 8029h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

TCNT is an 8-bit up-counter for the internal clock.

TCNT is initialized to 00h when the TME bit in TCSR is set to 0.

To read this counter, use 8-bit access.

To write to this counter, write data to WINA in 16 bits.

For details, see section 19.5.1, Notes on Register Access.



Downloaded from Arrow.com.

## 19.2.2 Timer Control/Status Register (TCSR)

Address: 0008 8028h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | TMS | TME | — | — | CKS[2:0] | | |
| Value after reset: | x | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

[Legend] x: Undefined

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | CKS[2:0] | Clock Select | b2 b0<br>0 0 0: PCLK/4 (cycle: 20.4 μs)<br>0 0 1: PCLK/64 (cycle: 326.4 μs)<br>0 1 0: PCLK/128 (cycle: 652.8 μs)<br>0 1 1: PCLK/512 (cycle: 2.6 ms)<br>1 0 0: PCLK/2048 (cycle: 10.4 ms)<br>1 0 1: PCLK/8192 (cycle: 41.8 ms)<br>1 1 0: PCLK/32768 (cycle: 167.1 ms)<br>1 1 1: PCLK/131072 (cycle: 668.5 ms)<br>Note: The overflow cycle for PCLK = 50 MHz is indicated in parentheses. | R/W |
| b4, b3 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b5 | TME | Timer Enable | 0: TCNT stops counting and is initialized to 00h.<br>1: TCNT starts counting. | R/W |
| b6 | TMS | Timer Mode Select | 0: Interval timer mode<br>When TCNT overflows, an interval timer interrupt (WOVI) is requested.<br>1: Watchdog timer mode<br>When TCNT overflows, WDTOVF# is output. | R/W |
| b7 | — | Reserved | This bit is always read as an undefined value. The write value should always be 1. | R/W |

TCSR selects the clock source to be input to TCNT, and the timer mode.

To read this register, use 8-bit access.

To write to this register, write data to WINA in 16 bits.

For details, see section 19.5.1, Notes on Register Access.

### CKS[2:0] Bits (Clock Select)

These bits select the clock source to be input to TCNT.

### TME Bit (Timer Enable)

Selects whether TCNT starts or stops counting. When this bit is set to 1, TCNT starts counting. When this bit is cleared to 0, TCNT stops counting and is initialized to 00h.

### TMS Bit (Timer Select)

Selects whether the WDT is used as a watchdog timer or interval timer.



Downloaded from Arrow.com.

### 19.2.3 Reset Control/Status Register (RSTCSR)

Address: 0008 802Bh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | WOVF | RSTE | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b4 to b0 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |
| b5 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b6 | RSTE | Reset Enable | 0: The LSI is not reset internally when TCNT overflows in watchdog timer mode. (TCNT and TCSR of the WDT are reset.)<br>1: The LSI is internally reset when TCNT overflows in watchdog timer mode. | R/W |
| b7 | WOVF | Watchdog Timer Overflow Flag | 0: TCNT has not overflowed in watchdog timer mode.<br>1: TCNT has overflowed in watchdog timer mode. | R/(W)* |

Note: * Only 0 can be written to this bit.

RSTCSR controls the generation of the internal reset signal when TCNT overflows, and selects the type of internal reset signal.

RSTCSR is initialized to 1Fh by a reset signal from the RES# pin or a deep software standby reset, but not by the WDT internal reset signal caused by a WDT overflow.

To read this register, use 8-bit access.

To write to this register, write data to WINB in 16 bits.

For details, see section 19.5.1, Notes on Register Access.

#### RSTE Bit (Reset Enable)

Selects whether or not this LSI is internally reset when TCNT overflows in watchdog timer mode.

#### WOVF Flag (Watchdog Timer Overflow Flag)

Indicates that TCNT overflows in watchdog timer mode. This bit cannot be set to 1 in interval timer mode.

[Setting condition]

- When TCNT overflows (changed form FFh to 00h) in watchdog timer mode

[Clearing condition]

- Reading RSTCSR when WOVF = 1, and then writing 0 to WOVF



Downloaded from Arrow.com.

### 19.2.4 Write Window A Register (WINA)

Address: 0008 8028h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Value after reset: | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

WINA is a write-only register for writing to TCNT and TCSR.

The writing method varies between TCNT and TCSR. For details, see section 19.5.1, Notes on Register Access.

To write to this register, use 16-bit access.

### 19.2.5 Write Window B Register (WINB)

Address: 0008 802Ah

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Value after reset: | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

WINB is a write-only register for writing to RSTCSR.

The writing method varies between writing 0 to the WOVF flag in RSTCSR and writing the RSTE bit in RSTCSR. For details, see section 19.5.1, Notes on Register Access. To write to this register, use 16-bit access.



Downloaded from Arrow.com.

## 19.3 Operation

### 19.3.1 Watchdog Timer Mode

To use the WDT in watchdog timer mode, set both the TMS bit and TME bit in TCSR to 1.

During watchdog timer operation, if TCNT overflows without being rewritten because of a system crash or another error, the WDTOVF# signal is output. This ensures that TCNT does not overflow while the system is operating normally. Software must prevent TCNT overflows by rewriting the TCNT value (normally 00h is written) before overflow occurs. This WDTOVF# signal can be used to reset the LSI internally in watchdog timer mode.

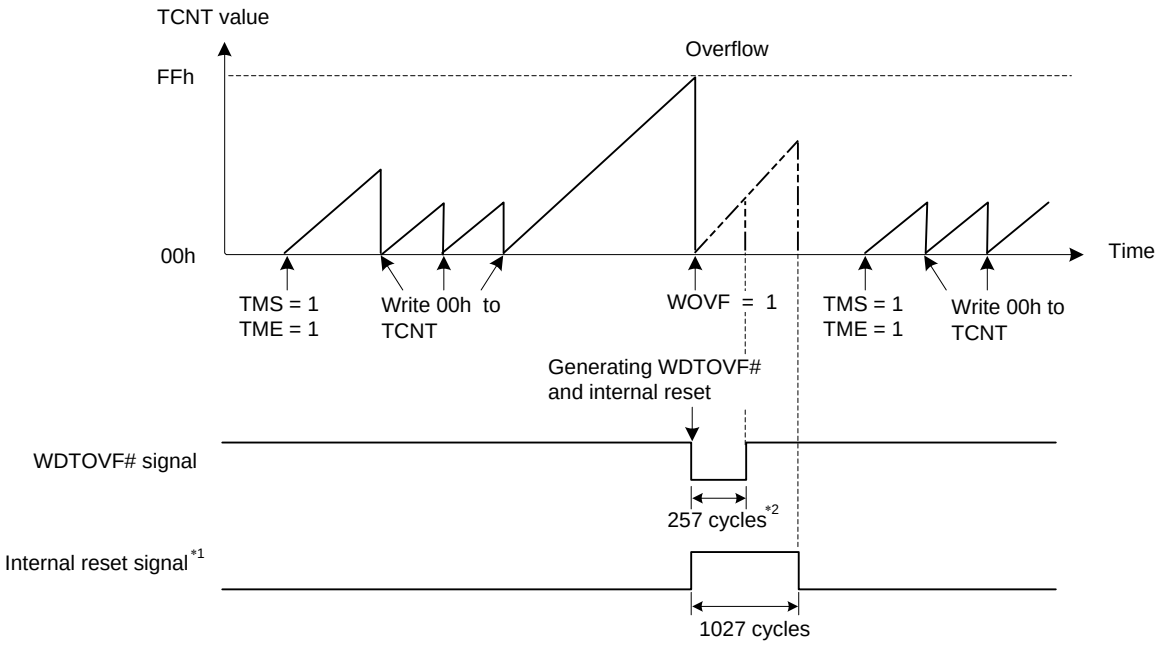
If TCNT overflows when the RSTE bit in RSTCSR is set to 1, a signal that resets this LSI internally is generated as the same time as the WDTOVF# signal. If a reset caused by a signal input to the RES# pin occurs at the same time as a reset caused by a WDT overflow, the RES# pin reset has priority and the WOVF flag in RSTCSR is cleared to 0.

The WDTOVF# signal is output for 257 cycles of PCLK when RSTE = 1, and for 256 cycles of PCLK when RSTE = 0. The internal reset signal is output for 1027 cycles of PCLK.

When RSTE = 1, an internal reset signal is generated. Since the system clock control register (SCKCR) is initialized, the multiplication ratio of PCLK becomes the initial value.

When RSTE = 0, an internal reset signal is not generated. Neither SCKCR nor the multiplication ratio of PCLK is changed.

When TCNT overflows in watchdog timer mode, the WOVF flag is set to 1.




Notes: 1. An internal reset signal is generated if TCNT overflows when the RSTE bit in RSTCSR is set to 1.
2. The WDTOVF# signal is output for 256 cycles when RSTE bit in RSTCSR is set to 0.

**Figure 19.2 Operation in Watchdog Timer Mode**



Downloaded from Arrow.com.

### 19.3.2 Interval Timer Mode

To use the WDT as an interval timer, set the TMS bit to 0 and the TME bit to 1 in TCSR.

When the WDT is used as an interval timer, an interval timer interrupt (WOVI) is generated each time TCNT overflows. Therefore, an interrupt can be generated at intervals.

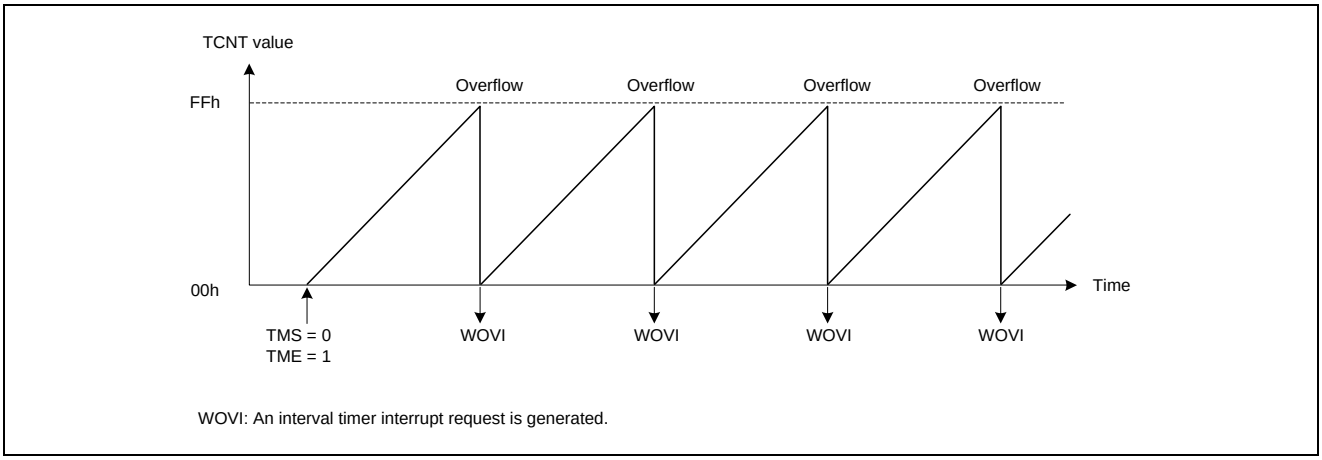



**Figure 19.3 Operation in Interval Timer Mode**

## 19.4 Interrupt Source

During interval timer mode operation, a TCNT overflow generates an interval timer interrupt (WOVI). For details, see section 10, Interrupt Control Unit (ICU).

**Table 19.4 WDT Interrupt Source**

| Name | Interrupt Source | Interrupt Status Flag | DTC Activation | DMAC Activation |
|---|---|---|---|---|
| WOVI | TCNT overflow | IR096.IR | Not possible | Not possible |



Downloaded from Arrow.com.

## 19.5 Usage Notes

### 19.5.1 Notes on Register Access

The watchdog timer's TCNT, TCSR, and RSTCSR differ from other registers in being more difficult to write to.

#### (1) Writing to TCNT Counter, TCSR Register, and RSTCSR Register

When writing to TCNT and TCSR, be sure to use a word transfer instruction to the write window A register (WINA) (00088028h). For writing, TCNT and TCSR are assigned to the same address. Accordingly, perform data transfer as shown in figure 19.4.

To write to TCNT, set 5Ah in the upper byte and data in the lower byte, and perform data transfer to TCNT.

To write to TCSR, set A5h in the upper byte and data in the lower byte, and perform data transfer to TCSR.

When writing to RSTCSR, use a word transfer instruction to the write window B register (WINB) (0008802Ah).

The method of writing 0 to the WOVF flag in RSTCSR differs from that of writing to the RSTE bit in RSTCSR.

To write 0 to the WOVF flag, set A5h in the upper byte and 00h in the lower byte and use 16-bit access, as shown in figure 19.4. This has no effect on the RSTE bit.

To write to the RSTE bit, set A5h in the upper byte and data to be written to RSTCSR in the lower byte and use 16-bit access, as shown in figure 19.4. This has no effect on the WOVF flag.

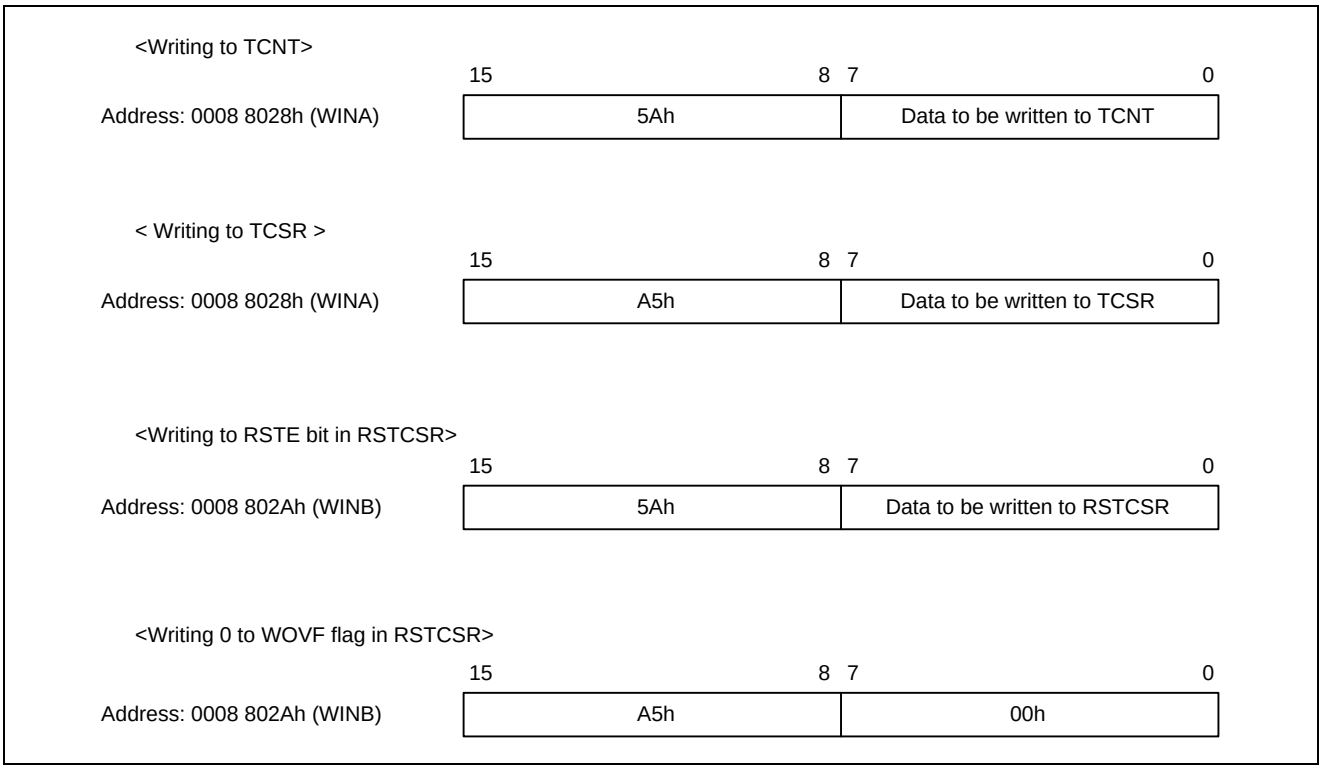



**Figure 19.4 Writing to TCNT, TCSR, and RSTCSR**



Downloaded from Arrow.com.

### (2) Reading from TCNT Counter, TCSR Register, and RSTCSR Register

These counter and registers can be read from in the same way as other registers.

TCSR is assigned to address 00088028h, TCNT to address 00088029h, and RSTCSR to address 0008802Ah. For reading, use 8-bit access.

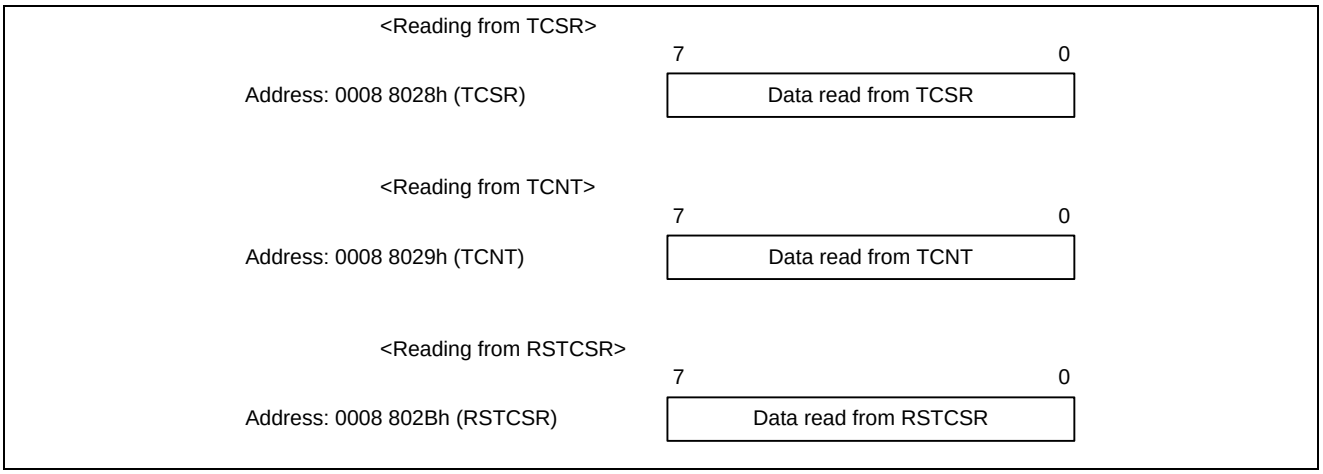



**Figure 19.5 Reading from TCNT, TCSR, and RSTCSR**

## 19.5.2 Conflict between Timer Counter (TCNT) Write and Increment

If a TCNT clock pulse is generated during a TCNT write cycle, the write takes priority and the timer counter is not incremented. Figure 19.6 shows this operation.

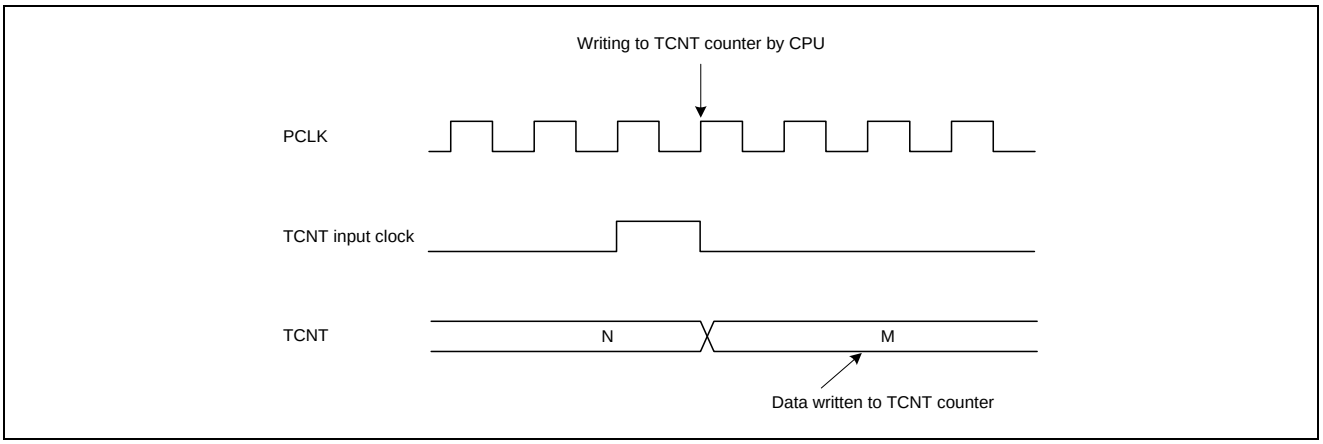



**Figure 19.6 Conflict between TCNT Write and Increment**

## 19.5.3 Changing Values of Bits CKS[2:0]

If bits CKS[2:0] bits in TCSR are written to while the watchdog timer is operating, errors could occur in the incrementation. The watchdog timer must be stopped (by clearing the TME bit in TCSR to 0) before the values of CKS[2:0] bits in TCSR are changed.



Downloaded from Arrow.com.

### 19.5.4 Switching between Watchdog Timer Mode and Interval Timer Mode

If the timer mode is switched from watchdog timer mode to interval timer mode while the watchdog timer is operating, errors could occur in the operation. The watchdog timer must be stopped (by clearing the TME bit in TCSR to 0) before switching the timer mode.

### 19.5.5 Internal Reset in Watchdog Timer Mode

This LSI is not reset internally if TCNT overflows while the RSTE bit in RSTCSR is cleared to 0 during watchdog time mode operation, but TCNT and the TCSR of the watchdog timer are reset.

TCNT, TCSR, and RSTCSR cannot be written to while the WDTOVF# signal is low. Also note that a read of the WOVF flag in RSTCSR is not recognized during this period. To clear the WOVF flag, therefore, read RSTCSR after the WDTOVF# signal goes high, then write 0 to the WOVF flag.

### 19.5.6 System Reset by WDTOVF# Signal

If the WDTOVF# signal is input to the RES# pin, this LSI will not be initialized correctly. Make sure that the WDTOVF# signal is not input logically to the RES# pin. To reset the entire system by means of the WDTOVF# signal, use a circuit like that shown in figure 19.7.

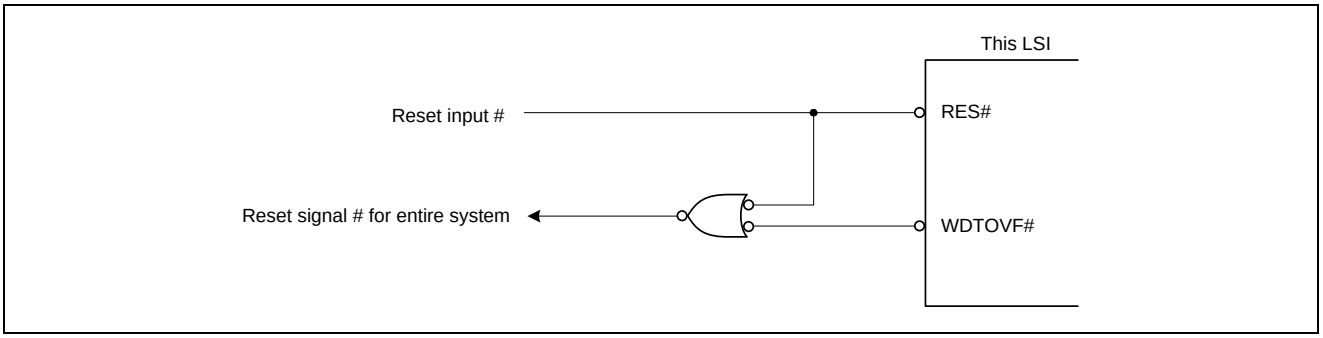



**Figure 19.7 Circuit for System Reset by WDTOVF# Signal (Example)**

### 19.5.7 Transition to Watchdog Timer Mode or Software Standby Mode

When the WDT operates in watchdog timer mode, a transition to software standby mode is not made even when the WAIT instruction is executed when the software standby bit (SSBY in SBYCR) in the standby control register is set to 1. Instead, a transition to sleep mode or all-module clock-stop mode is made.

To transit to software standby mode, the WAIT instruction must be executed after halting the watchdog timer (clearing the TME bit in TCSR to 0).

When the WDT operates in interval timer mode, a transition to software standby mode is made through execution of the WAIT instruction when the SSBY bit is set to 1. For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

# 20. Serial Communications Interface (SCI)

The RX610 Group has seven independent serial communications interface (SCI) units.

The SCI can handle both asynchronous and clock synchronous serial communications.

Asynchronous serial data communications can be carried out with standard asynchronous communications chips such as a Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communications Interface Adapter (ACIA).

The SCI also supports the smart card (IC card) interface supporting ISO/IEC 7816-3 (Identification Card) as an extended asynchronous communications mode.

## 20.1 Overview

Table 20.1 lists the specifications of the SCI and table 20.2 lists the functions of each SCI channel.

Figure 20.1 shows a block diagram of the SCI0 to SCI4. Figure 20.2 shows a block diagram of the SCI5 and SCI6.

**Table 20.1 Specifications of SCI**

| Item | | Specifications |
|---|---|---|
| Serial communications mode | | • Asynchronous<br>• Clock synchronous<br>• Smart card interface |
| Transfer speed | | Bit rate specifiable with on-chip baud rate generator. |
| Full-duplex communications | | Transmitter: Enables continuous transmission by double-buffering.<br>Receiver: Enables continuous reception by double-buffering. |
| Input/output pins | | See table 20.3. |
| Data transfer | | Selectable from LSB-first or MSB-first transfer |
| Interrupt sources | | Transmit-end, transmit-data-empty, receive-data-full, and receive error |
| Power-down function | | Module stop state can be set for each unit. |
| Asynchronous mode | Data length | 7 or 8 bits |
| | Transmission stop bit | 1 or 2 bits |
| | Parity | Even, odd, or none |
| | Receive error detection | Parity, overrun, and framing errors |
| | Break detection | Break can be detected by reading RxDn (n = 0 to 6) pin level directly in case of a framing error |
| | Clock source | Selectable from internal or external clock<br>Enables transfer rate clock input from TMR (SCI5 and SCI6) |
| Clock synchronous mode | Data length | 8 bits |
| | Receive error detection | Overrun errors |
| Smart card interface mode | Error processing | An error signal can be automatically transmitted on detection of a parity error during reception |
| | | Data can be automatically re-transmitted on receiving an error signal during transmission |
| | Data type | Both direct convention and inverse convention are supported. |



Downloaded from Arrow.com.

**Table 20.2 Function List of SCI Channels**

| Item | SCI0 to SCI4 | SCI5 and SCI6 |
|---|---|---|
| Asynchronous mode | Possible | Possible |
| Clock synchronous mode | Possible | Possible |
| Smart card interface mode | Possible | Possible |
| TMR clock input | Not possible | Possible |

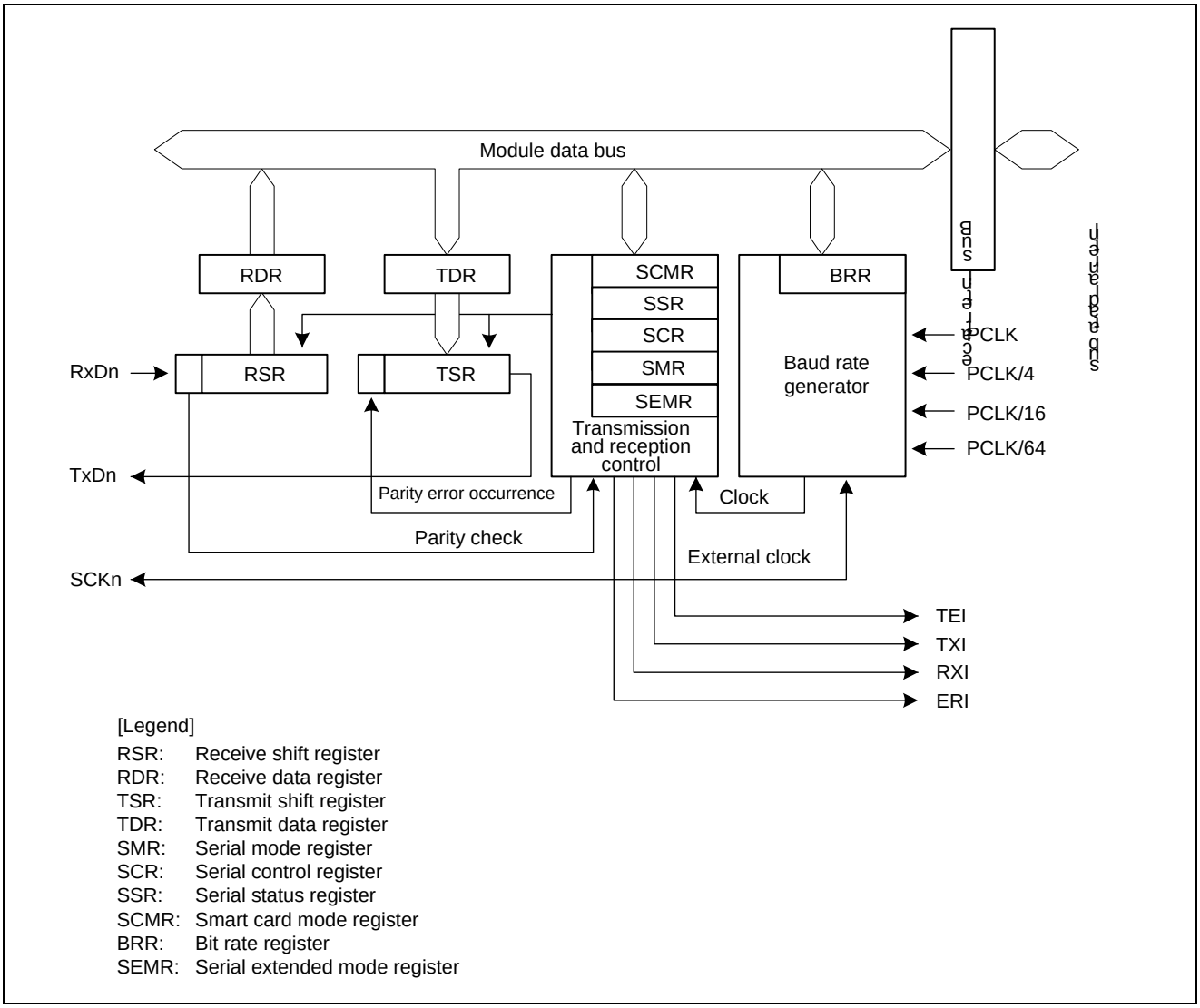



**Figure 20.1 Block Diagram of SCI0 to SCI4**



Downloaded from Arrow.com.

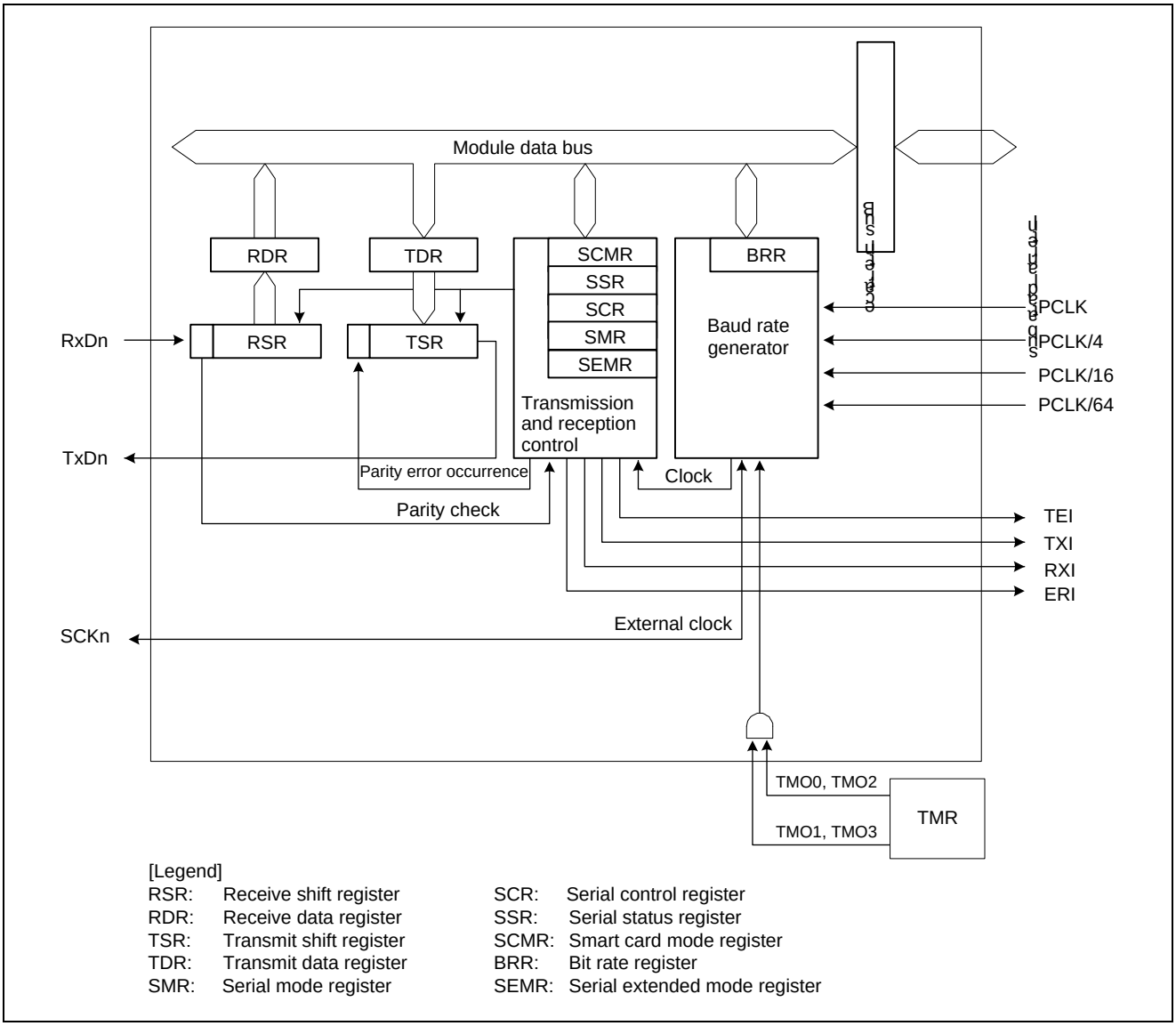



**Figure 20.2 Block Diagram of SCI5 and SCI6**



Downloaded from Arrow.com.

Table 20.3 lists the pin configuration of the SCI.

**Table 20.3 Pin Configuration of SCI**

| Channel | Pin Name | I/O | Function |
|---|---|---|---|
| SCI0 | SCK0 | I/O | SCI0 clock input/output |
| | RxD0 | Input | SCI0 receive data input |
| | TxD0 | Output | SCI0 transmit data output |
| SCI1 | SCK1 | I/O | SCI1 clock input/output |
| | RxD1 | Input | SCI1 receive data input |
| | TxD1 | Output | SCI transmit data output |
| SCI2 | SCK2 | I/O | SCI2 clock input/output |
| | RxD2 | Input | SCI2 receive data input |
| | TxD2 | Output | SCI2 transmit data output |
| SCI3 | SCK3 | I/O | SCI3 clock input/output |
| | RxD3 | Input | SCI3 receive data input |
| | TxD3 | Output | SCI3 transmit data output |
| SCI4 | SCK4 | I/O | SCI4 clock input/output |
| | RxD4 | Input | SCI receive data input |
| | TxD4 | Output | SCI4 transmit data output |
| SCI5 | SCK5 | I/O | SCI5 clock input/output |
| | RxD5 | Input | SCI5 receive data input |
| | TxD5 | Output | SCI5 transmit data output |
| SCI6 | SCK6 | I/O | SCI6 clock input/output |
| | RxD6 | Input | SCI6 receive data input |
| | TxD6 | Output | SCI6 transmit data output |



Downloaded from Arrow.com.

## 20.2 Register Descriptions

Table 20.4 lists the registers of the SCI.

**Table 20.4 Registers of SCI**

| Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| SCI0 | Serial mode register | SMR | 00h | 0008 8240h | 8 |
| | Bit rate register | BRR | FFh | 0008 8241h | 8 |
| | Serial control register | SCR | 0xh | 0008 8242h | 8 |
| | Transmit data register | TDR | FFh | 0008 8243h | 8 |
| | Serial status register | SSR | 84h | 0008 8244h | 8 |
| | Receive data register | RDR | 00h | 0008 8245h | 8 |
| | Smart card mode register | SCMR | F2h | 0008 8246h | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 8247h | 8 |
| SCI1 | Serial mode register | SMR | 00h | 0008 8248h | 8 |
| | Bit rate register | BRR | FFh | 0008 8249h | 8 |
| | Serial control register | SCR | 0xh | 0008 824Ah | 8 |
| | Transmit data register | TDR | FFh | 0008 824Bh | 8 |
| | Serial status register | SSR | 84h | 0008 824Ch | 8 |
| | Receive data register | RDR | 00h | 0008 824Dh | 8 |
| | Smart card mode register | SCMR | F2h | 0008 824Eh | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 824Fh | 8 |
| SCI2 | Serial mode register | SMR | 00h | 0008 8250h | 8 |
| | Bit rate register | BRR | FFh | 0008 8251h | 8 |
| | Serial control register | SCR | 0xh | 0008 8252h | 8 |
| | Transmit data register | TDR | FFh | 0008 8253h | 8 |
| | Serial status register | SSR | 84h | 0008 8254h | 8 |
| | Receive data register | RDR | 00h | 0008 8255h | 8 |
| | Smart card mode register | SCMR | F2h | 0008 8256h | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 8257h | 8 |
| SCI3 | Serial mode register | SMR | 00h | 0008 8258h | 8 |
| | Bit rate register | BRR | FFh | 0008 8259h | 8 |
| | Serial control register | SCR | 0xh | 0008 825Ah | 8 |
| | Transmit data register | TDR | FFh | 0008 825Bh | 8 |
| | Serial status register | SSR | 84h | 0008 825Ch | 8 |
| | Receive data register | RDR | 00h | 0008 825Dh | 8 |
| | Smart card mode register | SCMR | F2h | 0008 825Eh | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 825Fh | 8 |
| SCI4 | Serial mode register | SMR | 00h | 0008 8260h | 8 |
| | Bit rate register | BRR | FFh | 0008 8261h | 8 |
| | Serial control register | SCR | 0xh | 0008 8262h | 8 |
| | Transmit data register | TDR | FFh | 0008 8263h | 8 |
| | Serial status register | SSR | 84h | 0008 8264h | 8 |
| | Receive data register | RDR | 00h | 0008 8265h | 8 |
| | Smart card mode register | SCMR | F2h | 0008 8266h | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 8267h | 8 |
| **Channel** | **Register Name** | **Symbol** | **Value after Reset** | **Address** | **Access Size** |
| SCI5 | Serial mode register | SMR | 00h | 0008 8268h | 8 |



Downloaded from Arrow.com.

| | | | | | |
|---|---|---|---|---|---|
| | Bit rate register | BRR | FFh | 0008 8269h | 8 |
| | Serial control register | SCR | 0xh | 0008 826Ah | 8 |
| | Transmit data register | TDR | FFh | 0008 826Bh | 8 |
| | Serial status register | SSR | 84h | 0008 826Ch | 8 |
| | Receive data register | RDR | 00h | 0008 826Dh | 8 |
| | Smart card mode register | SCMR | F2h | 0008 826Eh | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 826Fh | 8 |
| SCI6 | Serial mode register | SMR | 00h | 0008 8270h | 8 |
| | Bit rate register | BRR | FFh | 0008 8271h | 8 |
| | Serial control register | SCR | 0xh | 0008 8272h | 8 |
| | Transmit data register | TDR | FFh | 0008 8273h | 8 |
| | Serial status register | SSR | 84h | 0008 8274h | 8 |
| | Receive data register | RDR | 00h | 0008 8275h | 8 |
| | Smart card mode register | SCMR | F2h | 0008 8276h | 8 |
| | Serial extended mode register | SEMR | 00h | 0008 8277h | 8 |



Downloaded from Arrow.com.

### 20.2.1 Receive Shift Register (RSR)

RSR is a shift register which is used to receive serial data input from the RxDn pin and converts it into parallel data.

When one frame of data has been received, it is transferred to RDR automatically.

RSR cannot be directly accessed by the CPU.

### 20.2.2 Receive Data Register (RDR)

Addresses: SCI0.RDR 0008 8245h, SCI1.RDR 0008 824Dh, SCI2.RDR 0008 8255h, SCI3.RDR 0008 25Dh
SCI4.RDR 0008 8265h, SCI5.RDR 0008 826Dh, SCI6.RDR 0008 8275h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

RDR is an 8-bit register that stores receive data.

When the SCI has received one frame of serial data, it transfers the received serial data from RSR to RDR where it is stored. This allows RSR to receive the next data.

Since RSR and RDR function as a double buffer in this way, continuous receive operations can be performed.

Read RDR only once after a receive data full interrupt (RXI) has occurred. Note that if next one frame of data is received before reading receive data from RDR, an overrun error occurs.

RDR cannot be written to by the CPU.

### 20.2.3 Transmit Data Register (TDR)

Addresses: SCI0.TDR 0008 8243h, SCI1.TDR 0008 824Bh, SCI2.TDR 0008 8253h, SCI3.TDR 0008 25Bh
SCI4.TDR 0008 8263h, SCI5.TDR 0008 826Bh, SCI6.TDR 0008 8273h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

TDR is an 8-bit register that stores transmit data.

When the SCI detects that TSR is empty, it transfers the transmit data written in TDR to TSR and starts transmission.

The double-buffered structures of TDR and TSR enable continuous serial transmission. If the next transmit data has already been written to TDR when one frame of data is transmitted, the SCI transfers the written data to TSR to continue transmission.

The CPU is able to read from or write to TDR at any time. Only write data for transmission to TDR once after each instance of the transmit data empty interrupt (TXI).



Downloaded from Arrow.com.

### 20.2.4 Transmit Shift Register (TSR)

TSR is a shift register that transmits serial data.

To perform serial data transmission, the SCI first automatically transfers transmit data from TDR to TSR, and then sends the data to the TxDn pin.

TSR cannot be directly accessed by the CPU.

### 20.2.5 Serial Mode Register (SMR)

Note: Some bits in SMR have different functions in serial communications interface mode and smart card interface mode.

#### (1) Serial Communications Interface Mode (SMIF in SCMR = 0)

Addresses: SCI0.SMR 0008 8240h, SCI1.SMR 0008 8248h, SCI2.SMR 0008 8250h, SCI3.SMR 0008 8258h
SCI4.SMR 0008 8260h, SCI5.SMR 0008 8268h, SCI6.SMR 0008 8270h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | CM | CHR | PE | PM | STOP | – | CKS[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b1, b0 | CKS[1:0] | Clock Select | b1 b0<br>00: PCLK clock (n = 0)*1<br>01: PCLK/4 clock (n = 1)*1<br>10: PCLK/16 clock (n = 2)*1<br>11: PCLK/64 clock (n = 3)*1 | R/W*4 |
| b2 | — | Reserved | This bit is read as 0. The write value should be 0. | R/W*4 |
| b3 | STOP | Stop Bit Length | (Valid only in asynchronous mode)<br>0: 1 stop bit<br>1: 2 stop bits | R/W*4 |
| b4 | PM | Parity Mode | (Valid only when the PE bit is 1 in asynchronous mode)<br>0: Selects even parity<br>1: Selects odd parity | R/W*4 |
| b5 | PE | Parity Enable | (Valid only in asynchronous mode)<br>• When transmitting<br>0: Parity bit addition is not performed<br>1: The parity bit is added<br>• When receiving<br>0: Parity bit checking is not performed<br>1: The parity bit is checked | R/W*4 |
| b6 | CHR | Character Length | (Valid only in asynchronous mode)<br>0: Selects 8 bits as the data length*2<br>1: Selects 7 bits as the data length*3 | R/W*4 |
| b7 | CM | Communications Mode | 0: Asynchronous mode<br>1: Clock synchronous mode | R/W*4 |

Notes: 1. n is the decimal notation of the value of n in BRR (see section 20.2.9, Bit Rate Register (BRR)).
2. In clock synchronous mode, this bit setting is invalid and a fixed data length of 8 bits is used.
3. LSB-first is fixed and the MSB (bit 7) in TDR is not transmitted in transmission.
4. Writable only when TE in SCR = 0 and RE in SCR = 0 (both serial transmission and reception are disabled).



Downloaded from Arrow.com.

SMR is used to set the SCI's serial transfer format and select the baud rate generator clock source.

#### CKS[1:0] Bits (Clock Select)

These bits select the clock source for the baud rate generator.

For the relation between the settings of these bits and the baud rate, see section 20.2.9, Bit Rate Register (BRR).

#### STOP Bit (Stop Bit Length)

Selects the stop bit length in transmission.

In reception, only the first stop bit is checked. If the second stop bit is 0, it is treated as the start bit of the next transmit frame.

#### PM Bit (Parity Mode)

Selects the parity mode (even or odd) for transmission and reception.

#### PE Bit (Parity Enable)

When this bit is set to 1, the parity bit is added to transmit data, and the parity bit is checked in reception.

#### CHR Bit (Character Length)

Selects the data length for transmission and reception.

In clock synchronous mode, a fixed data length of 8 bits is used.

#### CM Bit (Communications Mode)

Selects asynchronous or clock synchronous mode.



Downloaded from Arrow.com.

## (2) Smart Card Interface Mode (SMIF in SCMR = 1)

Addresses: SCI0.SMR 0008 8240h, SCI1.SMR 0008 8248h, SCI2.SMR 0008 8250h, SCI3.SMR 0008 8258h
SCI4.SMR 0008 8260h, SCI5.SMR 0008 8268h, SCI6.SMR 0008 8270h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | GM | BLK | PE | PM | BCP[1:0] | | CKS[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b1, b0 | CKS[1:0] | Clock Select | b1 b0<br>00: PCLK clock (n = 0)*1<br>01: PCLK/4 clock (n = 1)*1<br>10: PCLK/16 clock (n = 2)*1<br>11: PCLK/64 clock (n = 3)*1 | R/W*3 |
| b3, b2 | BCP[1:0] | Base Clock Pulse | Selects the number of base clock cycles in combination with the BCP2 bit in SCMR.<br>Setting values in BCP2 bit in SCMR and BCP[1:0] bits in SMR:<br>BCP2 b3 b2<br>0 0 0: 93 clock cycles (S = 93)*2<br>0 0 1: 128 clock cycles (S = 128)*2<br>0 1 0: 186 clock cycles (S = 186)*2<br>0 1 1: 512 clock cycles (S = 512)*2<br>1 0 0: 32 clock cycles (S = 32)*2 (Initial value)<br>1 0 1: 64 clock cycles (S = 64)*2<br>1 1 0: 372 clock cycles (S = 372)*2<br>1 1 1: 256 clock cycles (S = 256)*2 | R/W*3 |
| b4 | PM | Parity Mode | (Valid only when the PE bit is 1 in asynchronous mode)<br>0: Selects even parity<br>1: Selects odd parity | R/W*3 |
| b5 | PE | Parity Enable | (Valid only in asynchronous mode)<br>When this bit is set to 1, the parity bit is added to transmit data before transmission, and the parity bit is checked in reception.<br>Set this bit to 1 in smart card interface mode. | R/W*3 |
| b6 | BLK | Block Transfer Mode | 0: Normal mode operation<br>1: Block transfer mode operation | R/W*3 |
| b7 | GM | GSM Mode | 0: Normal mode operation<br>1: GSM mode operation | R/W*3 |

Notes: 1. n is the decimal notation of the value of n in BRR (see section 20.2.9, Bit Rate Register (BRR)).
2. S is the value of S in BRR (see section 20.2.9, Bit Rate Register (BRR)).
3. Writable only when TE in SCR = 0 and RE in SCR = 0 (both serial transmission and reception are disabled).

SMR is used to set the SCI's serial transfer format and select the baud rate generator clock source.



Downloaded from Arrow.com.

### CKS[1:0] Bits (Clock Select)

These bits select the clock source for the on-chip baud rate generator.

For the relationship between the settings of these bits and the baud rate, see section 20.2.9, Bit Rate Register (BRR).

### BCP[1:0] Bits (Base Clock Pulse)

These bits select the number of base clock cycles in a 1-bit data transfer time in smart card interface mode.

Set these bits in combination with the BCP2 bit in SCMR.

For details, see section 20.5.4, Receive Data Sampling Timing and Reception Margin.

### PM Bit (Parity Mode)

Selects the parity mode for transmission and reception (even or odd).

For details on the usage of this bit in smart card interface mode, see section 20.5.2, Data Format (Except in Block Transfer Mode).

### PE Bit (Parity Enable)

Set the PE bit to 1.

The parity bit is added to transmit data before transmission, and the parity bit is checked in reception.

### BLK Bit (Block Transfer Mode)

Setting this bit to 1 allows block transfer mode operation.

For details, see section 20.5.3, Block Transfer Mode.

### GM Bit (GSM Mode)

Setting this bit to 1 allows GSM mode operation.

In GSM mode, the SSR.TEND flag set timing is put forward to 11.0 etu (elementary time unit = 1-bit transfer time) from the start and the clock output control function is appended. For details, see 20.5.6, Serial Data Transmission (Except in Block Transfer Mode) and section 20.5.8, Clock Output Control.



Downloaded from Arrow.com.

## 20.2.6 Serial Control Register (SCR)

Note: Some bits in SCR have different functions in serial communications interface mode and smart card interface mode.

### (1) Serial Communications Interface Mode (SMIF in SCMR = 0)

Address: SCI0.SCR 0008 8242h, SCI1.SCR 0008 824Ah, SCI2.SCR 0008 8252h, SCI3.SCR 0008 825Ah
SCI4.SCR 0008 8262h, SCI5.SCR 0008 826Ah, SCI6.SCR 0008 8272h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | – | TEIE | CKE[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b1, b0 | CKE[1:0] | Clock Enable | • For SCI0 to SCI4<br>Asynchronous mode<br>b1 b0<br>0 0: On-chip baud rate generator<br>The SCKn pin functions as I/O port.<br>0 1: On-chip baud rate generator<br>The clock with the same frequency as the bit rate is output from the SCKn pin.<br>1 0: External clock<br>The clock with a frequency 16 times the bit rate should be input from the SCKn pin.<br>(When the SEMR.ABCS bit is 1, the clock with a frequency 8 times the bit rate should be input.)<br>1 1: External clock<br>The clock with a frequency 16 times the bit rate should be input from the SCKn pin.<br>(When the SEMR.ABCS bit is 1, the clock with a frequency 8 times the bit rate should be input.)<br><br>Clock synchronous mode<br>b1 b0<br>0 0: Internal clock<br>The SCKn pin functions as the clock output pin.<br>0 1: Internal clock<br>The SCKn pin functions as the clock output pin.<br>1 0: External clock<br>The SCKn pin functions as the clock input pin.<br>1 1: External clock<br>The SCKn pin functions as the clock input pin. | R/W*1 |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b1, b0 | CKE[1:0] | Clock Enable | • For SCI5 and SCI6<br>Asynchronous mode<br>b1 b0<br>0 0: On-chip baud rate generator<br>The SCKn pin functions as I/O port.<br>0 1: On-chip baud rate generator<br>The clock with the same frequency as the bit rate is output from the SCKn pin.<br>1 0: External clock or TMR clock<br>• When an external clock is used, the clock with a frequency 16 times the bit rate should be input from the SCKn pin. (When the SEMR.ABCS bit is 1, the clock with a frequency 8 times the bit rate should be input.)<br>• The TMR clock can be used.<br>1 1: External clock or TMR clock<br>• When an external clock is used, the clock with a frequency 16 times the bit rate should be input from the SCKn pin. (When the SEMR.ABCS bit is 1, the clock with a frequency 8 times the bit rate should be input.)<br>• The TMR clock can be used.<br><br>Clock synchronous mode<br>b1 b0<br>0 0: Internal clock<br>The SCKn pin functions as the clock output pin.<br>0 1: Internal clock<br>The SCKn pin functions as the clock output pin.<br>1 0: External clock<br>The SCKn pin functions as the clock input pin.<br>1 1: External clock<br>The SCKn pin functions as the clock input pin. | R/W*1 |
| b2 | TEIE | Transmit End Interrupt Enable | 0: A TEI interrupt request is disabled<br>1: A TEI interrupt request is enabled | R/W |
| b3 | — | Reserved | This bit is read as 0. The write value should be 0. | R/W |
| b4 | RE | Receive Enable | 0: Serial reception is disabled<br>1: Serial reception is enabled | R/W*2 |
| b5 | TE | Transmit Enable | 0: Serial transmission is disabled<br>1: Serial transmission is enabled | R/W*2 |
| b6 | RIE | Receive Interrupt Enable | 0: RXI and ERI interrupt requests are disabled<br>1: RXI and ERI interrupt requests are enabled | R/W |
| b7 | TIE | Transmit Interrupt Enable | 0: A TXI interrupt request is disabled<br>1: A TXI interrupt request is enabled | R/W |

Notes: 1. Writable only when TE = 0 and RE = 0.
2. A 1 can be written only when TE = 0 and RE = 0. After setting TE or RE to 1, only 0 can be written in TE and RE.

SCR is a register that enables or disables the SCI transfer operations and selects the transfer clock source.



Downloaded from Arrow.com.

### CKE[1:0] Bits (Clock Enable)

These bits select the clock source and SCKn pin function.

### TEIE Bit (Transmit End Interrupt Enable)

Enables or disables a TEI interrupt request.

A TEI interrupt request is disabled by clearing the TEIE bit to 0.

### RE Bit (Receive Enable)

Enables or disables serial reception.

When this bit is set to 1, serial reception is started by detecting the start bit in asynchronous mode or the synchronous clock input in clock synchronous mode. Note that SMR should be set prior to setting the RE bit to 1 in order to designate the reception format.

Even if reception is halted by clearing the RE bit to 0, the ORER, FER, and PER flags in SSR are not affected and the previous value is retained.

### TE Bit (Transmit Enable)

Enables or disables serial transmission.

When this bit is set to 1, serial transmission is started by writing transmit data to TDR. Note that SMR should be set prior to setting the TE bit to 1 in order to designate the transmission format.

### RIE Bit (Receive Interrupt Enable)

Enables or disables RXI and ERI interrupt requests.

An RXI interrupt request is disabled by clearing the RIE bit to 0.

An ERI interrupt request can be cancelled by reading 1 from the ORER, FER, or PER flag in SSR and then clearing the flag to 0, or clearing the RIE bit to 0.

### TIE Bit (Transmit Interrupt Enable)

Enables or disables notification of a TXI interrupt request.

Notification of a TXI interrupt request is disabled by clearing the TIE bit to 0.



Downloaded from Arrow.com.

## (2) Smart Card Interface Mode (SMIF in SCMR = 1)

Addresses: SCI0.SCR 0008 8242h, SCI1.SCR 0008 824Ah, SCI2.SCR 0008 8295h, SCI3.SCR 0008 25Ah
SCI4.SCR 0008 8262h, SCI5.SCR 0008 826Ah, SCI6.SCR 0008 8272h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | – | TEIE | CKE[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b1, b0 | CKE[1:0] | Clock Enable | • When GM in SMR = 0<br>b1 b0<br>0 0: Output disabled (SCKn pin functions as I/O port.)<br>0 1: Clock output<br>1 0: Setting prohibited<br>1 1: Setting prohibited<br><br>• When GM in SMR = 1<br>0 0: Output fixed low<br>0 1: Clock output<br>1 0: Output fixed high<br>1 1: Clock output | R/W*1 |
| b2 | TEIE | Transmit End Interrupt Enable | This bit should be 0 in smart card interface mode. | R/W |
| b3 | — | Reserved | This bit is read as 0. The write value should be 0. | R/W |
| b4 | RE | Receive Enable | 0: Serial reception is disabled<br>1: Serial reception is enabled | R/W*2 |
| b5 | TE | Transmit Enable | 0: Serial transmission is disabled<br>1: Serial transmission is enabled | R/W*2 |
| b6 | RIE | Receive Interrupt Enable | 0: RXI and ERI interrupt requests are disabled<br>1: RXI and ERI interrupt requests are enabled | R/W |
| b7 | TIE | Transmit Interrupt Enable | 0: A TXI interrupt request is disabled<br>1: A TXI interrupt request is enabled | R/W |

Notes: 1. Writable only when TE = 0 and RE = 0.
2. A 1 can be written only when TE = 0 and RE = 0. After setting TE or RE to 1, only 0 can be written in TE and RE.

SCR is a register that enables/disables the SCI transfer operations and selects the transfer clock source. For details on interrupt requests, see 20.6, Interrupt Sources.

### CKE[1:0] Bits (Clock Enable)

These bits control the clock output from the SCKn pin.

In GSM mode, clock output can be dynamically switched. For details, see section 20.5.8, Clock Output Control.

### TEIE Bit (Transmit End Interrupt Enable)

This bit should be 0 in smart card interface mode.



Downloaded from Arrow.com.

#### RE Bit (Receive Enable)

Enables or disables serial reception.

When this bit is set to 1, serial reception is started by detecting the start bit. Note that SMR should be set prior to setting the RE bit to 1 in order to designate the reception format.

Even if reception is halted by clearing the RE bit to 0, the ORER, FER, and PER flags in SSR are not affected and the previous value is retained.

#### TE Bit (Transmit Enable)

Enables or disables serial transmission.

When this bit is set to 1, serial transmission is started by writing transmit data to TDR. Note that SMR should be set prior to setting the TE bit to 1 in order to designate the transmission format.

#### RIE Bit (Receive Interrupt Enable)

Enables or disables RXI and ERI interrupt requests.

An RXI interrupt request is disabled by clearing the RIE bit to 0.

An ERI interrupt request can be cancelled by reading 1 from the ORER, FER, or PER flag in SSR and then clearing the flag to 0, or clearing the RIE bit to 0.

#### TIE Bit (Transmit Interrupt Enable)

Enables or disables notification of a TXI interrupt request.

Notification of a TXI interrupt request is disabled by clearing the TIE bit to 0.



Downloaded from Arrow.com.

## 20.2.7 Serial Status Register (SSR)

Note: Some bits in SSR have different functions in serial communications interface mode and smart card interface mode.

### (1) Serial Communications Interface Mode (SMIF in SCMR = 0)

Addresses: SCI0.SSR 0008 8244h, SCI1.SSR 0008 824Ch, SCI2.SSR 0008 8254h, SCI3.SSR 0008 25Ch, SCI4.SSR 0008 8264h, SCI5.SSR 0008 826Ch, SCI6.SSR 0008 8274h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | ORER | FER | PER | TEND | – | – |
| Value after reset: | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

x: Undefined

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b0, b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b2 | TEND | Transmit End Flag | 0: A character is being transmitted.<br>1: Character transfer has been completed. | R |
| b3 | PER | Parity Error Flag | 0: No parity error occurred<br>1: A parity error has occurred | R/(W)*1 |
| b4 | FER | Framing Error Flag | 0: No framing error occurred<br>1: A framing error has occurred | R/(W)*1 |
| b5 | ORER | Overrun Error Flag | 0: No overrun error occurred<br>1: An overrun error has occurred | R/(W)*1 |
| b6 | RDRF | Receive Data Full Flag | 0: When data is transferred from RDR<br>1: When data has been received normally, and transferred from RSR to RDR | R/W*2 |
| b7 | TDRE | Transmit Data Empty Flag | 0: When data is transferred to TDR<br>1: When data is transferred from TDR to TSR | R/W*2 |

Notes: 1. Only 0 can be written to this bit, to clear the flag.
2. Write 1 when writing is necessary.

SSR is a register containing status flags of the SCI.

#### TEND Flag (Transmission End Flag)

Indicates completion of transmission.

[Setting conditions]

- Clearing of the SCR.TE bit to 0 (disabling serial transmission operations)
- The TDR is not updated at the time of transmission of the tail-end bit of a character being transmitted

[Clearing condition]

- Writing of further data for transmission to the TDR

When the TEND flag is cleared by writing the data for transmission to the TDR, read the TEND flag and check that it has actually been cleared to 0.



Downloaded from Arrow.com.

### PER Bit (Parity Error Flag)

Indicates that a parity error has occurred during reception in asynchronous mode and the reception ends abnormally.

[Setting condition]

- When a parity error is detected during reception
  Although receive data when the parity error occurs is transferred to RDR, no RXI interrupt request occurs. Note that when the PER flag is being set to 1, the subsequent serial reception cannot be performed. In clock synchronous mode, serial transmission also cannot continue.

[Clearing condition]

- When a 0 is written to PER after reading PER = 1 (After writing a 0 to it, read the PER bit and check that it has actually been cleared to 0.)
  Even when the RE bit in SCR is cleared to 0 (which indicates that serial reception is disabled), the PER flag is not affected and retains its previous value.

### FER Bit (Framing Error Flag)

Indicates that a framing error has occurred during reception in asynchronous mode and the reception ends abnormally.

[Setting condition]

- When the stop bit is 0
  In 2-stop-bit mode, only the first stop bit is checked whether it is 1 but the second stop bit is not checked. Note that although receive data when the framing error occurs is transferred to RDR, no RXI interrupt request occurs. In addition, when the FER flag is being set to 1, the subsequent serial reception cannot be performed. In clock synchronous mode, serial transmission also cannot continue.

[Clearing condition]

- When a 0 is written to FER after reading FER = 1 (After writing a 0 to it, read the FER bit and check that it has actually been cleared to 0.)
  Even when the RE bit in SCR is cleared to 0, the FER flag is not affected and retains its previous value.

### ORER Bit (Overrun Error Flag)

Indicates that an overrun error has occurred during reception and the reception ends abnormally.

[Setting condition]

- When the next data is received before receive data is read from RDR
  In RDR, receive data prior to an overrun error occurrence is retained, but data received after the overrun error occurrence is lost. When the ORER flag is set to 1, subsequent serial reception cannot be performed. Note that, in clock synchronous mode, serial transmission also cannot continue.

[Clearing condition]

- When a 0 is written to ORER after reading ORER = 1 (After writing a 0 to it, read the ORER bit and check that it has actually been cleared to 0.)
  Even when the RE bit in SCR is cleared to 0, the ORER flag is not affected and retains its previous value.



Downloaded from Arrow.com.

### RDRF Bit (Receive Data Full Flag)

Indicates whether RDR has received data.

[Setting condition]

- When data has been received normally, and transferred from RSR to RDR

[Clearing condition]

- When data is transferred from RDR.

### TDRE Bit (Transmit Data Empty Flag)

Indicates whether TDR has data to be transmitted.

[Setting condition]

- When data is transferred from TDR to TSR

[Clearing condition]

- When data is transferred to TDR



Downloaded from Arrow.com.

## (2) Smart Card Interface Mode (SMIF in SCMR = 1)

Addresses: SCI0.SSR 0008 8244h, SCI1.SSR 0008 824Ch, SCI2.SSR 0008 8254h, SCI3.SSR 0008 25Ch, SCI4.SSR 0008 8264h, SCI5.SSR 0008 826Ch, SCI6.SSR 0008 8274h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | ORER | ERS | PER | TEND | – | – |
| Value after reset: | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

x: Undefined

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b0, b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b2 | TEND | Transmit End Flag | 0: A character is being transmitted.<br>1: Character transfer has been completed. | R |
| b3 | PER | Parity Error Flag | 0: No parity error occurred<br>1: A parity error has occurred | R/(W)*1 |
| b4 | ERS | Error Signal Status Flag | 0: Low error signal not responded<br>1: Low error signal responded | R/(W)*1 |
| b5 | ORER | Overrun Error Flag | 0: No overrun error occurred<br>1: An overrun error has occurred | R/(W)*1 |
| b6 | RDRF | Receive Data Full Flag | 0: When data is transferred from RDR<br>1: When data has been received normally, and transferred from RSR to RDR | R/W*2 |
| b7 | TDRE | Transmit Data Empty Flag | 0: When data is transferred to TDR<br>1: When data is transferred from TDR to TSR | R/W*2 |

Notes: 1. Only 0 can be written to this bit, to clear the flag.
2. Write 1 when writing is necessary.

SSR is a register containing status flags of the SCI.

### TEND Flag (Transmission End Flag)

With no error signal from the receiving side, this bit is set to 1 when further data for transfer is ready to be transferred to the TDR register.

[Setting conditions]

- When SCR.TE bit = 0 (disabling serial transmission operations) and ERS flag = 0
- When a specified period has elapsed after the latest transmission of one byte, the ERS flag is 0, and the TDR register is not updated
  The set timing is determined by register settings as listed below.
  When SMR.GM = 0 and SMR.BLK = 0, 12.5 etu after the start of transmission
  When SMR.GM = 0 and SMR.BLK = 1, 11.5 etu after the start of transmission
  When SMR.GM = 1 and SMR.BLK = 0, 11.0 etu after the start of transmission
  When SMR.GM = 1 and SMR.BLK = 1, 11.0 etu after the start of transmission

[Clearing condition]

- Writing further data for transmission to the TDR register



Downloaded from Arrow.com.

### PER Flag (Parity Error Flag)

Indicates that a parity error has occurred during reception in asynchronous mode and the reception ends abnormally.

[Setting condition]

- When a parity error is detected during reception
  Although receive data when the parity error occurs is transferred to RDR, no RXI interrupt request occurs. Note that when the PER flag is being set to 1, the subsequent serial reception cannot be performed. In clock synchronous mode, serial transmission also cannot continue.

[Clearing condition]

- When a 0 is written to PER after reading PER = 1 (After writing a 0 to it, read the PER bit and check that it has actually been cleared to 0.)
  Even when the RE bit in SCR is cleared to 0 (which indicates that serial reception is disabled), the PER flag is not affected and retains its previous value.

### ERS Flag (Error Signal Status Flag)

[Setting condition]

- When a low error signal is sampled

[Clearing condition]

- When a 0 is written to ERS after reading ERS = 1

### ORER Flag (Overrun Error Flag)

Indicates that an overrun error has occurred during reception and the reception ends abnormally.

[Setting condition]

- When the next data is received before receive data is read from RDR
  In RDR, the receive data prior to an overrun error occurrence is retained, but data received following the overrun error occurrence is lost. When the ORER flag is set to 1, subsequent serial reception cannot be performed.

[Clearing condition]

- When a 0 is written to ORER after reading ORER = 1 (After writing a 0 to it, read the ORER bit and check that it has actually been cleared to 0.)
  Even when the RE bit in SCR is cleared to 0, the ORER flag is not affected and retains its previous value.

### RDRF Bit (Receive Data Full Flag)

Indicates whether RDR has received data.

[Setting condition]

- When data has been received normally, and transferred from RSR to RDR
  [Clearing condition]
- When data is transferred from RDR.



Downloaded from Arrow.com.

### TDRE Bit (Transmit Data Empty Flag)

Indicates whether TDR has data to be transmitted.

[Setting condition]

- When data is transferred from TDR to TSR

[Clearing condition]

- When data is transferred to TDR



Downloaded from Arrow.com.

## 20.2.8 Smart Card Mode Register (SCMR)

Addresses: SCI0.SCMR 0008 8246h, SCI1.SCMR 0008 824Eh, SCI2.SCMR 0008 8256h, SCI3.SCMR 0008 25Eh
SCI4.SCMR 0008 8266h, SCI5.SCMR 0008 826Eh, SCI6.SCMR 0008 8276h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | BCP2 | – | – | – | SDIR | SINV | – | SMIF |
| Value after reset: | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b0 | SMIF | Smart Card Interface Mode Select | 0: Serial communications interface mode<br>1: Smart card interface mode | R/W*1 |
| b1 | — | Reserved | This bit is read as 1. The write value should be 1. | R/W |
| b2 | SINV | Smart Card Data Invert | 0: TDR contents are transmitted as they are. Receive data is stored as it is in RDR.<br>1: TDR contents are inverted before being transmitted. Receive data is stored in inverted form in RDR. | R/W*1 |
| b3 | SDIR | Bit Order Select | 0: Transfer with LSB-first<br>1: Transfer with MSB-first | R/W*1 |
| b6 to b4 | — | Reserved | These bits are read as 1. The write value should be 1. | R/W |
| b7 | BCP2 | Base Clock Pulse 2 | Selects the number of base clock cycles in combination with the BCP1 and BCP0 bits in SMR.<br>Setting values in BCP2 bit in SCMR and BCP1 and BCP0 bits in SMR<br>BCP2 BCP1 BCP0<br>0 0 0: 93 clock cycles (S = 93)*2<br>0 0 1: 128 clock cycles (S = 128)*2<br>0 1 0: 186 clock cycles (S = 186)*2<br>0 1 1: 512 clock cycles (S = 512)*2<br>1 0 0: 32 clock cycles (S = 32)*2 (Initial Value)<br>1 0 1: 64 clock cycles (S = 64)*2<br>1 1 0: 372 clock cycles (S = 372)*2<br>1 1 1: 256 clock cycles (S = 256)*2 | R/W*1 |

Notes: 1. Writable only when TE in SCR = 0 and RE in SCR = 0 (both serial transmission and reception are disabled).
2. S is the value of S in BRR (see section 20.2.9, Bit Rate Register (BRR)).

SCMR selects smart card interface mode and its format.

### SMIF Bit (Smart Card Interface Mode Select)

When this bit is set to 1, smart card interface mode is selected.

When this bit is set to 0, asynchronous or clock synchronous mode is selected.

### SINV Bit (Smart Card Data Invert)

Inverts the transmit/receive data logic level. This bit does not affect the logic level of the parity bit. To invert the parity bit, invert the PM bit in SMR.



Downloaded from Arrow.com.

### SDIR Bit (Bit Order Select)

Selects the serial/parallel conversion format.

### BCP2 Bit (Base Clock Pulse 2)

Selects the number of base clock cycles in a 1-bit data transfer time in smart card interface mode. Set this bit in combination with the BCP1 and BCP2 bits in SMR.



Downloaded from Arrow.com.

## 20.2.9 Bit Rate Register (BRR)

Addresses: SCI0.BRR 0008 8241h, SCI1.BRR 0008 8249h, SCI2.BRR 0008 8251h, SCI3.BRR 0008 8259h
SCI4.BRR 0008 8261h, SCI5.BRR 0008 8269h, SCI6.BRR 0008 8271h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

BRR is an 8-bit register that adjusts the bit rate.

As the SCI performs baud rate generator control independently for each channel, different bit rates can be set for each channel. Table 20.5 shows the relationships between the N setting in BRR and bit rate B for normal asynchronous mode and clock synchronous mode, and smart card interface mode.

The initial value of BRR is FFh.

BRR can be read from by the CPU at all times, but it can be written to only when the TE and RE bits in SCR are 0.

**Table 20.5 Relationships between N Setting in BRR and Bit Rate B**

| Mode | ABCS Bit in SEMR | BRR Setting | Error |
|---|---|---|---|
| Asynchronous | 0 | $N = \frac{PCLK \times 10^6}{64 \times 2^{2n-1} \times B} - 1$ | $Error\ (\%) = \{ \frac{PCLK \times 10^6}{B \times 64 \times 2^{2n-1} \times (N+1)} - 1 \} \times 100$ |
| | 1 | $N = \frac{PCLK \times 10^6}{32 \times 2^{2n-1} \times B} - 1$ | $Error\ (\%) = \{ \frac{PCLK \times 10^6}{B \times 32 \times 2^{2n-1} \times (N+1)} - 1 \} \times 100$ |
| Clock synchronous | | $N = \frac{PCLK \times 10^6}{8 \times 2^{2n-1} \times B} - 1$ | |
| Smart card interface | | $N = \frac{PCLK \times 10^6}{S \times 2^{2n+1} \times B} - 1$ | $Error\ (\%) = \{ \frac{PCLK \times 10^6}{B \times S \times 2^{2n+1} \times (N+1)} - 1 \} \times 100$ |

[Legend]
B: Bit rate (bps)
N: BRR setting for baud rate generator ($0 \leq N \leq 255$)
PCLK: Operating frequency (MHz)
n and S: Determined by the SMR setting shown in the following table.

| SMR Setting | | | |
|---|---|---|---|
| **CKS[1:0] bps** | | **Clock Source** | **n** |
| 0 | 0 | PCLK clock | 0 |
| 0 | 1 | PCLK/4 clock | 1 |
| 1 | 0 | PCLK/16 clock | 2 |
| 1 | 1 | PCLK/64 clock | 3 |



Downloaded from Arrow.com.

| SCMR Setting | SMR Setting | | |
|---|---|---|---|
| BCP2 Bit | BCP[1:0] bps | Base Clock | S |
| 0 | 0 0 | 93 clock cycles | 93 |
| 0 | 0 1 | 128 clock cycles | 128 |
| 0 | 1 0 | 186 clock cycles | 186 |
| 0 | 1 1 | 512 clock cycles | 512 |
| 1 | 0 0 | 32 clock cycles | 32 |
| 1 | 0 1 | 64 clock cycles | 64 |
| 1 | 1 0 | 372 clock cycles | 372 |
| 1 | 1 1 | 256 clock cycles | 256 |

Tables 20.6 shows sample N settings in BRR in normal asynchronous mode. Table 20.7 shows the maximum bit rate settable for each operating frequency. Tables 20.9 and 20.11 show sample N settings in BRR in clock synchronous mode and smart card interface mode, respectively. In smart card interface mode, the number of base clock cycles S in a 1-bit data transfer time can be selected. For details, see section 20.5.4, Receive Data Sampling Timing and Reception Margin. Tables 20.8 and 20.10 show the maximum bit rates with external clock input.

When the ABCS bit in the serial extended mode register (SEMR) is set to 1 in asynchronous mode, the bit rate is two times that of shown in table 20.6.



Downloaded from Arrow.com.

**Table 20.6 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (1)**

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 | | | 9.8304 | | | 10 | | | 12 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 141 | 0.03 | 2 | 174 | -0.26 | 2 | 177 | -0.25 | 2 | 212 | 0.03 |
| 150 | 2 | 103 | 0.16 | 2 | 127 | 0.00 | 2 | 129 | 0.16 | 2 | 155 | 0.16 |
| 300 | 1 | 207 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 | 2 | 77 | 0.16 |
| 600 | 1 | 103 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 | 1 | 155 | 0.16 |
| 1200 | 0 | 207 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 | 1 | 77 | 0.16 |
| 2400 | 0 | 103 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 | 0 | 155 | 0.16 |
| 4800 | 0 | 51 | 0.16 | 0 | 63 | 0.00 | 0 | 64 | 0.16 | 0 | 77 | 0.16 |
| 9600 | 0 | 25 | 0.16 | 0 | 31 | 0.00 | 0 | 32 | -1.36 | 0 | 38 | 0.16 |
| 19200 | 0 | 12 | 0.16 | 0 | 15 | 0.00 | 0 | 15 | 1.73 | 0 | 19 | -2.34 |
| 31250 | 0 | 7 | 0.00 | 0 | 9 | -1.70 | 0 | 9 | 0.00 | 0 | 11 | 0.00 |
| 38400 | - | - | - | 0 | 7 | 0.00 | 0 | 7 | 1.73 | 0 | 9 | -2.34 |

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 12.288 | | | 14 | | | 16 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 2 | 217 | 0.08 | 2 | 248 | -0.17 | 3 | 70 | 0.03 |
| 150 | 2 | 159 | 0.00 | 2 | 181 | 0.16 | 2 | 207 | 0.16 |
| 300 | 2 | 79 | 0.00 | 2 | 90 | 0.16 | 2 | 103 | 0.16 |
| 600 | 1 | 159 | 0.00 | 1 | 181 | 0.16 | 1 | 207 | 0.16 |
| 1200 | 1 | 79 | 0.00 | 1 | 90 | 0.16 | 1 | 103 | 0.16 |
| 2400 | 0 | 159 | 0.00 | 0 | 181 | 0.16 | 0 | 207 | 0.16 |
| 4800 | 0 | 79 | 0.00 | 0 | 90 | 0.16 | 0 | 103 | 0.16 |
| 9600 | 0 | 39 | 0.00 | 0 | 45 | -0.93 | 0 | 51 | 0.16 |
| 19200 | 0 | 19 | 0.00 | 0 | 22 | -0.93 | 0 | 25 | 0.16 |
| 31250 | 0 | 11 | 2.40 | 0 | 13 | 0.00 | 0 | 15 | 0.00 |
| 38400 | 0 | 9 | 0.00 | — | — | — | 0 | 12 | 0.16 |

Notes: This is an example when the ABCS bit in SEMR is 0.
When the ABCS bit is set to 1, the bit rate is two times.



Downloaded from Arrow.com.

**Table 20.6 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (2)**

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 17.2032 | | | 18 | | | 19.6608 | | | 20 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 75 | 0.48 | 3 | 79 | -0.12 | 3 | 86 | 0.31 | 3 | 88 | -0.25 |
| 150 | 2 | 223 | 0.00 | 2 | 233 | 0.16 | 2 | 255 | 0.00 | 3 | 64 | 0.16 |
| 300 | 2 | 111 | 0.00 | 2 | 116 | 0.16 | 2 | 127 | 0.00 | 2 | 129 | 0.16 |
| 600 | 1 | 223 | 0.00 | 1 | 233 | 0.16 | 1 | 255 | 0.00 | 2 | 64 | 0.16 |
| 1200 | 1 | 111 | 0.00 | 1 | 116 | 0.16 | 1 | 127 | 0.00 | 1 | 129 | 0.16 |
| 2400 | 0 | 223 | 0.00 | 0 | 233 | 0.16 | 0 | 255 | 0.00 | 1 | 64 | 0.16 |
| 4800 | 0 | 111 | 0.00 | 0 | 116 | 0.16 | 0 | 127 | 0.00 | 0 | 129 | 0.16 |
| 9600 | 0 | 55 | 0.00 | 0 | 58 | -0.69 | 0 | 63 | 0.00 | 0 | 64 | 0.16 |
| 19200 | 0 | 27 | 0.00 | 0 | 28 | 1.02 | 0 | 31 | 0.00 | 0 | 32 | -1.36 |
| 31250 | 0 | 16 | 1.20 | 0 | 17 | 0.00 | 0 | 19 | -1.70 | 0 | 19 | 0.00 |
| 38400 | 0 | 13 | 0.00 | 0 | 14 | -2.34 | 0 | 15 | 0.00 | 0 | 15 | 1.73 |

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25 | | | 30 | | | 33 | | | 50 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 3 | 110 | -0.02 | 3 | 132 | 0.13 | 3 | 145 | 0.33 | 3 | 221 | -0.02 |
| 150 | 3 | 80 | 0.47 | 3 | 97 | -0.35 | 3 | 106 | 0.39 | 3 | 162 | -0.15 |
| 300 | 2 | 162 | -0.15 | 2 | 194 | 0.16 | 2 | 214 | -0.07 | 3 | 80 | 0.47 |
| 600 | 2 | 80 | 0.47 | 2 | 97 | -0.35 | 2 | 106 | 0.39 | 2 | 162 | -0.15 |
| 1200 | 1 | 162 | -0.15 | 1 | 194 | 0.16 | 1 | 214 | -0.07 | 2 | 80 | 0.47 |
| 2400 | 1 | 80 | 0.47 | 1 | 97 | -0.35 | 1 | 106 | 0.39 | 1 | 162 | -0.15 |
| 4800 | 0 | 162 | -0.15 | 0 | 194 | 0.16 | 0 | 214 | -0.07 | 1 | 80 | 0.47 |
| 9600 | 0 | 80 | 0.47 | 0 | 97 | -0.35 | 0 | 106 | 0.39 | 1 | 40 | -0.77 |
| 19200 | 0 | 40 | -0.76 | 0 | 48 | -0.35 | 0 | 53 | -0.54 | 0 | 80 | 0.47 |
| 31250 | 0 | 24 | 0.00 | 0 | 29 | 0 | 0 | 32 | 0 | 0 | 49 | 0.00 |
| 38400 | 0 | 19 | 1.73 | 0 | 23 | 1.73 | 0 | 26 | -0.54 | 0 | 40 | -0.77 |

Notes: This is an example when the ABCS bit in SEMR is 0.
When the ABCS bit is set to 1, the bit rate is two times.



Downloaded from Arrow.com.

**Table 20.7 Maximum Bit Rate for Each Operating Frequency (Asynchronous Mode)**

| PCLK (MHz) | Maximum Bit Rate (bps) | n | N | PCLK (MHz) | Maximum Bit Rate (bps) | n | N |
|---|---|---|---|---|---|---|---|
| 8 | 250000 | 0 | 0 | 17.2032 | 537600 | 0 | 0 |
| 9.8304 | 307200 | 0 | 0 | 18 | 562500 | 0 | 0 |
| 10 | 312500 | 0 | 0 | 19.6608 | 614400 | 0 | 0 |
| 12 | 375000 | 0 | 0 | 20 | 625000 | 0 | 0 |
| 12.288 | 384000 | 0 | 0 | 25 | 781250 | 0 | 0 |
| 14 | 437500 | 0 | 0 | 30 | 937500 | 0 | 0 |
| 16 | 500000 | 0 | 0 | 33 | 1031250 | 0 | 0 |
| | | | | 50 | 1562500 | 0 | 0 |

Note: When the ABCS bit in SEMR is set to 1, the bit rate is two times.

**Table 20.8 Maximum Bit Rate with External Clock Input (Asynchronous Mode) (1)**

| PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) | PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) |
|---|---|---|---|---|---|
| 8 | 2.0000 | 125000 | 17.2032 | 4.3008 | 268800 |
| 9.8304 | 2.4576 | 153600 | 18 | 4.5000 | 281250 |
| 10 | 2.5000 | 156250 | 19.6608 | 4.9152 | 307200 |
| 12 | 3.0000 | 187500 | 20 | 5.0000 | 312500 |
| 12.288 | 3.0720 | 192000 | 25 | 6.2500 | 390625 |
| 14 | 3.5000 | 218750 | 30 | 7.5000 | 468750 |
| 16 | 4.0000 | 250000 | 33 | 8.2500 | 515625 |
| | | | 50 | 12.500 | 781250 |

Note: This is an example when the ABCS bit in SEMR is 0.

**Table 20.8 Maximum Bit Rate with External Clock Input (Asynchronous Mode) (2)**

| PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) | PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) |
|---|---|---|---|---|---|
| 8 | 2.0000 | 250000 | 17.2032 | 4.3008 | 537600 |
| 9.8304 | 2.4576 | 307200 | 18 | 4.5000 | 562500 |
| 10 | 2.5000 | 312500 | 19.6608 | 4.9152 | 614400 |
| 12 | 3.0000 | 375000 | 20 | 5.0000 | 625000 |
| 12.288 | 3.0720 | 384000 | 25 | 6.2500 | 781250 |
| 14 | 3.5000 | 437500 | 30 | 7.5000 | 937500 |
| 16 | 4.0000 | 500000 | 33 | 8.2500 | 1031250 |
| | | | 50 | 12.500 | 1562500 |

Note: This is an example when the ABCS bit in SEMR is 1.



Downloaded from Arrow.com.

**Table 20.9 BRR Settings for Various Bit Rates (Clock Synchronous Mode)**

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 | | 10 | | 16 | | 20 | | 25 | | 30 | | 33 | | 50 | |
| | n | N | n | N | n | N | n | N | n | N | n | N | n | N | n | N |
| 110 | | | | | | | | | | | | | | | | |
| 250 | 3 | 124 | - | - | 3 | 249 | | | | | | | | | | |
| 500 | 2 | 249 | - | - | 3 | 124 | - | - | | | 3 | 233 | | | | |
| 1k | 2 | 124 | - | - | 2 | 249 | - | - | 3 | 97 | 3 | 116 | 3 | 128 | 3 | 194 |
| 2.5k | 1 | 199 | 1 | 249 | 2 | 99 | 2 | 124 | 2 | 155 | 2 | 187 | 2 | 205 | 3 | 77 |
| 5k | 1 | 99 | 1 | 124 | 1 | 199 | 1 | 249 | 2 | 77 | 2 | 93 | 2 | 102 | 2 | 155 |
| 10k | 0 | 199 | 0 | 249 | 1 | 99 | 1 | 124 | 1 | 155 | 1 | 187 | 1 | 205 | 2 | 77 |
| 25k | 0 | 79 | 0 | 99 | 0 | 159 | 0 | 199 | 0 | 249 | 1 | 74 | 1 | 82 | 1 | 124 |
| 50k | 0 | 39 | 0 | 49 | 0 | 79 | 0 | 99 | 0 | 124 | 0 | 149 | 0 | 164 | 1 | 61 |
| 100k | 0 | 19 | 0 | 24 | 0 | 39 | 0 | 49 | 0 | 62 | 0 | 74 | 0 | 82 | 0 | 124 |
| 250k | 0 | 7 | 0 | 9 | 0 | 15 | 0 | 19 | 0 | 24 | 0 | 29 | 0 | 32 | 0 | 49 |
| 500k | 0 | 3 | 0 | 4 | 0 | 7 | 0 | 9 | - | - | 0 | 14 | - | - | 0 | 24 |
| 1M | 0 | 1 | - | - | 0 | 3 | 0 | 4 | - | - | - | - | - | - | - | - |
| 2M | 0 | 0* | - | - | 0 | 1 | - | - | - | - | - | - | - | - | - | - |
| 2.5M | | | 0 | 0* | - | - | 0 | 1 | - | - | 0 | 2 | - | - | 0 | 4 |
| 4M | | | | | 0 | 0* | - | - | - | - | - | - | - | - | - | - |
| 5M | | | | | | | 0 | 0* | - | - | - | - | - | - | - | - |
| 6.25M | | | | | | | | | 0 | 0* | - | - | - | - | 0 | 1 |
| 7.5M | | | | | | | | | | | 0 | 0* | - | - | - | - |
| 8.25M | | | | | | | | | | | | | 0 | 0* | - | - |
| 12.5M | | | | | | | | | | | | | | | 0 | 0* |

[Legend]

Space: Setting prohibited.

-: Can be set, but there will be error.

Note: * Continuous transmission or reception is not poss ble.

**Table 20.10 Maximum Bit Rate with External Clock Input (Clock Synchronous Mode)**

| PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) | PCLK (MHz) | External Input Clock (MHz) | Maximum Bit Rate (bps) |
|---|---|---|---|---|---|
| 8 | 1.3333 | 1333333.3 | 20 | 3.3333 | 3333333.3 |
| 10 | 1.6667 | 1666666.7 | 25 | 4.1667 | 4166666.7 |
| 12 | 2.0000 | 2000000.0 | 30 | 5.0000 | 5000000.0 |
| 14 | 2.3333 | 2333333.3 | 33 | 5.5000 | 5500000.0 |
| 16 | 2.6667 | 2666666.7 | 50 | 8.3333 | 8333333.3 |
| 18 | 3.0000 | 3000000.0 | | | |



Downloaded from Arrow.com.

**Table 20.11 BRR Settings for Various Bit Rates (Smart Card Interface Mode, n = 0, S = 372)**

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7.1424 | | | 10.00 | | | 10.7136 | | | 13.00 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 9600 | 0 | 0 | 0.00 | 0 | 1 | 30 | 0 | 1 | 25 | 0 | 1 | 8.99 |

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14.2848 | | | 16.00 | | | 18.00 | | | 20.00 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 9600 | 0 | 1 | 0.00 | 0 | 1 | 12.01 | 0 | 2 | 15.99 | 0 | 2 | 6.66 |

| Bit Rate (bps) | Operating Frequency PCLK (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25.00 | | | 30.00 | | | 33.00 | | | 50.00 | | |
| | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 9600 | 0 | 3 | 12.49 | 0 | 3 | 5.01 | 0 | 4 | 7.59 | 0 | 6 | 0.01 |

**Table 20.12 Maximum Bit Rate for Each Operating Frequency (Smart Card Interface Mode, S = 372)**

| PCLK (MHz) | Maximum Bit Rate (bps) | n | N | PCLK (MHz) | Maximum Bit Rate (bps) | n | N |
|---|---|---|---|---|---|---|---|
| 10.00 | 13441 | 0 | 0 | 18.00 | 24194 | 0 | 0 |
| 10.7136 | 14400 | 0 | 0 | 20.00 | 26882 | 0 | 0 |
| 13.00 | 17473 | 0 | 0 | 25.00 | 33602 | 0 | 0 |
| 16.00 | 21505 | 0 | 0 | 30.00 | 40323 | 0 | 0 |
| | | | | 33.00 | 44355 | 0 | 0 |
| | | | | 50.00 | 67205 | 0 | 0 |



Downloaded from Arrow.com.

## 20.2.10 Serial Extended Mode Register (SEMR)

Addresses: SCI0.SEMR 0008 8247h, SCI1.SEMR 0008 824Fh, SCI2.SEMR 0008 8257h, SCI3.SEMR 0008 825Fh
SCI4.SEMR 0008 8267h, SCI5.SEMR 0008 826Fh, SCI6.SEMR 0008 8277h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | ABCS | – | – | – | ACS0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Function | R/W |
|---|---|---|---|---|
| b0 | ACS0 | Asynchronous Mode Clock Source Select | (Valid only in asynchronous mode)<br>0: External clock input (SCI0 to SCI6)<br>1: TMR clock input (valid only for SCI5 and SCI6)<br>The following table shows the correspondence between SCI channels and compare match outputs. | R/W* |
| b3 to b1 | — | Reserved | These bits are read as 0. The write value should be 0. | R/W |
| b4 | ABCS | Asynchronous Mode Base Clock Select | (Valid only in asynchronous mode)<br>0: Selects 16 base clock cycles for 1-bit period<br>1: Selects 8 base clock cycles for 1-bit period | R/W* |
| b7 to b5 | — | Reserved | These bits are read as 0. The write value should be 0.. | R/W |

| SCI | TMR | Compare Match Output |
|---|---|---|
| SCI5 | Unit 0 | TMO0, TMO1 |
| SCI6 | Unit 1 | TMO2, TMO3 |

Note: * Writable only when TE in SCR = 0 and RE in SCR = 0 (both serial transmission and reception are disabled).

SEMR selects the clock source for 1-bit period in asynchronous mode.

For SCI5 and SCI6, the TMOn output (n = 0 to 3) of TMR units 0 and 1 can be set as the serial transfer base clock.

Figure 20.3 shows a setting example when the TMOn output of TMRn (n = 0 to 3) is selected.

### ACS0 Bit (Asynchronous Mode Clock Source Select)

Selects the clock source in the asynchronous mode.

The ACS0 bit is valid in asynchronous mode (CM bit in SMR = 0) and when an external clock input is selected (CKE[1:0] bits in SCR = 10b or 11b). An external clock input or internal TMR clock input can be selected.

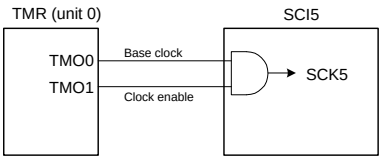
These bits for the other SCI channels than SCI5 and SCI6 are reserved. The write values to these bits for other than SCI5 and SCI6 should always be 0.

### ABCS Bit (Asynchronous Mode Base Clock Select)

Selects the number of base clock pulses for 1-bit period.



Downloaded from Arrow.com.




This figure shows an example when TMR clock is input to SCI5.
When generating 187.5 kbps of TMR average transfer rate for PCLK = 32 MHz:

(1) Generate a frequency of 4 MHz using TMO0 as the base clock.
(2) Set TMO1 in the compare match count of TCNT in TMR0 and generate 3/4 clock enable to set an average transfer rate of 3 MHz/16 = 187.5 kbps.

Setting examples of TMR and SCI
TMR0.TCR = 08h (TMR0.TCNT is cleared at compare match in TMR0.TCORA, the count is incremented at rising edge of PCLK/2)
TMR0.TCCR = 09h
TMR1.TCR = 08h (TMR1.TCNT is cleared at compare match in TMR1.TCORA, the count is incremented at compare match A in TMR0.TCNT)
TMR1.TCCR = 18h
TMR0.TCSR = 09h (Low is output at compare match in TMR0.TCORA, High is output at compare match in TMR0.TCORB)
TMR1.TCSR = 09h (Low is output at compare match in TMR1.TCORA, High is output at compare match in TMR1.TCORB)
TMR0.TCNT = TMR1.TCNT = 0
TMR0.TCORA = 03h, TMR0.TCORB = 01h
TMR1.TCORA = 03h, TMR1.TCORB = 00h
SCI5.SEMR = 10h

When using SCI6, set TM02 as the base clock and TMO3 as clock enable.

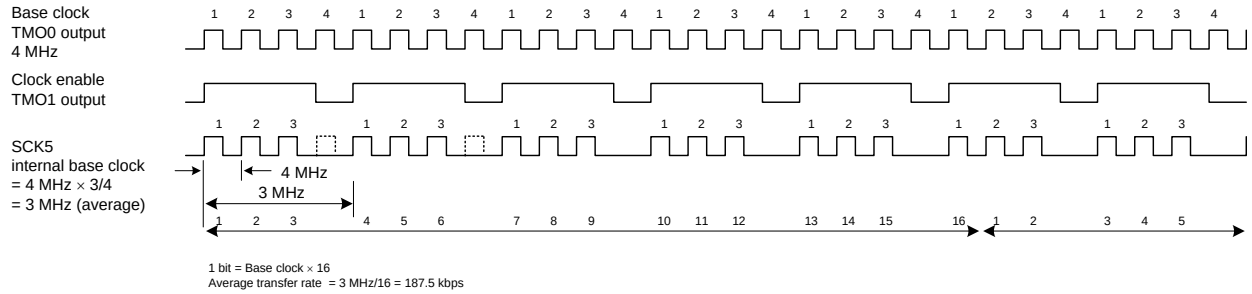



**Figure 20.3 Example of Average Transfer Rate Setting when TMR Clock is Input**



Downloaded from Arrow.com.

## 20.3 Operation in Asynchronous Mode

Figure 20.4 shows the general format for asynchronous serial communications.

One frame consists of a start bit (low level), transmit/receive data, a parity bit, and stop bits (high level).

In asynchronous serial communications, the communications line is usually held in the mark state (high level).

The SCI monitors the communications line, and when it goes to the space state (low level), recognizes a start bit and starts serial communications.

Inside the SCI, the transmitter and receiver are independent units, enabling full-duplex communications. Both the transmitter and the receiver also have a double-buffered structure, so that data can be read or written during transmission or reception, enabling continuous data transmission and reception.

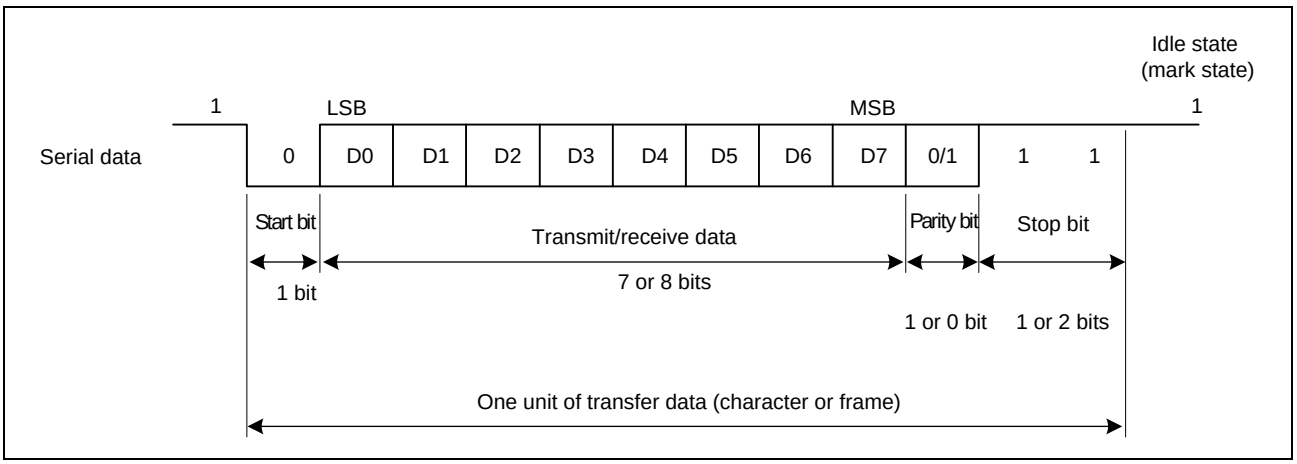



**Figure 20.4 Data Format in Asynchronous Serial Communications (Example with 8-Bit Data, Parity, Two Stop Bits)**



Downloaded from Arrow.com.

### 20.3.1 Serial Data Transfer Format

Table 20.13 shows the serial data transfer formats that can be used in asynchronous mode.

Any of 8 transfer formats can be selected according to the SMR setting.

**Table 20.13 Serial Transfer Formats (Asynchronous Mode)**

| SMR Setting | | | Serial Transfer 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 |
| 0 | 0 | 1 | S 8-bit data STOP STOP |
| 0 | 1 | 0 | S 8-bit data P STOP |
| 0 | 1 | 1 | S 8-bit data P STOP STOP |
| 1 | 0 | 0 | S 7-bit data STOP |
| 1 | 0 | 1 | S 7-bit data STOP STOP |
| 1 | 1 | 0 | S 7-bit data P STOP |
| 1 | 1 | 1 | S 7-bit data P STOP STOP |

[Legend] S: Start bit
STOP: Stop bit
P: Parity bit



Downloaded from Arrow.com.

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

In asynchronous mode, the SCI operates on a base clock with a frequency of 16 times* the bit rate.

In reception, the SCI samples the falling edge of the start bit using the base clock, and performs internal synchronization. Since receive data is sampled at the rising edge of the 8th pulse* of the base clock, data is latched at the middle of each bit, as shown in figure 20.5. Thus the reception margin in asynchronous mode is determined by formula (1) below.

$$M = \left| (0.5 - \frac{1}{2N}) - (L - 0.5) F - \frac{D - 0.5}{N} (1+F) \right| \times 100\ [\%] \quad \cdots \text{Formula (1)}$$

[Legend]
M: Reception margin
N: Ratio of bit rate to clock (N = 16 when ABCS in SEMR = 0, N = 8 when ABCS in SEMR = 1)
D: Duty cycle of clock (D = 0.5 to 1.0)
L: Frame length (L = 9 to 12)
F: Absolute value of clock frequency deviation

Assuming values of F = 0 and D = 0.5 in formula (1), the reception margin is determined by the formula below.

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

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

Note: * This is an example when the ABCS bit in SEMR is 0. When the ABCS bit is 1, a frequency of 8 times the bit rate is used as a base clock and receive data is sampled at the rising edge of the 4th pulse of the base clock.

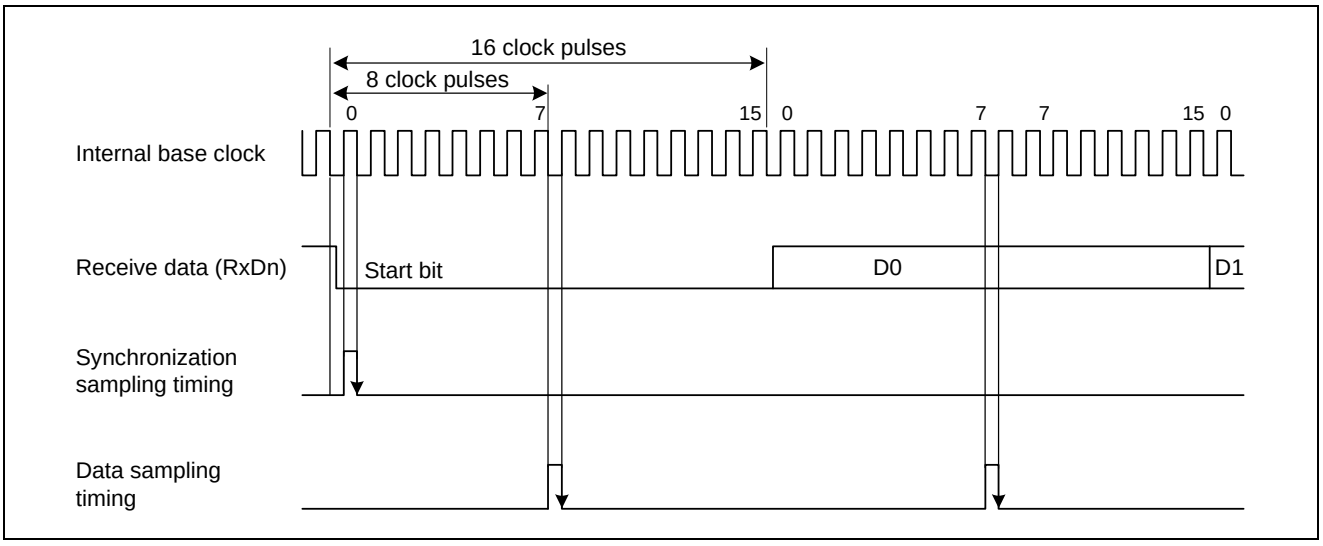



**Figure 20.5 Receive Data Sampling Timing in Asynchronous Mode**



Downloaded from Arrow.com.

### 20.3.3 Clock

Either an internal clock generated by the on-chip baud rate generator or an external clock input to the SCKn pin can be selected as the SCI's transfer clock, according to the setting of the CA bit in SMR and the CKE[1:0] bits in SCR.

When an external clock is input to the SCKn pin, the clock frequency should be 16 times the bit rate (when ABCS in SEMR = 0) and 8 times the bit rate (when ABCS in SEMR = 1). In addition, when an external clock is specified, the base clock of TMR0 and TMR1 can be selected by the ACS0 bit in SEMR of SCIn (n = 5, 6).

When the SCI is operated on an internal clock, the clock can be output from the SCKn pin. The frequency of the clock output in this case is equal to the bit rate, and the phase is such that the rising edge of the clock is in the middle of the transmit data, as shown in figure 20.6.

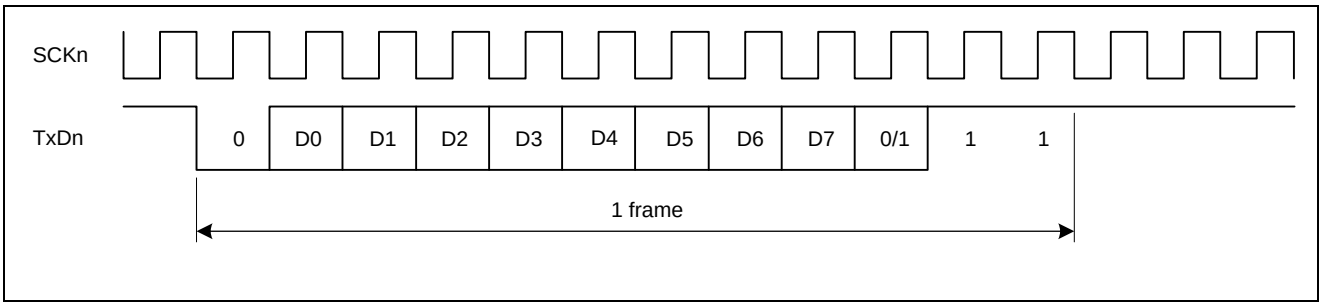



**Figure 20.6 Phase Relationship between Output Clock and Transmit Data (Asynchronous Mode)**



Downloaded from Arrow.com.

### 20.3.4 SCI Initialization (Asynchronous Mode)

Before transmitting and receiving data, start by writing the initial value "00h" to the SCR and then continue through the procedure for SCI given in the sample flowchart (figure 20.7). Whenever the operating mode or transfer format is changed, the SCR must be initialized before the change is made.

When the external clock is used in asynchronous mode, ensure that the clock signal is supplied even during initialization.

Note that clearing the SCR.RE bit to 0 initializes neither the ORER, FER, and PER flags in SSR nor RDR.

Moreover, note that switching the value of the SCR.TE bit from 1 to 0 or 0 to 1 while the SCR.TIE bit is 1 leads to the generation of a TXI interrupt request.

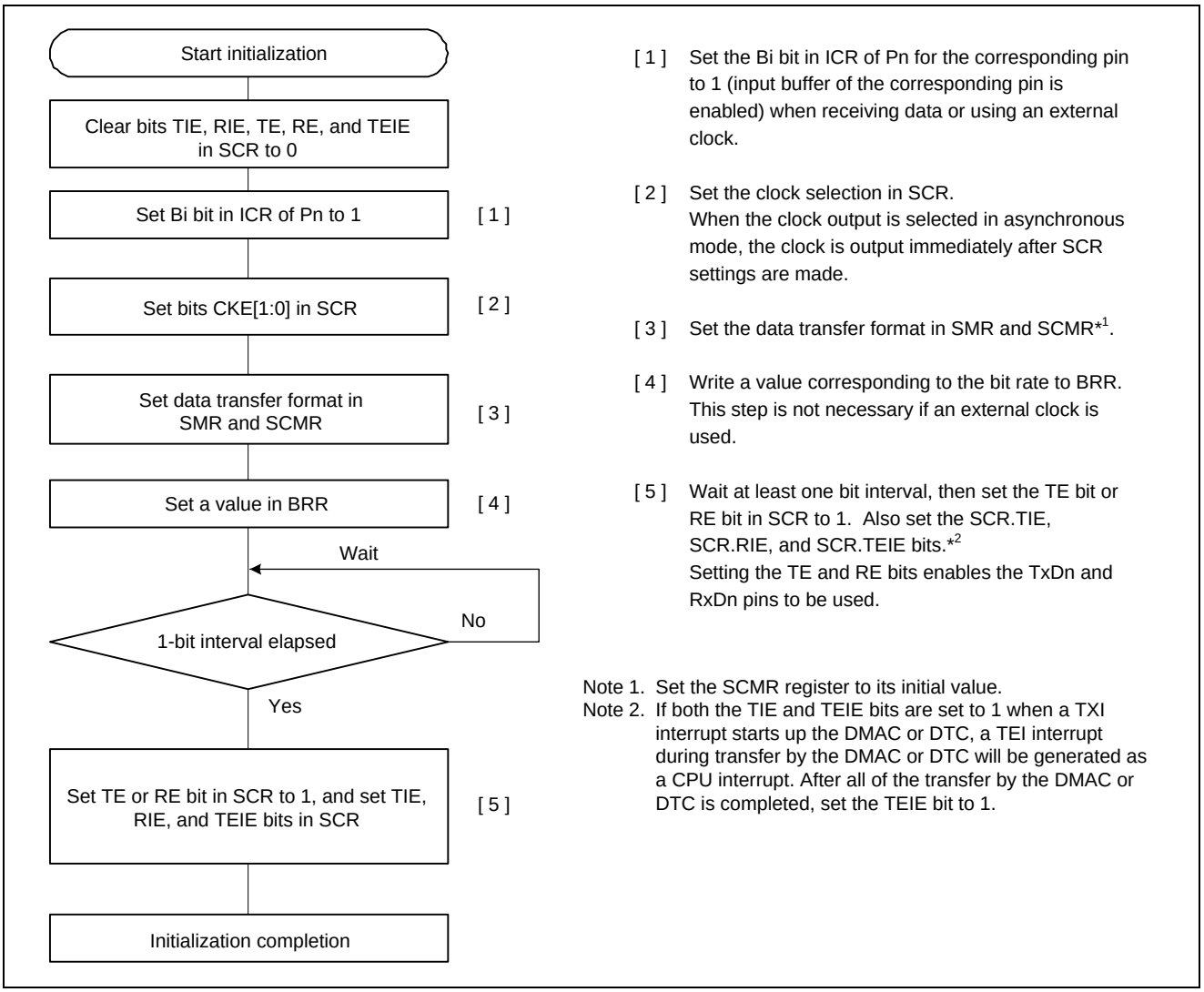



**Figure 20.7 Sample SCI Initialization Flowchart (Asynchronous Mode)**



Downloaded from Arrow.com.

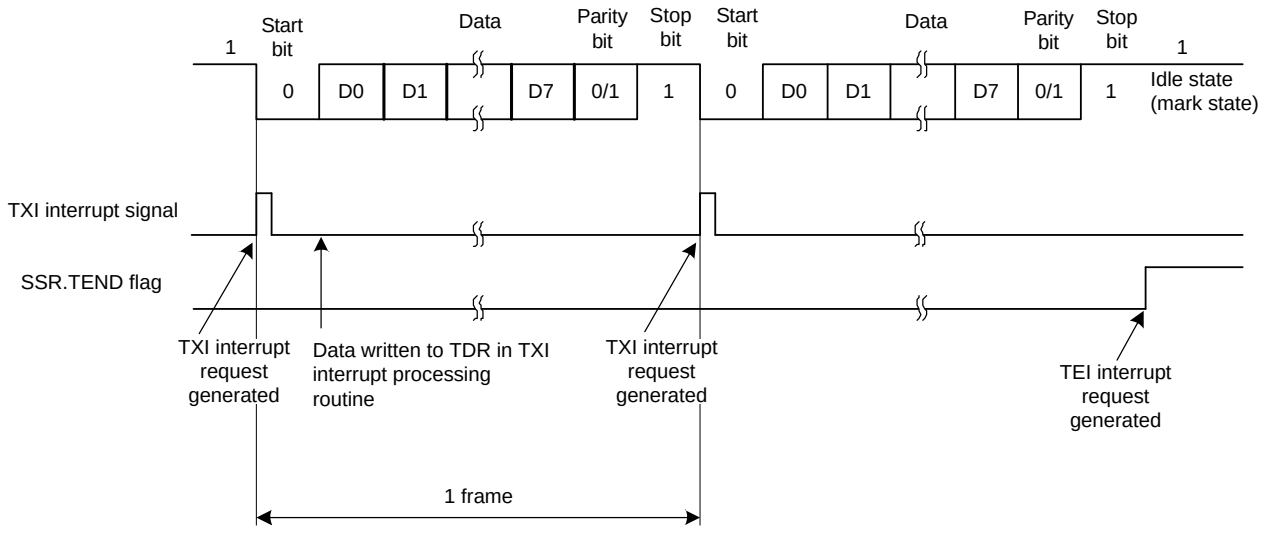
### 20.3.5 Serial Data Transmission (Asynchronous Mode)

Figure 20.8 shows an example of the operation for serial transmission in asynchronous mode.

In serial transmission, the SCI operates as described below.

1. The SCI transfers data from TDR to TSR when data is written to TDR in the TXI interrupt processing routine. The TXI interrupt request at the beginning of transmission is generated when the TE bit in SCR is set to 1 after the TIE bit in SCR is set to 1 or when these two bits are set to 1 simultaneously by a single instruction.
2. After transferring data from TDR to TSR, the SCI starts transmission. If the TIE bit is set to 1 at this time, an SCR.TXI interrupt request is generated. Continuous transmission is enabled by writing the next transmit data to TDR in the TXI interrupt processing routine before transmission of the current transmit data is completed.
3. Data is sent from the TxDn pin in the following order: start bit, transmit data, parity bit (may be omitted depending on the format), and stop bit.
4. The SCI checks for updating of (writing to) the TDR at the time of stop bit output.
5. When TDR is updated, the next transmit data is transferred from TDR to TSR, the stop bit is sent, and then serial transmission of the next frame is started.
6. If TDR is not updated, the TEND flag in SSR is set to 1, the stop bit is sent, and then the mark state is entered in which high is output. If the TEND flag in SSR is set to 1 at this time, a TEI interrupt request is generated.

Figure 20.9 shows a sample flowchart for serial transmission in asynchronous mode.

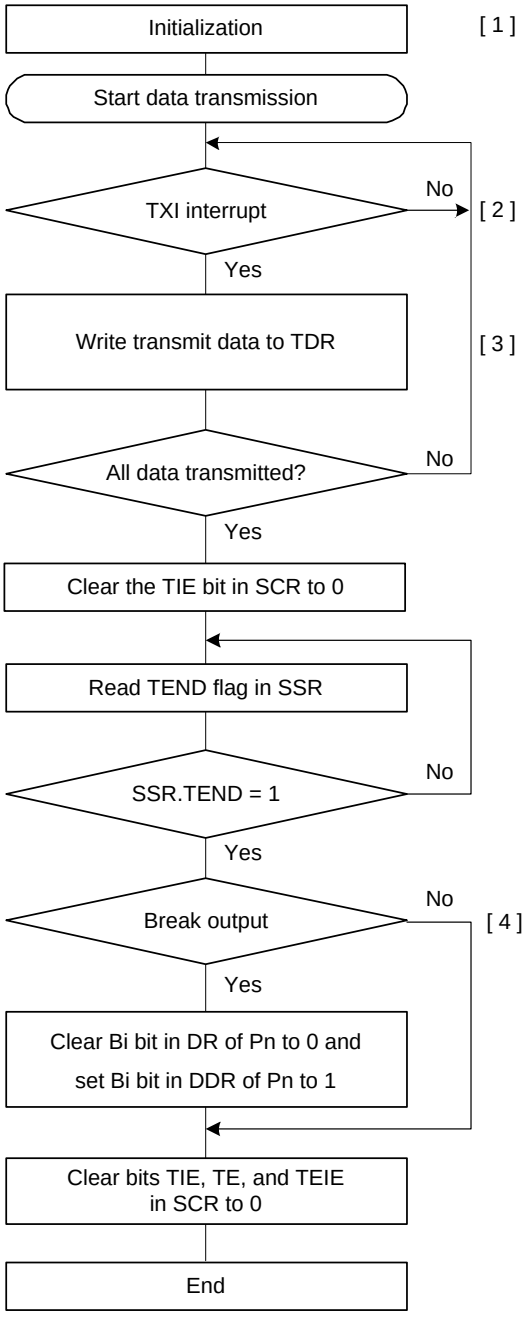



Note: * For the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU).

**Figure 20.8 Example of Operation for Serial Transmission in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit)**



Downloaded from Arrow.com.




[ 1 ] SCIn initialization:
The TxDn pin is automatically designated as the transmit data output pin. After the TE bit in SCR is set to 1, a 1 is output for a frame, and transmission is enabled.

[ 2 ] Transmit data write to TDR upon accepting a TXI interrupt request:
When transmit data is transferred from TDR to TSR, a transmit data empty interrupt (TXI) request is generated. Write transmit data to TDR once in the TXI interrupt processing routine.

[ 3 ] Serial transmission continuation procedure:
To continue serial transmission, write transmit data to TDR once upon accepting a TXI interrupt request. Transmit data can also be written to TDR by initiating the DMAC or DTC.

[ 4 ] Break output at the end of serial transmission:
To output a break in serial transmission, set the Bi bit in DDR of Pn for the port corresponding to the TxDn pin to 1 (output port), clear the Bi bit in DR of Pn to 0 (low level output), then clear the TE bit in SCR to 0.

**Figure 20.9 Example of Serial Transmission Flowchart in Asynchronous Mode**



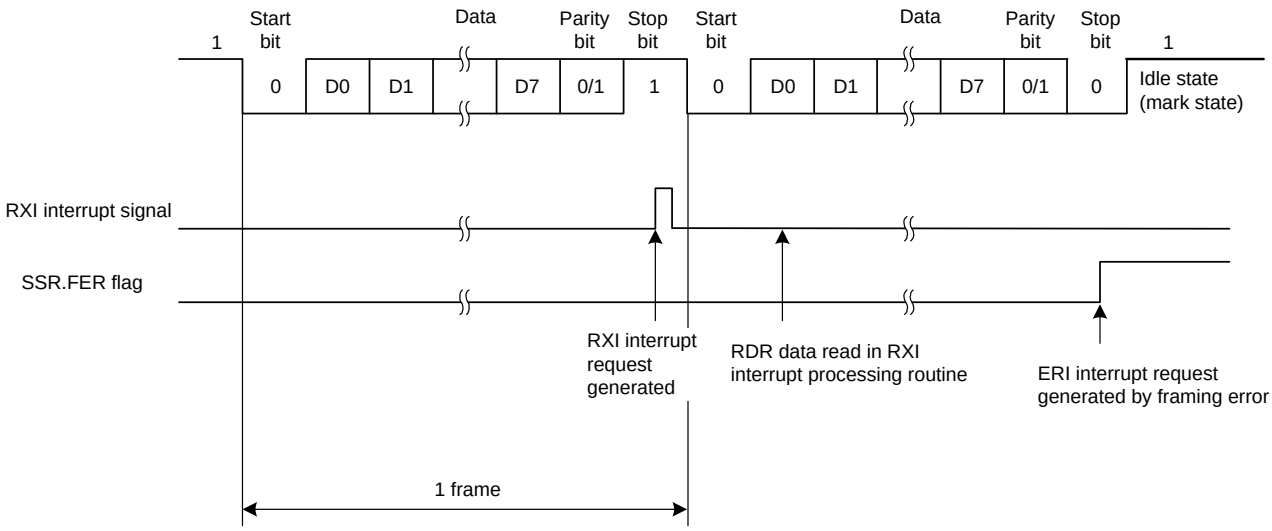
Downloaded from Arrow.com.

### 20.3.6 Serial Data Reception (Asynchronous Mode)

Figure 20.10 shows an example of the operation for serial data reception in asynchronous mode.

In serial data reception, the SCI operates as described below.

1. When the SCI monitors the communications line and detects a start bit, it performs internal synchronization, stores receive data in RSR, and checks the parity bit and stop bit.
2. If an overrun error occurs, the ORER flag in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR.
3. If a parity error is detected, the PER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated.
4. If a framing error (when the stop bit is 0) is detected, the FER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated.
5. When reception finishes successfully, receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an RXI interrupt request is generated. Continuous reception is enabled by reading the receive data transferred to RDR in this RXI interrupt processing routine before reception of the next receive data is completed.




Note: * For the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU).

**Figure 20.10 Example of SCI Operation for Serial Reception in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit)**



Downloaded from Arrow.com.

Table 20.14 shows the states of the SSR status flags and receive data handling when a receive error is detected.

If a receive error is detected, an ERI interrupt request is generated but an RXI interrupt request is not generated. Data reception cannot be resumed while the receive error flag is set to 1. Accordingly, clear the ORER, FER, and PER bits to 0 before resuming reception. Moreover, be sure to read the RDR during overrun error processing.

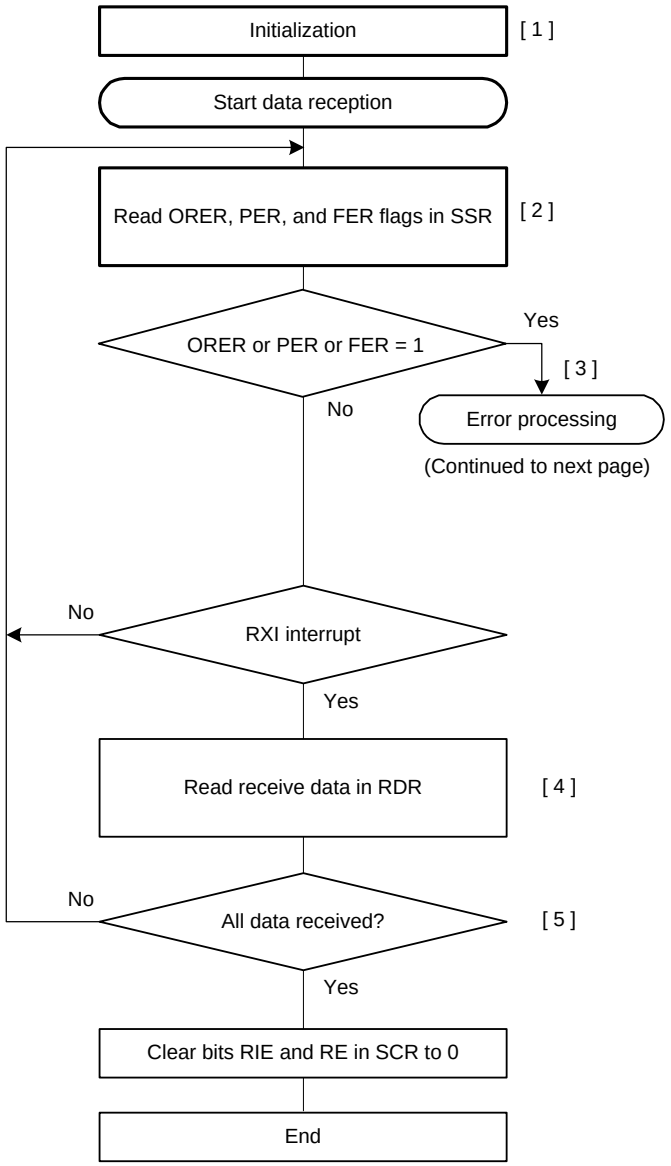
Figure 20.11 shows samples of flowcharts for serial data reception.

**Table 20.14 SSR Status Flags and Receive Data Handling**

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



Downloaded from Arrow.com.




[ 1 ] SCIn initialization:
The RxDn pin is automatically designated as the receive data input pin.

[ 2 ] [ 3 ] Receive error processing and break detection:
If a receive error occurs, an ERI interrupt is generated. After performing the appropriate error processing, be sure to clear the ORER, PER, and FER flags to 0. Reception cannot be resumed if any of these flags is set to 1. In the case of a framing error, a break can be detected by reading the value of the input port corresponding to the RxDn pin.

[ 4 ] Read the receive data in RDR once in the RXI interrupt processing routine.

[ 5 ] Serial reception continuation procedure:
To continue serial reception, before the stop bit of the current frame is received, read data from RDR in the RXI interrupt processing routine. The RDR data can also be read by initiating the DMAC or DTC.

**Figure 20.11 Example of Serial Reception Flowchart (1) (Asynchronous Mode)**



Downloaded from Arrow.com.

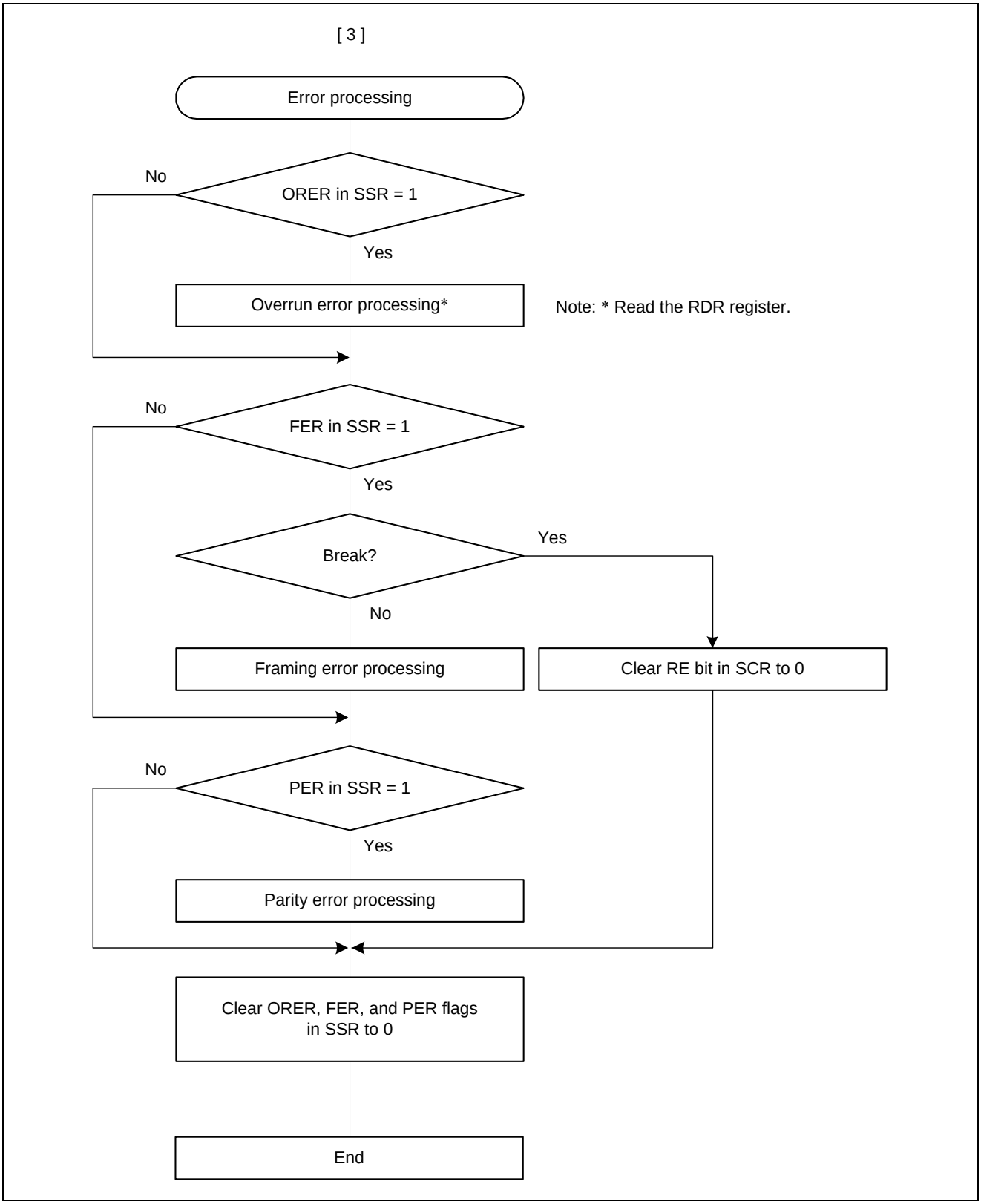



**Figure 20.12 Example of Serial Reception Flowchart (2) (Asynchronous Mode)**



Downloaded from Arrow.com.

## 20.4 Operation in Clock Synchronous Mode

Figure 20.13 shows the data format for clock synchronous serial data communications.

In clock synchronous mode, data is transmitted or received in synchronization with clock pulses. One character in transfer data consists of 8-bit data. In clock synchronous mode, no parity bit can be added.

In data transmission, the SCI outputs data from one falling edge of the synchronization clock to the next. In data reception, the SCI receives data in synchronization with the rising edge of the synchronization clock. After 8-bit data is output, the transmission line holds the MSB output state.

Inside the SCI, the transmitter and receiver are independent units, enabling full-duplex communications by use of a common clock. Both the transmitter and the receiver also have a double-buffered structure, so that the next transmit data can be written during transmission or the previous receive data can be read during reception, enabling continuous data transfer.

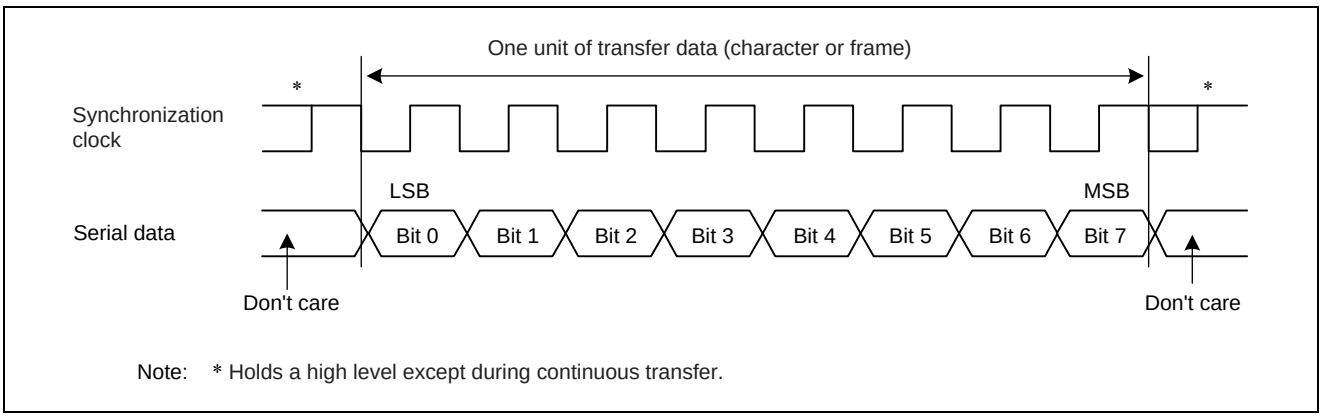



**Figure 20.13 Data Format in Clock Synchronous Serial Communications (LSB-First)**

### 20.4.1 Clock

Either an internal clock generated by the on-chip baud rate generator or an external synchronization clock input at the SCKn pin can be selected, according to the setting of the CKE[1:0] bits in SCR.

When the SCI is operated on an internal clock, the synchronization clock is output from the SCKn pin. Eight synchronization clock pulses are output in the transfer of one character, and when no transfer is performed the clock is held high. However, the synchronization clock is continuously output only during data reception until an overrun error occurs or the RE bit in SCR is cleared to 0.



Downloaded from Arrow.com.

## 20.4.2 SCI Initialization (Clock Synchronous Mode)

Before transmitting and receiving data, start by writing the initial value "00h" to the SCR and then continue through the procedure for SCI given in the sample flowchart (figure 20.14). Whenever the operating mode or transfer format is changed, the SCR must be initialized before the change is made.

Note that clearing the SCR.RE bit to 0 initializes neither the ORER, FER, and PER flags in SSR nor RDR.

Moreover, note that switching the value of the SCR.TE bit from 1 to 0 or 0 to 1 while the SCR.TIE bit is 1 leads to the generation of a TXI interrupt request.

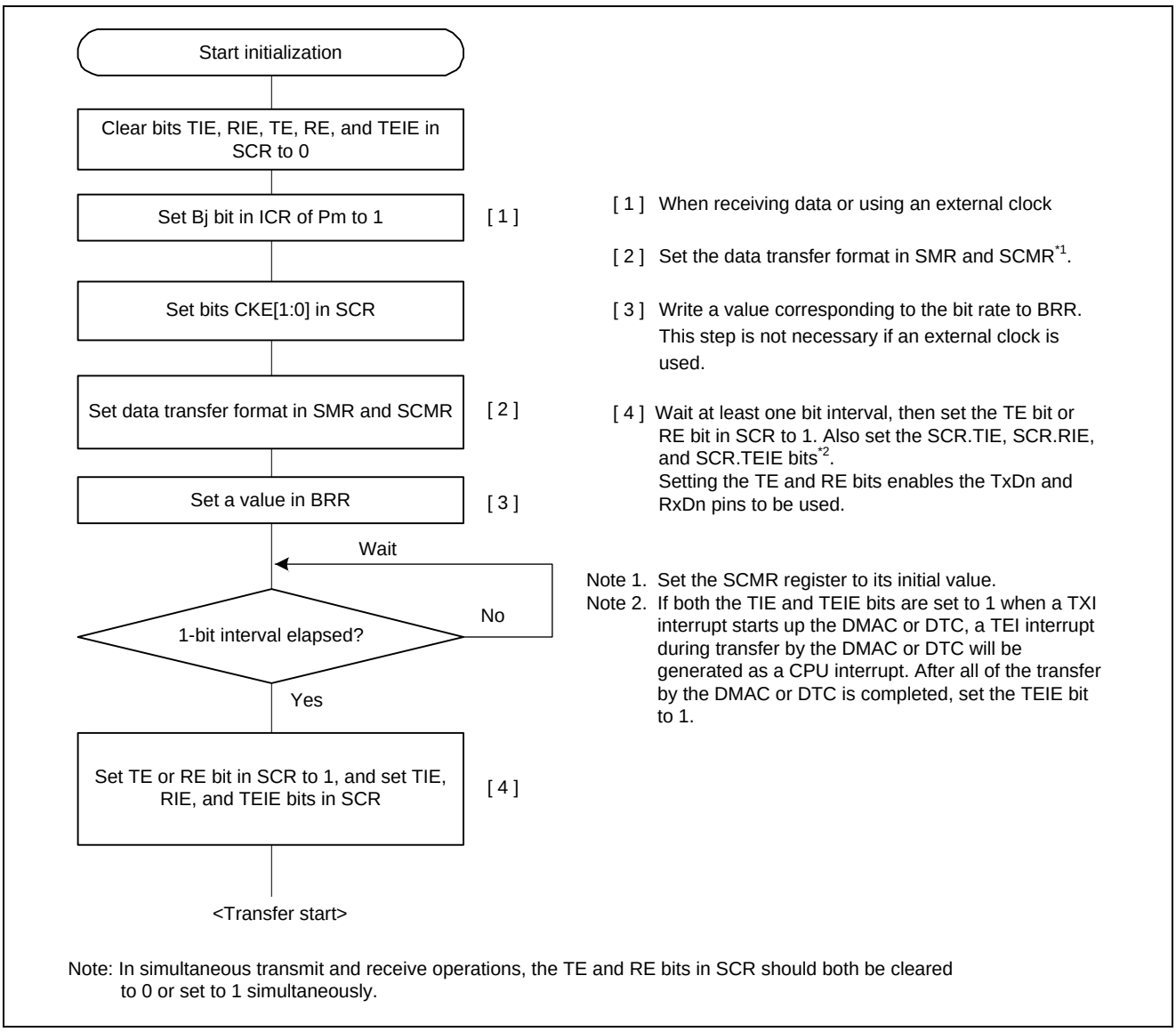



**Figure 20.14 Example of SCI Initialization Flowchart (Clock Synchronous Mode)**



Downloaded from Arrow.com.

### 20.4.3 Serial Data Transmission (Clock Synchronous Mode)

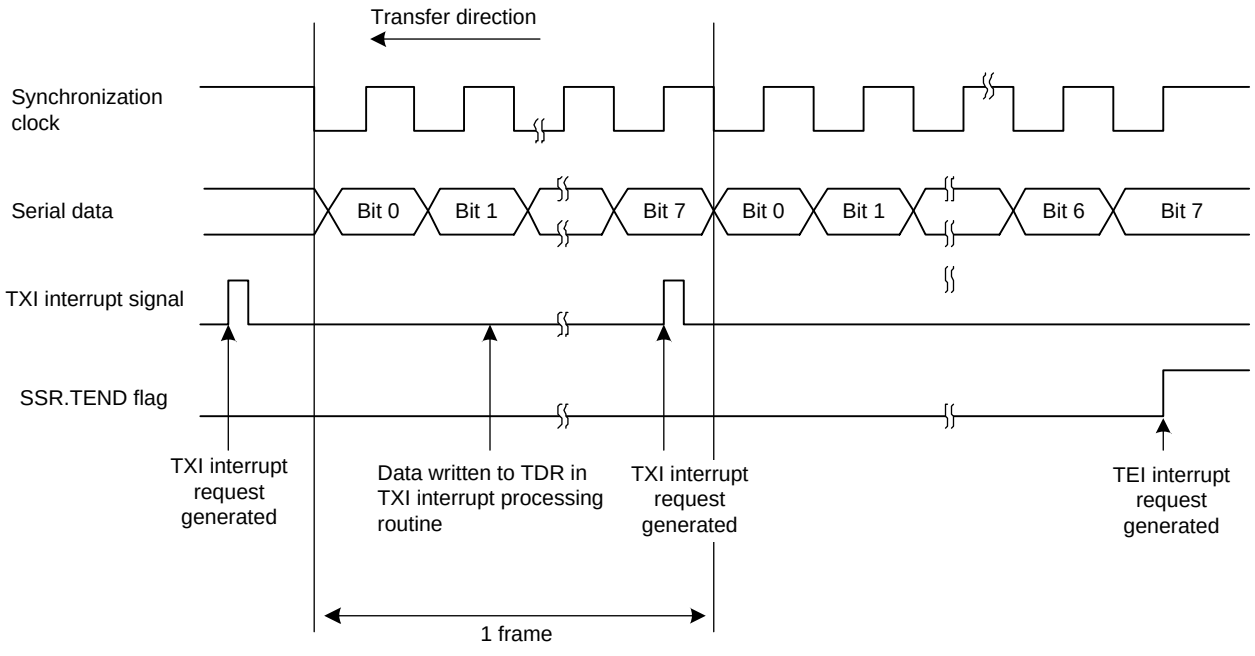
Figure 20.15 shows an example of the operation for serial transmission in clock synchronous mode.

In serial data transmission, the SCI operates as described below.

1. The SCI transfers data from TDR to TSR when data is written to TDR in the TXI interrupt processing routine. The TXI interrupt request at the beginning of transmission is generated when the TE bit in SCR is set to 1 after the TIE bit in SCR is set to 1 or when these two bits are set to 1 simultaneously by a single instruction.
2. After transferring data from TDR to TSR, the SCI starts transmission. When the TIE bit is set to 1 at this time, a TXI interrupt request is generated. Continuous transmission is enabled by writing the next transmit data to TDR in this TXI interrupt processing routine before transmission of the current transmit data has finished.
3. 8-bit data is sent from the TxDn pin in synchronization with the output clock when clock output mode has been specified and in synchronization with the input clock when use of an external clock has been specified.
4. The SCI checks for updating of (writing to) the TDR at the time of stop bit output.
5. When TDR is updated, the next transmit data is transferred from TDR to TSR, and serial transmission of the next frame is started.
6. If TDR is not updated, set the SSR flag in TEND to 1 and the TxDn pin retains the output state of the last bit. If the TEIE bit in SCR is set to 1 at this time, a TEI interrupt request is generated. The SCKn pin is held high.

Figure 20.16 shows a sample flowchart of serial data transmission.

Transmission will not start while a receive error flag (ORER, FER, or PER in SSR) is set to 1. Be sure to clear the receive error flags to 0 before starting transmission. Note that clearing the RE bit in SCR to 0 does not clear the receive error flags.

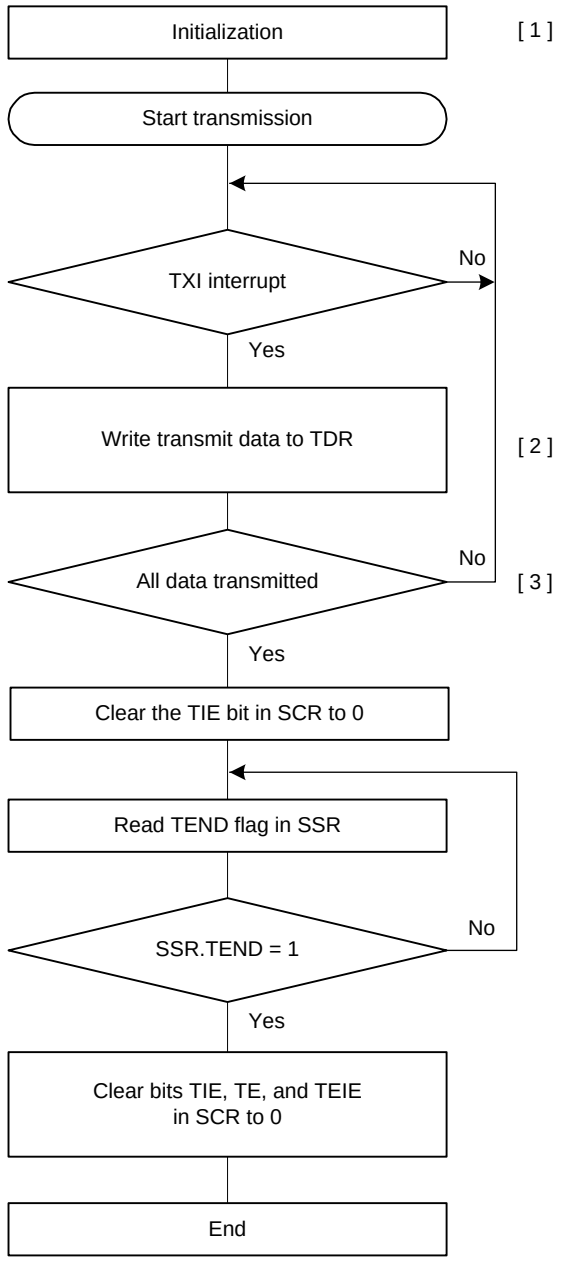



Note: * For the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU).

**Figure 20.15 Example of Operation for Serial Transmission in Clock Synchronous Mode**



Downloaded from Arrow.com.




[ 1 ] SCIn initialization:
The TxDn pin is automatically designated as the transmit data output pin.

[ 2 ] Transmit data write to TDR upon accepting a TXI interrupt request:
When transmit data is transferred from TDR to TSR, a transmit data empty interrupt (TXI) request is generated.
Write transmit data to TDR once upon accepting a transmit data empty interrupt (TXI) request.

[ 3 ] Serial transmission continuation procedure:
To continue serial transmission, write transmit data to TDR upon accepting a TXI interrupt request. Transmit data can also be written to TDR by initiating the DMAC or DTC when a TXI interrupt request is accepted.

**Figure 20.16 Example of Serial Transmission Flowchart (Clock Synchronous Mode)**



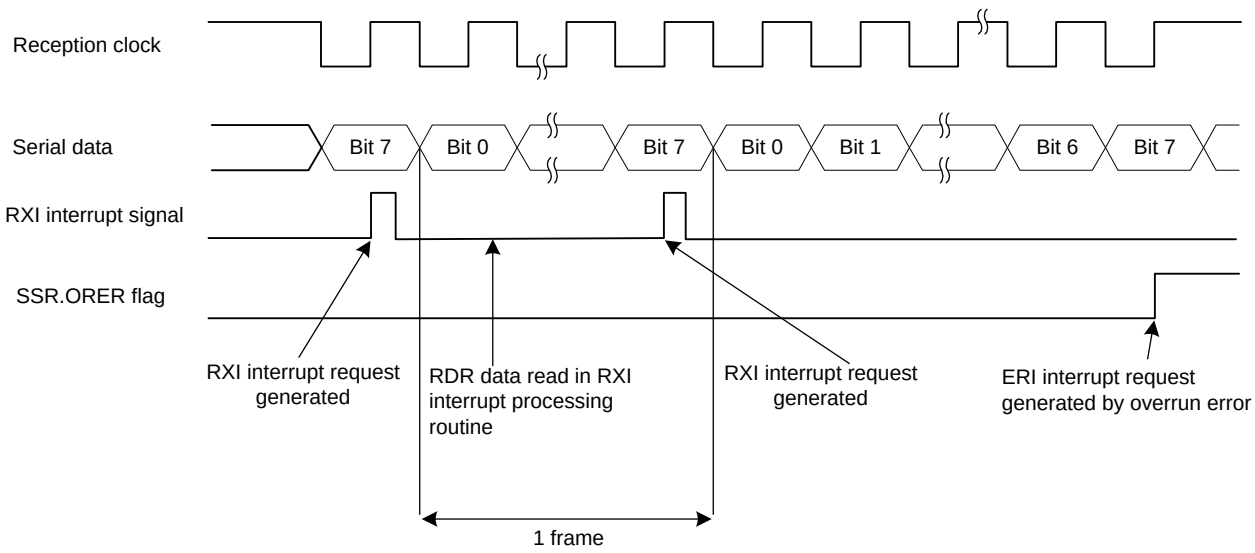
Downloaded from Arrow.com.

### 20.4.4 Serial Data Reception (Clock Synchronous Mode)

Figure 20.17 shows an example of SCI operation for serial reception in clock synchronous mode.

In serial data reception, the SCI operates as described below.

1. The SCI performs internal initialization and starts receiving data in synchronization with a synchronization clock input or output, and stores the receive data in RSR.
   If an overrun error occurs, the ORER bit in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR.
2. When reception finishes successfully, receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an RXI interrupt request is generated. Continuous reception is enabled by reading the receive data transferred to RDR in this RXI interrupt processing routine before reception of the next receive data is completed.




Note: * For the corresponding interrupt vector number, see section 10, Interrupt Control Unit (ICU).

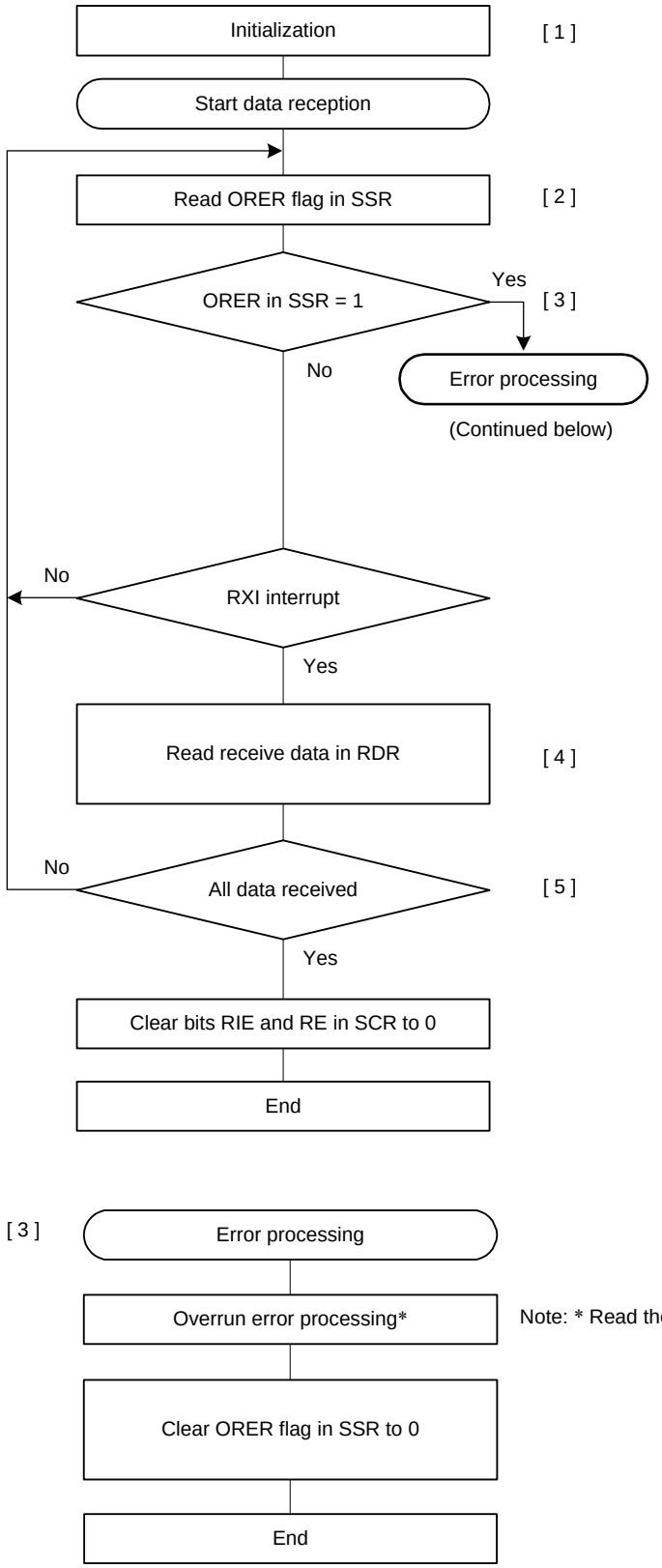
**Figure 20.17 Example of Operation for Serial Reception in Clock Synchronous Mode**

Data transfer cannot be resumed while a receive error flag is set to 1. Accordingly, clear the ORER, FER, and PER bits in SSR to 0 before resuming reception. Moreover, be sure to read the RDR during overrun error processing.

Figure 20.18 shows a sample flowchart for serial data reception.



Downloaded from Arrow.com.




[ 1 ] SCIn initialization:
Set the port to be used as the RxDn pin to an input port.

[ 2 ] [ 3 ] Receive error processing:
If a receive error occurs, read the ORER flag in SSR, perform the relevant error processing, and then clear the ORER flag to 0. Data reception cannot be resumed while the ORER flag is set to 1.

[ 4 ] Read the receive data in RDR once upon accepting an RXI interrupt request.

[ 5 ] Serial reception continuation procedure:
To continue serial reception, before the MSB (bit 7) of the current frame is received, finish reading the receive data in RDR. The RDR data can also be read by initiating the DMAC or DTC when an RXI interrupt request is accepted.

Note: * Read the RDR register.

**Figure 20.18 Example of Serial Reception Flowchart (Clock Synchronous Mode)**



Downloaded from Arrow.com.

### 20.4.5 Simultaneous Serial Data (Clock Synchronous Mode)

Figure 20.19 shows a sample flowchart for simultaneous serial transmit and receive operations in clock synchronous mode.

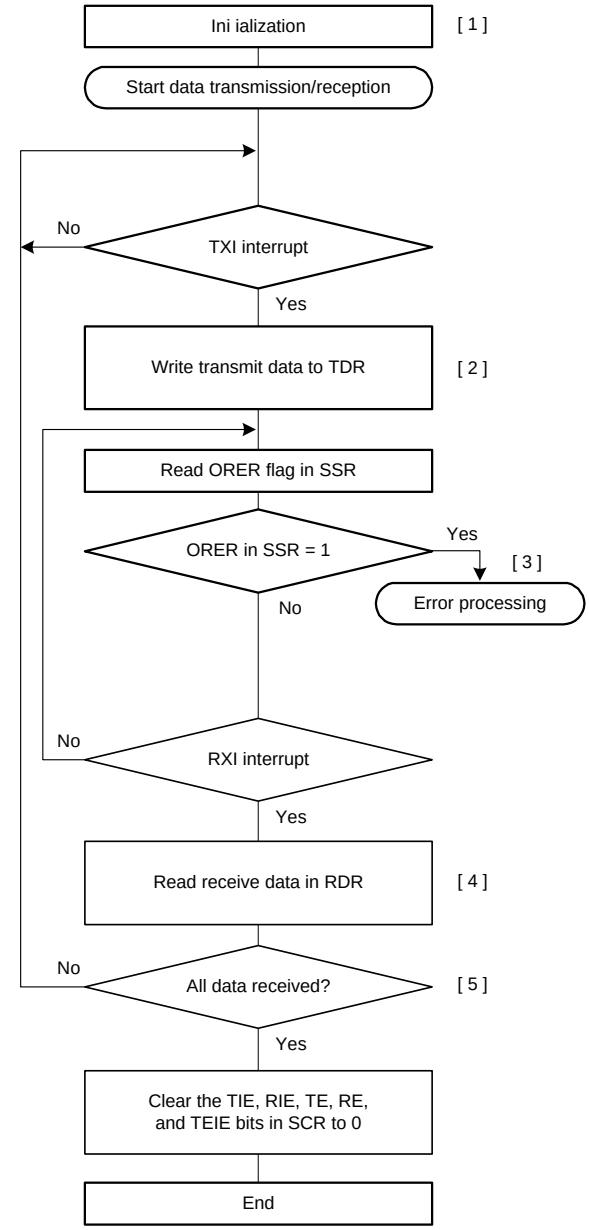
After initializing the SCI, the following procedure should be used for simultaneous serial data transmit and receive operations.

To switch from transmit mode to simultaneous transmit and receive mode, check that the SCI has finished by reading that the TEND flag in SSR in SSR is set to 1, and then initialize the SCR register. Then set the TIE, RIE, TE, RE, and TEIE bits in SCR to 1 simultaneously by a single instruction.

To switch from receive mode to simultaneous transmit and receive mode, check that the SCI has finished reception, and then clear the RIE and RE bits to 0. Then check that the receive error flags (ORER, FER, and PER in SSR) are cleared to 0, and then the TIE, RIE, TE, RE, and TEIE bits in SCR to 1 simultaneously by a single instruction.



Downloaded from Arrow.com.




[ 1 ] SCI initialization:
The TxDn pin is designated as the transmit data output pin, and the RxDn pin is designated as the receive data input pin, enabling simultaneous transmit/receive operations.

[ 2 ] Transmit data write:
Write transmit data to TDR once upon accep ing a TXI interrupt request.

[ 3 ] Receive error processing:
If a receive error occurs, read the ORER flag in SSR, perform the relevant error processing, and then clear the ORER flag to 0. Data reception cannot be resumed while the ORER flag is set to 1.

[ 4 ] Receive data read:
Read the receive data in RDR once upon accepting an RXI interrupt request.

[ 5 ] Serial transmission/reception con inuation procedure:
To continue serial transmission/reception, before the MSB (bit 7) of the current frame is received, finish reading the receive data in RDR by the RXI interrupt. Also, before the MSB (bit 7) of the current frame is transmitted, write data to TDR by the TX interrupt.
Transmit data can also be written to TDR by initiating the DMAC or DTC when a transmit data empty interrupt (TXI) request is accepted. Similarly, the RDR data can also be read by initiating the DMAC or DTC when a receive data full interrupt (RXI) request is accepted.

Note: When switching from transmit or receive operation to simultaneous transmit and receive operations, first clear the TIE, RIE, TE, RE, and TEIE bits in SCR to 0, and then set these bits to 1 simultaneously.

**Figure 20.19 Example of Simultaneous Serial Transmission/Reception Flowchart (Clock Synchronous Mode)**



Downloaded from Arrow.com.

## 20.5 Operation in Smart Card Interface Mode

The SCI supports the smart card (IC card) interface conforming to the ISO/IEC 7816-3 (Identification Card) standard, as an extended function of the SCI.

Smart card interface mode can be selected using the appropriate register.

### 20.5.1 Sample Connection

Figure 20.20 shows a sample connection between the smart card (IC card) and this LSI.

As in the figure, since this LSI communicates with the IC card using a single transmission line, interconnect the TxDn and RxDn pins and pull up the data transmission line to Vcc using a resistor.

Setting the TE and RE bits in SCR to 1 with the IC card disconnected enables closed transmission/reception allowing self diagnosis.

To supply the IC card with the clock pulses generated by the SCI, input the SCKn pin output to the CLK pin of the IC card.

The output port of this LSI can be used to output a reset signal.

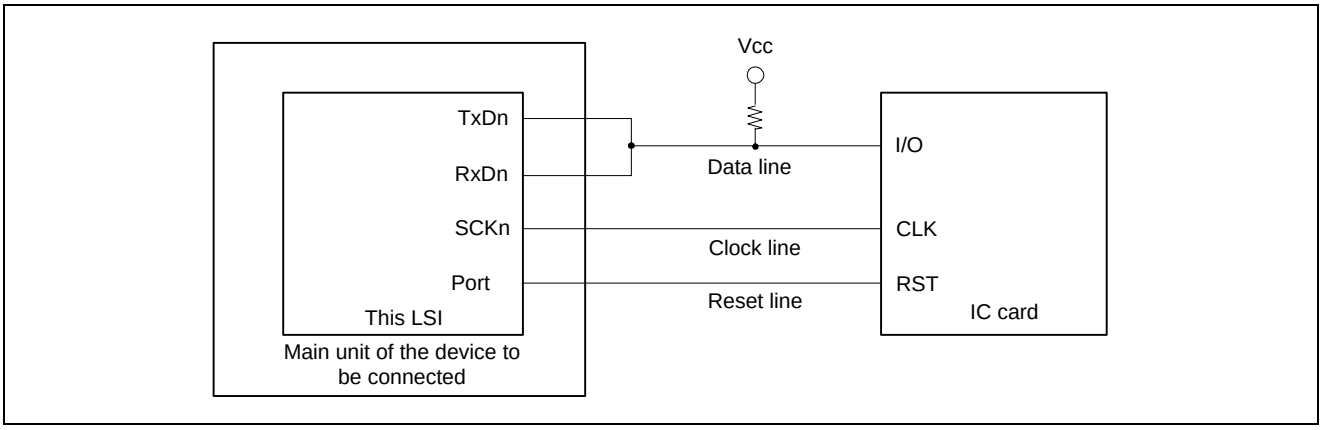



**Figure 20.20 Sample Connection with a Smart Card (IC Card)**



Downloaded from Arrow.com.

### 20.5.2 Data Format (Except in Block Transfer Mode)

Figure 20.21 shows the data transfer formats in smart card interface mode.

- One frame consists of 8-bit data and a parity bit in asynchronous mode.
- During transmission, at least 2 etu (elementary time unit: time required for transferring one bit) is secured as a guard time from the end of the parity bit until the start of the next frame.
- If a parity error is detected during reception, a low-level error signal is output for 1 etu after 10.5 etu has passed from the start bit.
- If an error signal is sampled during transmission, the same data is automatically re-transmitted after at least 2 etu.

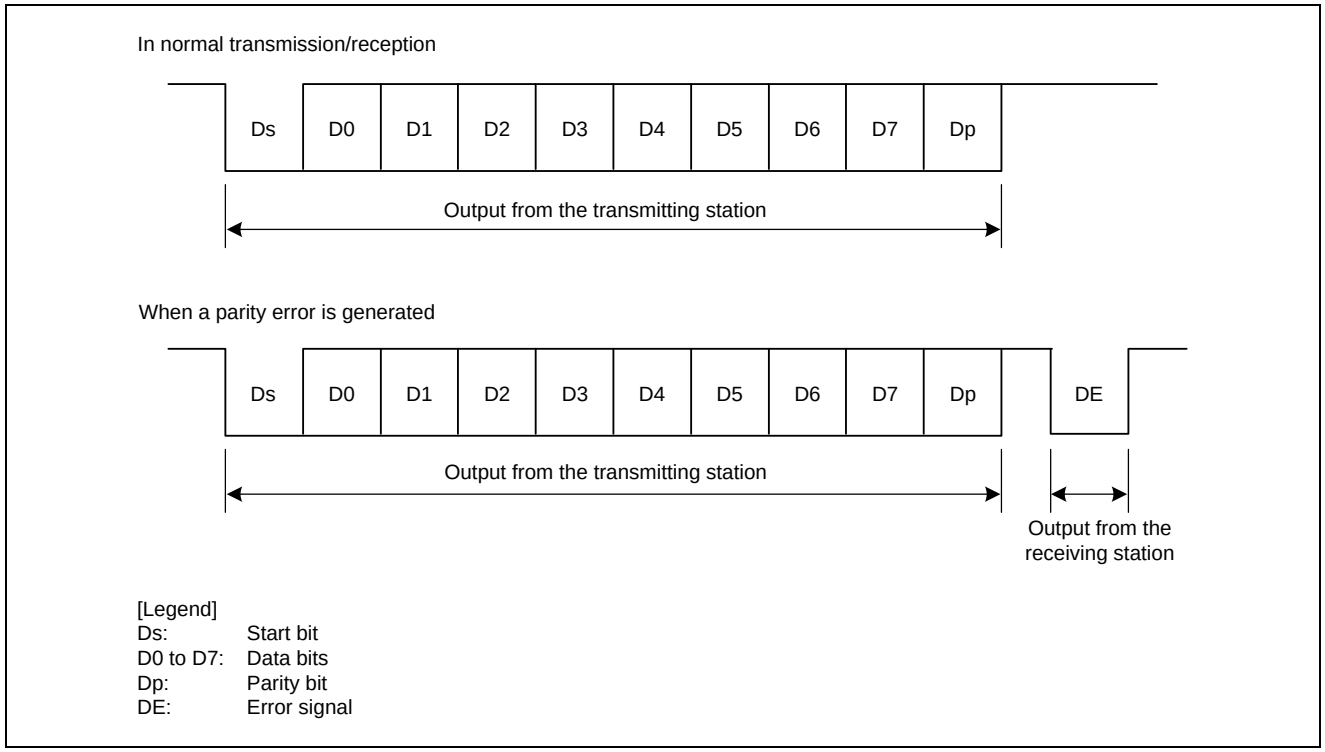



**Figure 20.21 Data Formats in Smart Card Interface Mode**



Downloaded from Arrow.com.

For communications with IC cards of the direct convention type and inverse convention type, follow the procedure below.

### (1) Direct Convention Type

For the direct convention type, logic levels 1 and 0 correspond to states Z and A, respectively, and data is transferred with LSB-first as the start character, as shown in figure 20.22. Therefore, data in the start character in the figure is 3Bh.

When using the direct convention type, write 0 to both the SDIR and SINV bits in SCMR. Write 0 to the PM bit in SMR in order to use even parity, which is prescribed by the smart card standard.

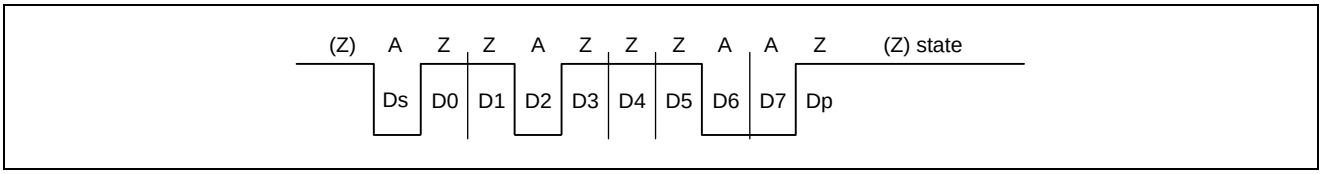



**Figure 20.22 Direct Convention (SDIR in SCMR = 0, SINV in SCMR =0, PM in SMR = 0)**

### (2) Inverse Convention Type

For the inverse convention type, logic levels 1 and 0 correspond to states A and Z, respectively and data is transferred with MSB-first as the start character, as shown in figure 20.23. Therefore, data in the start character in the figure is 3Fh.

When using the inverse convention type, write 1 to both the SDIR and SINV bits in SCMR. The parity bit is logic level 0 to produce even parity, which is prescribed by the smart card standard, and corresponds to state Z. Since the SNIV bit of this LSI only inverts data bits D7 to D0, write 1 to the PM bit in SMR to invert the parity bit for both transmission and reception.

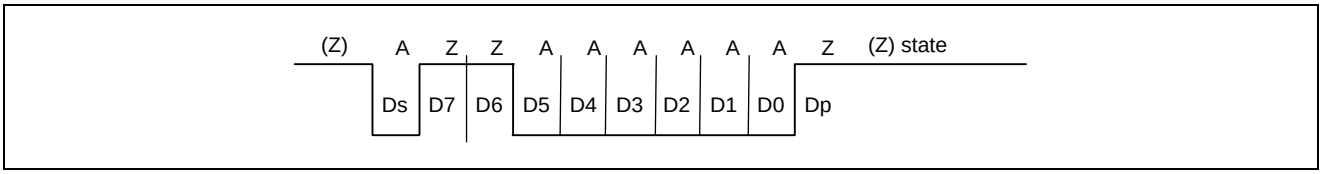



**Figure 20.23 Inverse Convention (SDIR in SCMR = 1, SINV in SCMR =1, PM in SMR = 1)**

## 20.5.3 Block Transfer Mode

Block transfer mode is different from normal smart card interface mode in the following respects.

- Even if a parity error is detected during reception, no error signal is output. Since the PER bit in SSR is set by error detection, clear the PER bit before receiving the parity bit of the next frame.
- During transmission, at least 1 etu is secured as a guard time from the end of the parity bit until the start of the next frame.
- Since the same data is not re-transmitted during transmission, the SSR.TEND flag is set 11.5 etu after transmission start.
- In block transfer mode, the ERS flag in SSR indicates the error signal status as in normal smart card interface mode, but the flag is always read as 0 because no error signal is transferred.



Downloaded from Arrow.com.

### 20.5.4 Receive Data Sampling Timing and Reception Margin

Only the internal clock generated by the on-chip baud rate generator can be used as a transfer clock in smart card interface mode.

In this mode, the SCI can operate on a base clock with a frequency of 32, 64, 372, 256, 93, 128, 186, or 512 times the bit rate according to the settings of the BCP2 bit in SCMR and BCP[1:0] bits in SMR (the frequency is always 16 times the bit rate in normal asynchronous mode).

For data reception, the falling edge of the start bit is sampled with the base clock to perform internal synchronization. For data reception, the falling edge of the start bit is sampled with the base clock to perform internal synchronization. Receive data is sampled on the 16th, 32nd, 186th, 128th, 46th, 64th, 93rd, and 256th rising edges of the base clock so that it can be latched at the middle of each bit as shown in figure 20.24. The reception margin here is determined by the following formula.

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

[Legend]
M: Reception margin (%)
N: Ratio of bit rate to clock (N = 32, 64, 372, 256)
D: Duty cycle of clock (D = 0 to 1.0)
L: Frame length (L = 10)
F Absolute value of clock frequency deviation

Assuming values of F = 0, D = 0.5, and N = 372 in the above formula, the reception margin is determined by the formula below.

$$M = \{0.5 - 1/(2 \times 372)\} \times 100\ (\%) = 49.866\%$$



Downloaded from Arrow.com.

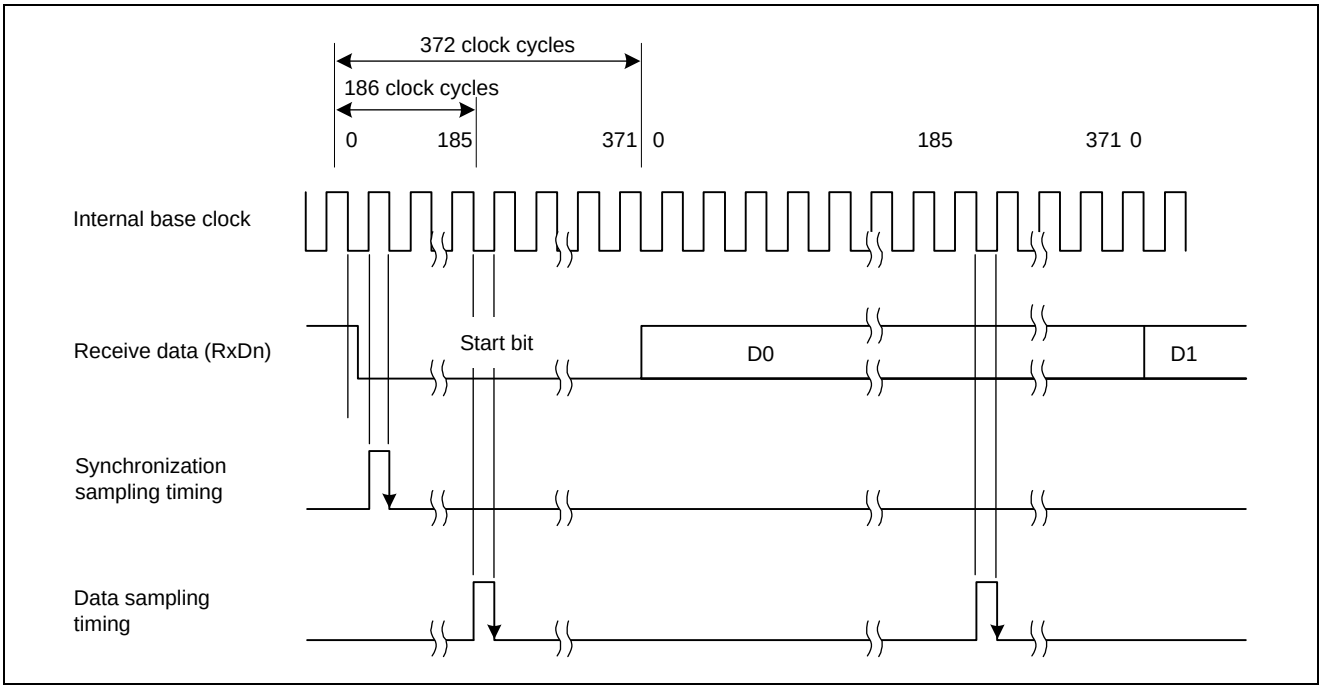



**Figure 20.24 Receive Data Sampling Timing in Smart Card Interface Mode (When Clock Frequency is 372 Times the Bit Rate)**



Downloaded from Arrow.com.

### 20.5.5 Initialization of the SCI

Before transmitting and receiving data, initialize the SCI using the following procedure. Initialization is also necessary before switching from transmission to reception and vice versa.

1. Write the initial value "00h" to the SCR.
2. Set the Bj bit (j = 0 to 7) in ICR of Pm (m = 0 to 9, A to E) of the corresponding pin to 1.
3. Set the error flags ORER, ERS, and PER in SSR to 0.
4. Set bits GM, BLK, OE, BCP[1:0], and CKS[1:0] in SMR and the BCP2 bit in SCMR appropriately. Also set the PE bit in SMR to 1.
5. Set bits SDIR, SINV, and SMIF in SCMR appropriately. Also set the Bj bit in DDR of Pm corresponding to the TxDn pin to 0. Then the TxDn and RxDn pins are changed from port pins to SCI pins, placing the pins into high impedance state.
6. Set the value corresponding to the bit rate in BRR.
7. Set the CKE[1:0] bits in SCR appropriately, and set bits TIE, RIE, TE, RE, and TEIE in SCR to 0 at the same time. When the CKE0 bit is set to 1, the SCKn pin is allowed to output clock pulses.
8. Wait for at least a 1-bit interval, and then set the TIE, RIE, TE, and RE bits in SCR. Setting the TE and RE bits to 1 simultaneously is prohibited except for self diagnosis.

To change reception mode to transmission mode, first check that reception has completed, and then initialize the SCI. At the end of initialization, set TE = 1 and RE = 0. Reception completion can be verified by reading the RXI request, ORER, or PER flag in SSR.

To change transmission mode to reception mode, first check that transmission has completed, and then initialize the SCI. At the end of initialization, set TE = 0 and RE = 1. Transmission completion can be verified by reading the SSR.TEND flag.



Downloaded from Arrow.com.

### 20.5.6 Serial Data Transmission (Except in Block Transfer Mode)

Serial data transmission in smart card interface mode (except in block transfer mode) is different from that in normal serial communications interface mode in that an error signal is sampled and data can be re-transmitted. Figure 20.25 shows the data re-transfer operation during transmission.

1. When an error signal from the receiving end is sampled after one-frame data has been transmitted, the ERS flag in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Clear the ERS flag to 0 before the next parity bit is sampled.
2. For a frame in which an error signal is received, the SSR.TEND flag is not set. Data is re-transferred from TDR to TSR allowing automatic data retransmission.
3. If no error signal is returned from the receiving end, the ERS flag in SSR is not set to 1.
4. In this case, the SCI judges that transmission of one-frame data (including retransfer) has been completed, and the SSR.TEND flag is set. If the TIE bit in SCR is set to 1 at this time, a TXI interrupt request is generated. Writing transmit data to TDR starts transmission of the next data.

Figure 20.27 shows a sample flowchart of serial transmission. All the processing steps are automatically performed using a TXI interrupt request to activate the DTC or DMAC.

When the SSR.TEND flag is set to 1 in transmission, if the TIE bit in SCR is set to 1, a TXI interrupt request is generated. The DTC or DMAC is activated by a TXI interrupt request if the TXI interrupt request is specified as a source of DTC or DMAC activation beforehand, allowing transfer of transmit data. The TEND flag is automatically cleared to 0 when the DTC or DMAC transfers the data.

If an error occurs, the SCI automatically re-transmits the same data. During this retransmission, the TEND flag is kept to 0 and the DTC or DMAC is not activated. Therefore, the SCI and DTC or DMAC automatically transmit the specified number of bytes, including retransmission in the case of error occurrence. However, since the ERS flag is not automatically cleared; set the RIE bit to 1 beforehand to enable an ERI interrupt request to be generated at error occurrence, and clear the ERS flag to 0.

When transmitting/receiving data using the DTC or DMAC, be sure to make settings to enable the DTC or DMAC before making SCI settings.

For DTC or DMAC settings, see section 12, DMA Controller (DMAC) and section 13, Data Transfer Controller (DTC).

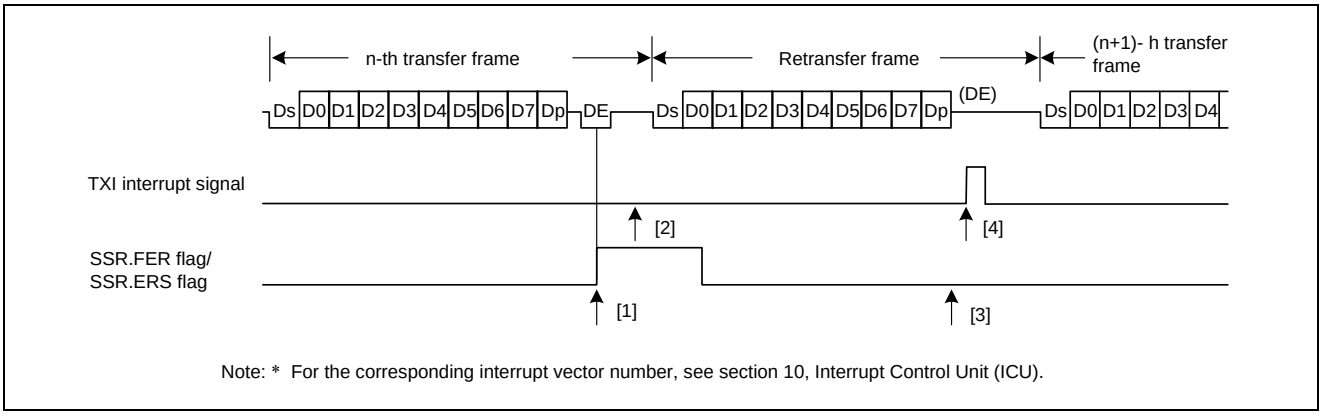



**Figure 20.25 Data Retransfer Operation in SCI Transmission Mode**



Downloaded from Arrow.com.

Note that the SSR.TEND flag is set in different timings depending on the GM bit setting in SMR. Figure 20.26 shows the TEND flag generation timing.

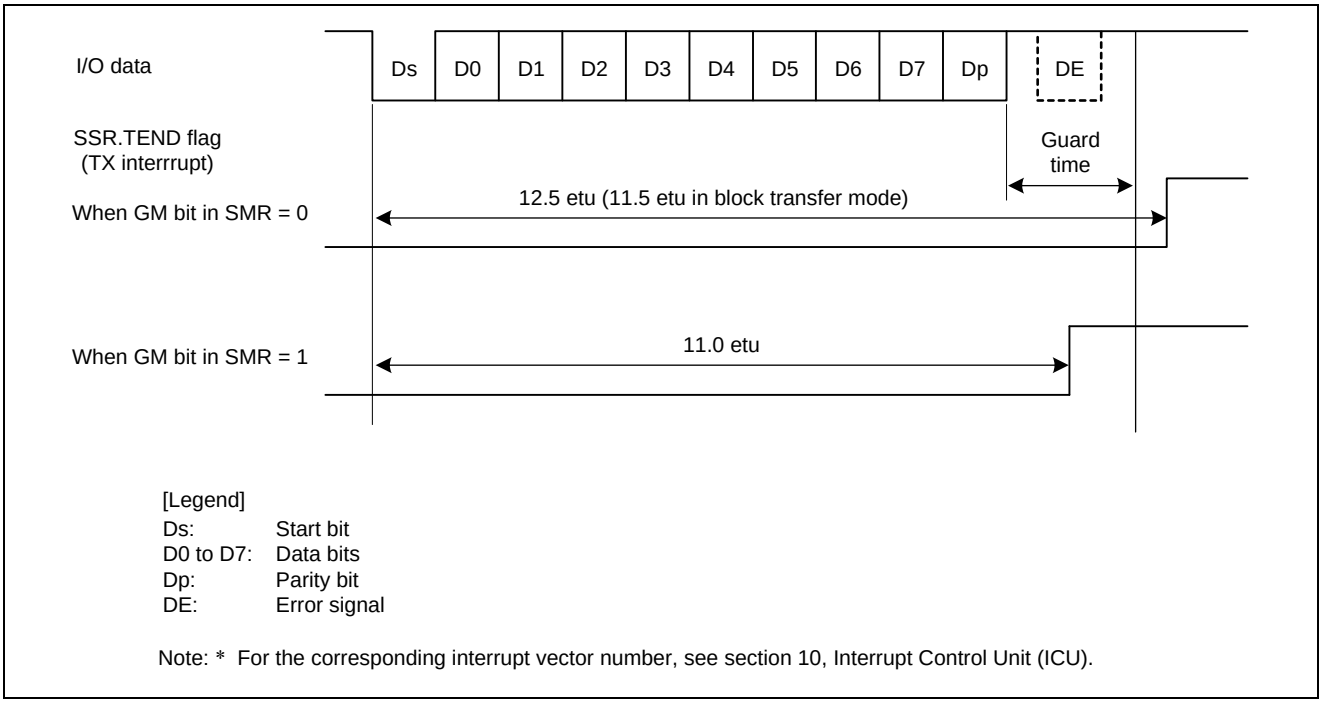



**Figure 20.26 SSR.TEND Flag Generation Timing during Transmission**



Downloaded from Arrow.com.

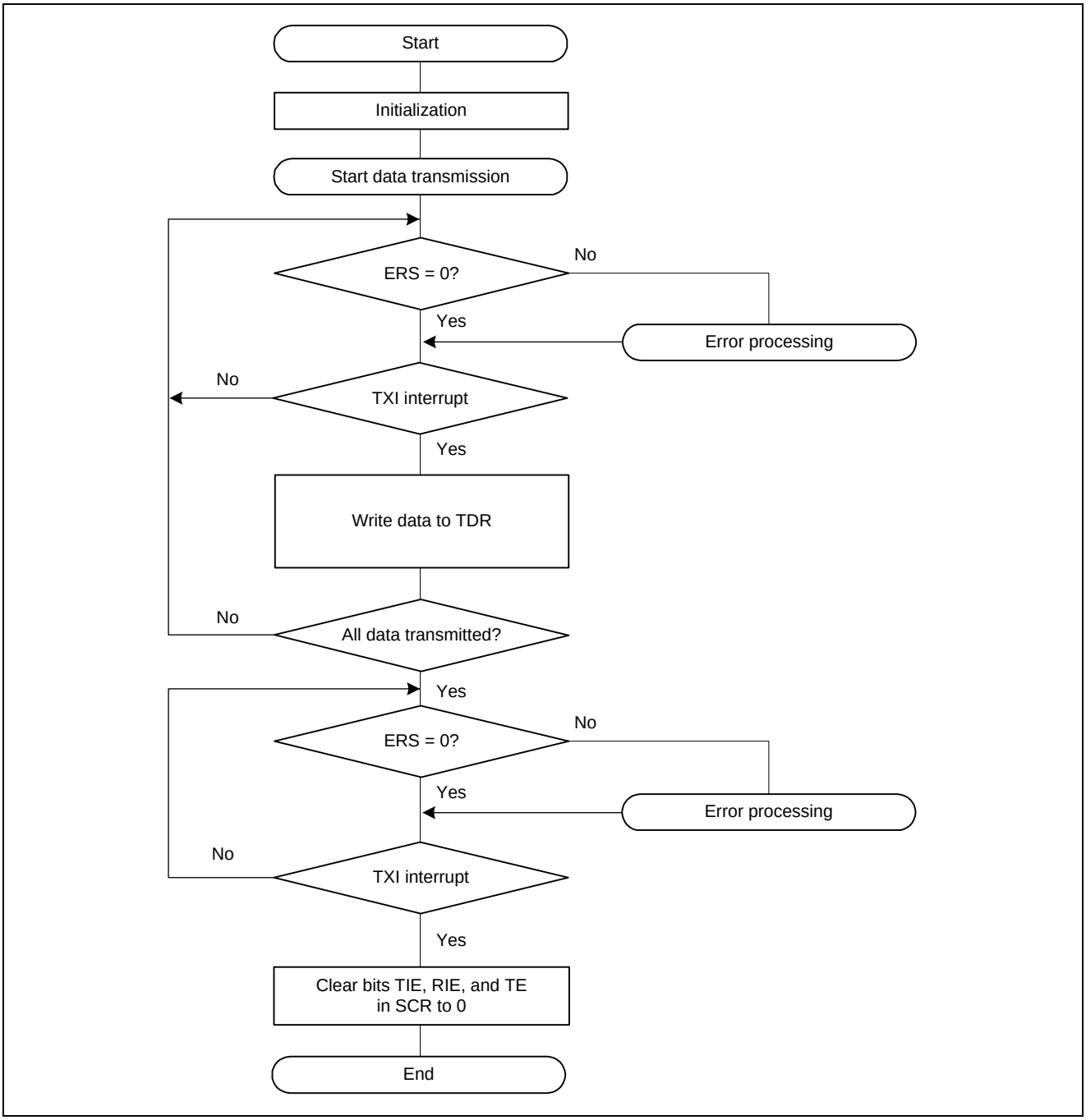



**Figure 20.27 Sample Serial Transmission Flowchart**



Downloaded from Arrow.com.

### 20.5.7 Serial Data Reception (Except in Block Transfer Mode)

Serial data reception in smart card interface mode is similar to that in serial communications interface mode. Figure 20.28 shows the data retransfer operation in reception mode.

1. If a parity error is detected in receive data, the PER flag in SSR is set to 1. When the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Clear the PER flag to 0 before the next parity bit is sampled.
2. For a frame in which a parity error is detected, no RXI interrupt is generated.
3. When no parity error is detected, the PER flag in SSR is not set to 1.
4. In this case, data is determined to have been received successfully. When the RIE bit in SCR is set to 1, an RXI interrupt request is generated.

Figure 20.29 shows a sample flowchart for serial data reception. All the processing steps are automatically performed using an RXI interrupt request to activate the DTC or DMAC.

In reception, setting the RIE bit to 1 allows an RXI interrupt request to be generated. The DTC or DMAC is activated by an RXI interrupt request if the RXI interrupt request is specified as a source of DTC or DMAC activation beforehand, allowing transfer of receive data.

If an error occurs during reception and either the ORER or PER flag in SSR is set to 1, a receive error interrupt (ERI) request is generated. Clear the error flag. If an error occurs, the DTC or DMAC is not activated and receive data is skipped. Therefore, the number of bytes of receive data specified in the DTC or DMAC is transferred.

Even if a parity error occurs and the PER flag is set to 1 during reception, receive data is transferred to RDR, thus allowing the data to be read.

Note: For operations in block transfer mode, see section 20.3, Operation in Asynchronous Mode.

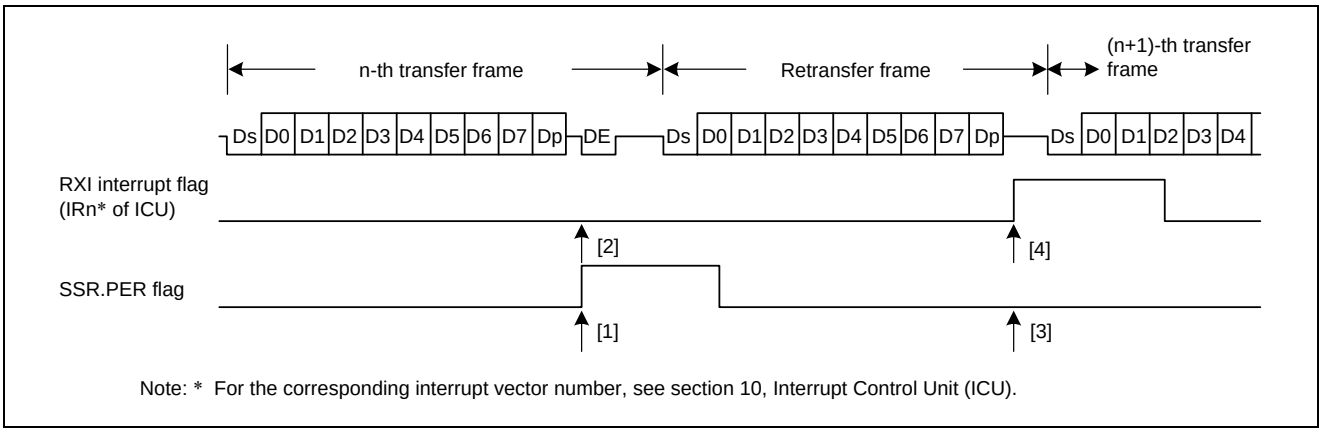



**Figure 20.28 Data Retransfer Operation in SCI Reception Mode**



Downloaded from Arrow.com.

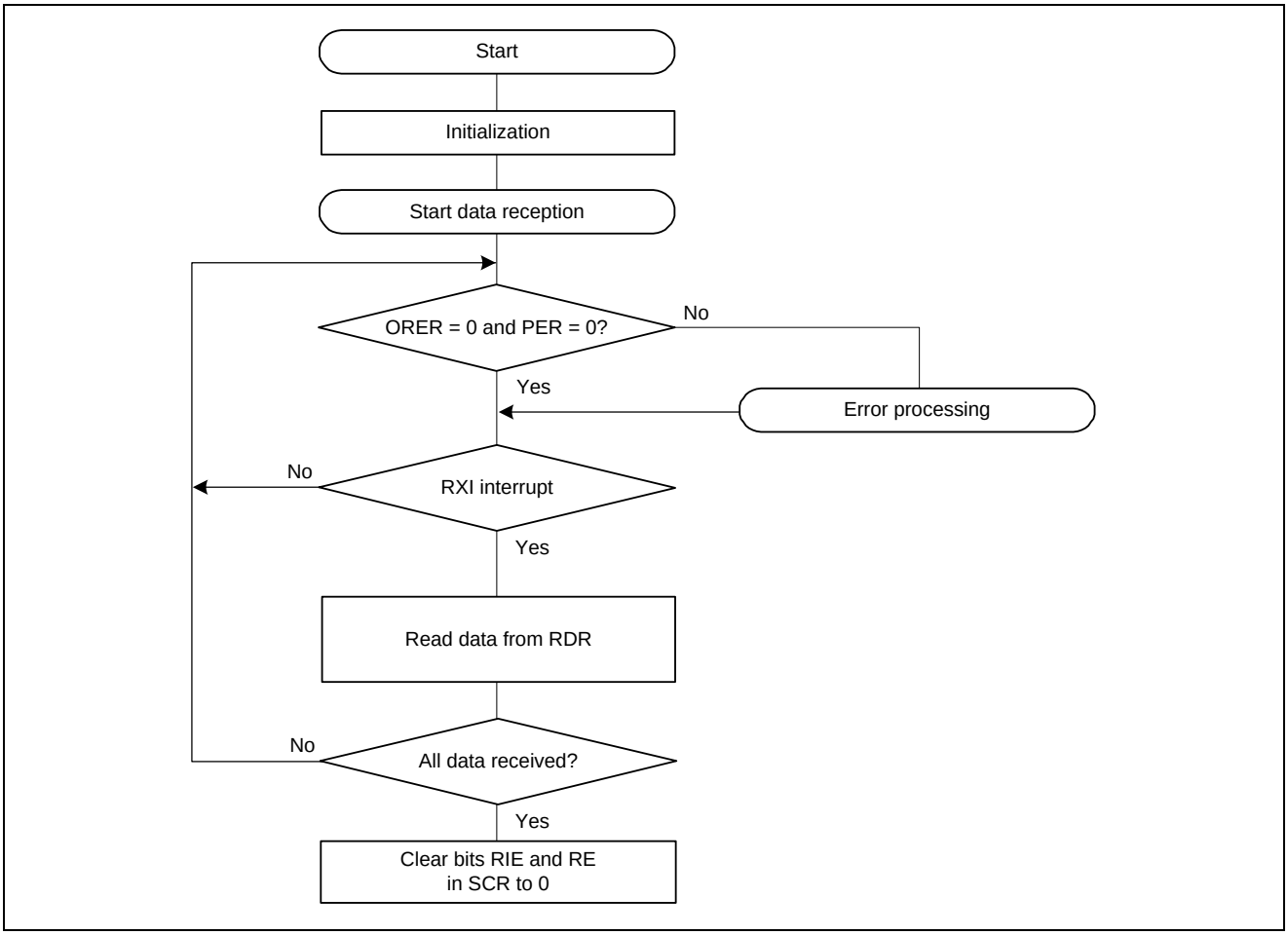



**Figure 20.29 Sample Serial Reception Flowchart**

## 20.5.8 Clock Output Control

Clock output can be fixed using the CKE[1:0] bits in SCR when the GM bit in SMR is set to 1. Specifically, the minimum width of a clock pulse can be specified.

Figure 20.30 shows an example of clock output stop timing when the CKE0 bit is controlled with GM = 1 and CKE1 = 0.

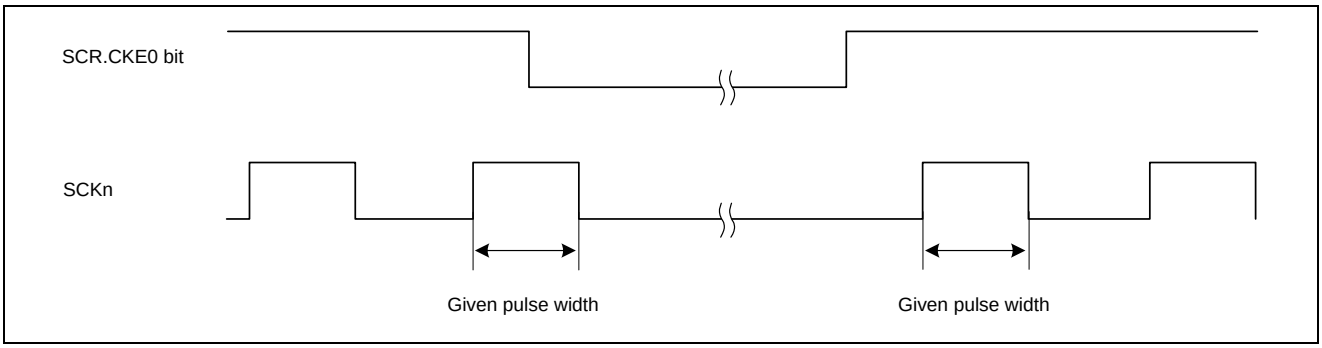



**Figure 20.30 Clock Output Stop Timing**

At power-on and transitions to/from software standby mode, use the following procedure to secure the appropriate clock duty cycle.



Downloaded from Arrow.com.

### (1) At Power-On

To secure the appropriate clock duty cycle simultaneously with power-on, use the following procedure.

1. Initially, port input is enabled in the high-impedance state. To fix the potential level, use a pull-up or pull-down resistor.
2. Fix the SCKn pin to the specified output using the CKE1 bit in SCR.
3. Set SMR and SCMR to enable smart card interface mode. Set the CKE0 bit in SCR to 1 to start clock output.

### (2) At Mode Switching

#### (a) At transition from smart card interface mode to software standby mode

1. Set the data register (DR of Pm) and data direction register (DDR of Pm) corresponding to the SCKn pin to the values for the output fixed state in software standby mode.
2. Write 0 to the TE and RE bits in SCR to stop transmission/reception.
   Simultaneously, set the CKE1 bit in SCR to the value for the output fixed state in software standby mode.
3. Write 0 to the CKE0 bit in SCR to stop the clock.
4. Wait for one cycle of the serial clock. In the mean time, the clock output stops and is fixed low level after the specified high-level period is output.
5. Make a transition to software standby mode.

#### (b) At transition from software standby mode to smart card interface mode

1. Cancel software standby mode.
2. Set the CKE0 bit in SCR to 1. The clock output with the specified frequency is then re-started.

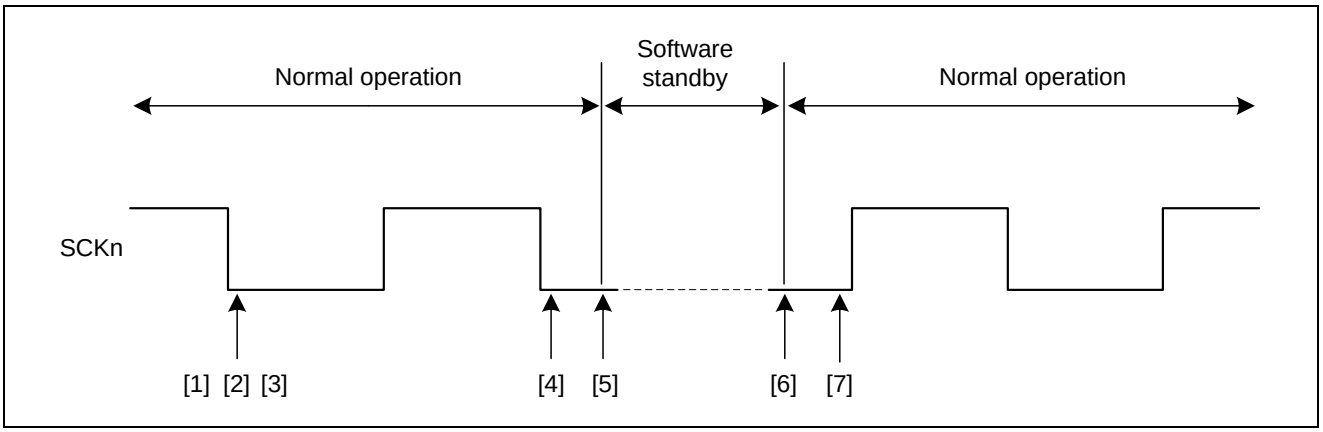



**Figure 20.31 Clock Stop and Restart Procedure**



Downloaded from Arrow.com.

## 20.6 Interrupt Sources

### 20.6.1 Interrupts in Serial Communications Interface Mode

Table 20.15 lists interrupt sources in normal serial communications interface mode. A different interrupt vector is assigned to each interrupt source, and individual interrupt sources can be enabled or disabled with the enable bits in SCR.

Transfer of data from the transmit data register (TDR) to the TSR while the SCR.TIE bit is 1 leads to the generation of a TXI interrupt request. Moreover, setting the TIE bit in SCR to 1 and then setting the TE bit to 1, or setting the TIE and TE bits in SCR to 1 simultaneously using one instruction, leads to the generation of a TXI interrupt request. A TXI interrupt request can activate the DTC or DMAC for data transfer.

Setting of received data in the RDR while the SCR.RIE bit is 1 leads to the generation of an RXI interrupt request. An RXI interrupt can activate the DTC or DMAC for data transfer.

Setting the ORER, FER, or PER flag in the SSR to 1 while the SCR.RIE bit is 1 leads to generation of an ERI interrupt request.

If the TDR has not been updated by the time of transmission of the tail-end bit of data being transmitted, the SSR.TEND flag is set to 1 and, if the value of the SCR.TEIE bit is 1, a TEI interrupt request is generated. Writing of data to the TDR during TXI interrupt processing leads to clearing of the SSR.TEND flag and clearing of the TEI interrupt at its source.

The TXI interrupt request is generated by setting the TIE bit in SCR to 1 and then setting the TE bit to 1 or by setting the TIE and TE bits in SCR to 1 simultaneously using one instruction. The TXI interrupt is not generated if the TE bit is set while the TIE bit is 0 or the TIE bit is set to 1 after the TE bit is set to 1. Therefore, to disable the TXI interrupt, perform the transmit end interrupt processing, and then start data transfer again, for example, in the final data transmission, the TXI interrupt should be enabled/disabled using the corresponding ICU.IERm.IENj bit.

**Table 20.15 SCI Interrupt Sources**

| Name | Interrupt Source | Interrupt Flag | DTC Activation | DMAC Activation | Priority |
|---|---|---|---|---|---|
| ERI | Receive error | ORER, FER, or PER | Not possible | Not possible | High |
| RXI | Receive data full | — | Possible | Possible | ↑ |
| TXI | Transmit data empty | — | Possible | Possible | |
| TEI | Transmit end | TEND | Not possible | Not possible | Low |



Downloaded from Arrow.com.

### 20.6.2 Interrupts in Smart Card Interface Mode

Table 20.16 lists interrupt sources in smart card interface mode. A transmit end interrupt (TEI) request cannot be used in this mode.

**Table 20.16 SCI Interrupt Sources**

| Name | Interrupt Source | Interrupt Flag | DTC Activation | DMAC Activation | Priority |
|---|---|---|---|---|---|
| ERI | Receive error or error signal detection | ORER, PER, or ERS | Not possible | Not possible | High |
| RXI | Receive data full | — | Possible | Possible | ↑ |
| TXI | Transmit data empty | TEND | Possible | Possible | Low |

Data transmission/reception using the DTC or DMAC is also possible in smart card interface mode, similar to in the normal SCI mode. In transmission, when the SSR.TEND flag is set to 1, a TXI interrupt request is generated. This TXI interrupt request activates the DTC or DMAC allowing transfer of transmit data if the TXI request is specified beforehand as a source of DTC or DMAC activation. The TEND flag is automatically cleared to 0 when the DTC or DMAC transfers the data.

If an error occurs, the SCI automatically re-transmits the same data. During the retransmission, the TEND flag is kept to 0 and the DTC or DMAC is not activated. Therefore, the SCI and DTC or DMAC automatically transmit the specified number of bytes, including retransmission in the case of error occurrence. However, the ERS flag in SSR is not automatically cleared to 0 at error occurrence. Therefore, the ERS flag must be cleared by previously setting the RIE bit in SCR to 1 to enable an ERI interrupt request to be generated at error occurrence.

When transmitting/receiving data using the DTC or DMAC, be sure to make settings to enable the DTC or DMAC before making SCI settings. For DTC or DMAC settings, see section 12, DMA Controller (DMAC) and section 13, Data Transfer Controller (DTC).

In reception, an RXI interrupt request is generated when receive data is set to RDR. This RXI interrupt request activates the DTC or DMAC allowing transfer of receive data if the RXI request is specified beforehand as a source of DTC or DMAC activation. If an error occurs, the error flag is set. Therefore, the DTC or DMAC is not activated and an ERI interrupt request is issued to the CPU instead; the error flag must be cleared.



Downloaded from Arrow.com.

## 20.7 Usage Notes

### 20.7.1 Setting the Module Stop Function

Operation of the SCI can be disabled or enabled using the module stop control register B (MSTPCRB). The initial setting is for operation of the SCI to be halted. Register access is enabled by clearing the module stop state. For details, see section 8, Low Power Consumption.

### 20.7.2 Break Detection and Processing

When a framing error is detected, a break can be detected by reading the RxDn pin value directly. In a break, the input from the RxDn pin becomes all 0s, and so the FER flag in SSR is set to 1 (framing error), and the PER flag in SSR may also be set to 1 (parity error). The SCI continues the receive operation even after a break is received. Therefore, note that even if the FER flag is cleared to 0 (no framing error), it will be set to 1 again.

### 20.7.3 Mark State and Break Detection

When the TE bit in SCR is 0 (serial transmission disabled), the TxDn pin is used as an I/O port. This can be used to set the TxDn pin to mark state or send a break during serial data transmission.

To maintain the communications line in mark state (the state of 1) until the TE bit is set to 1 (to enable serial transmission), set both Bj bit in DR of Pm and Bj bit in DDR of Pm to 1. Since the TE bit is cleared to 0 at this time, the TxDn pin becomes an I/O port, and 1 is output from the TxDn pin.

To send a break during serial data transmission, first set Bj bit in DDR of Pm = 1 and Bj bit in DR of Pm = 0, and then clear the TE bit to 0. When the TE bit is cleared to 0, the transmitter is initialized regardless of the current transmission state, the TxDn pin becomes an I/O port, and a 0 is output from the TxDn pin.

### 20.7.4 Receive Error Flags and Transmit Operations (Clock Synchronous Mode Only)

Transmission cannot be started when a receive error flag (ORER, FER, or RER) in SSR is set to 1, even if data is written to TDR. Be sure to clear the receive error flags to 0 before starting transmission. Note also that the receive error flags cannot be cleared to 0 even if the RE bit in SCR is cleared to 0 (serial reception disabled).

### 20.7.5 Writing Data to TDR

Data can always be written to TDR. However, if new data is written to TDR when transmit data is remaining in TDR, the previous data in TDR is lost because it has not been transferred to TSR yet. Be sure to write transmit data to TDR in the TXI interrupt request processing routine.



Downloaded from Arrow.com.

### 20.7.6 Restrictions on Clock Synchronous Transmission

When the external clock source is used as a synchronization clock, update TDR by the DMAC or DTC and wait for at least five clock cycles before allowing the transmit clock to be input. If the transmit clock is input within four clock cycles after TDR is updated, the SCI may malfunction.

### 20.7.7 Restrictions on Using DTC or DMAC

When using the DMAC or DTC to read RDR, be sure to set the receive end interrupt (RXI) as the activation source of the relevant SCI.

### 20.7.8 SCI Operations during Power-Down State

#### (1) Transmission

Before specifying the module stop state or making a transition to software standby mode, stop the transmit operations (TIE = TE = TEIE = 0 in SCR). TSR, TDR, and SSR are reset. The states of the output pins in the module stop state or in software standby mode depend on the port settings, and the output pins are held high after cancellation. If the transition is made during data transmission, the data being transmitted will be undefined.

To transmit data in the same transmission mode after cancellation of the power-down state, set the TE bit to 1, read SSR, and write data to TDR sequentially to start data transmission. To transmit data with a different transmission mode, initialize the SCI first.

Figure 20.32 shows a sample flowchart for transition to software standby mode during transmission. Figures 20.33 and 20.34 show the port pin states during transition to software standby mode.

Before specifying the module stop state or making a transition to software standby mode from the transmission mode using DTC transfer, stop the transmit operations (TE = 0). To start transmission after cancellation using the DTC, set the TE bit to 1. The TXI interrupt flag is set to 1 and transmission starts using the DTC.

#### (2) Reception

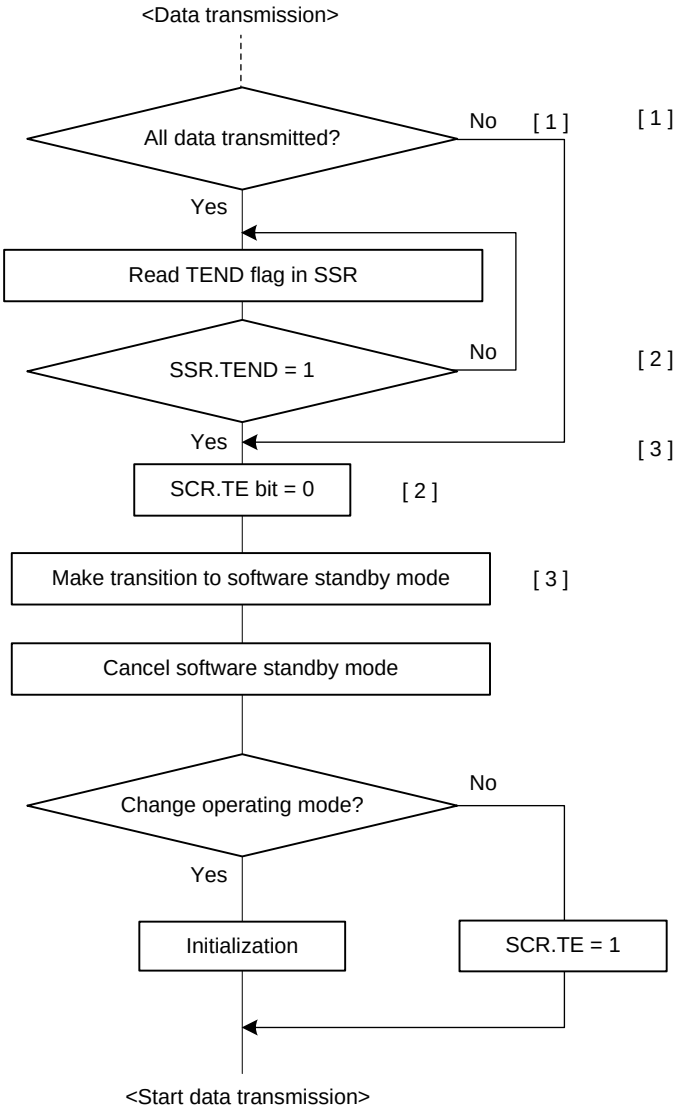
Before specifying the module stop state or making a transition to software standby mode, stop the receive operations (RE = 0 in SCR). RSR, RDR, and SSR are reset. If transition is made during data reception, the data being received will be invalid.

To receive data in the same reception mode after cancellation of the power-down state, set the RE bit to 1, and then start reception. To receive data in a different reception mode, initialize the SCI first.

Figure 20.35 shows a sample flowchart for mode transition during reception.



Downloaded from Arrow.com.




[ 1 ] Data being transmitted is lost halfway. Data can be normally transmitted from the CPU by setting the TE bit in SCR to 1, reading SSR, and writing data to TDR after canceling software standby mode. However, if the DTC has been activated, the data remaining in the DTC will be transmitted when both the TE and TIE bits in SCR are set to 1.

[ 2 ] Clear the TIE and TEIE bits in SCR to 0 when they are 1.

[ 3 ] Setting for the module stop state is included.

**Figure 20.32 Example of Flowchart for Transition to Software Standby Mode during Transmission**

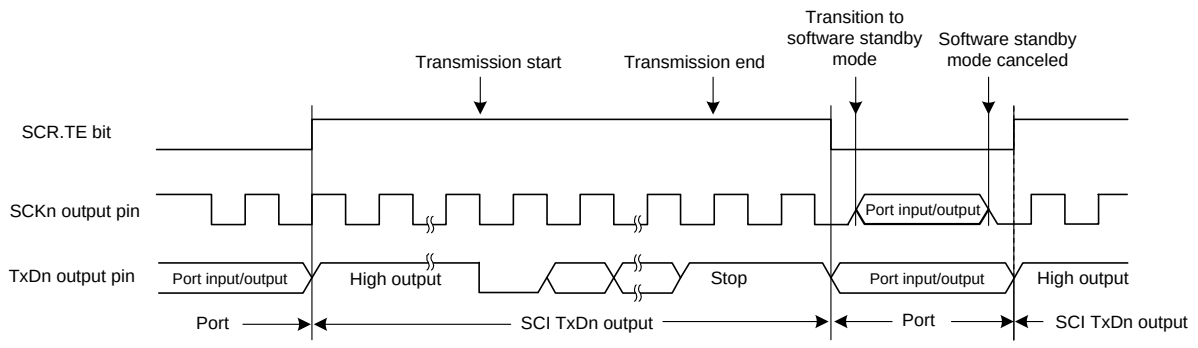



**Figure 20.33 Port Pin States during Transition to Software Standby Mode (Internal Clock, Asynchronous Transmission)**



Downloaded from Arrow.com.

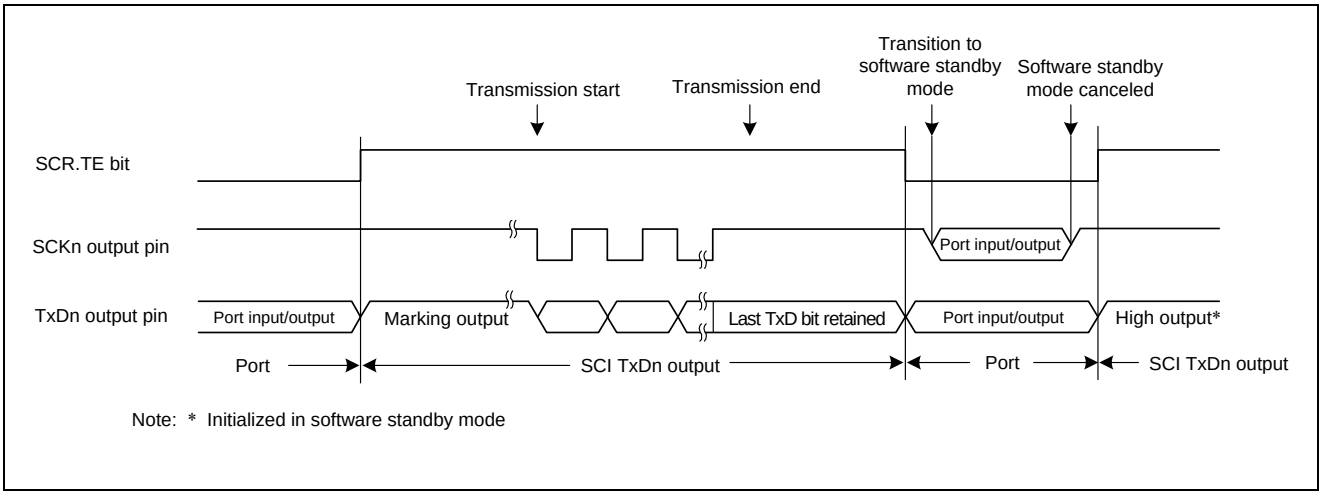



**Figure 20.34 Port Pin States during Transition to Software Standby Mode (Internal Clock, Clock Synchronous Transmission)**

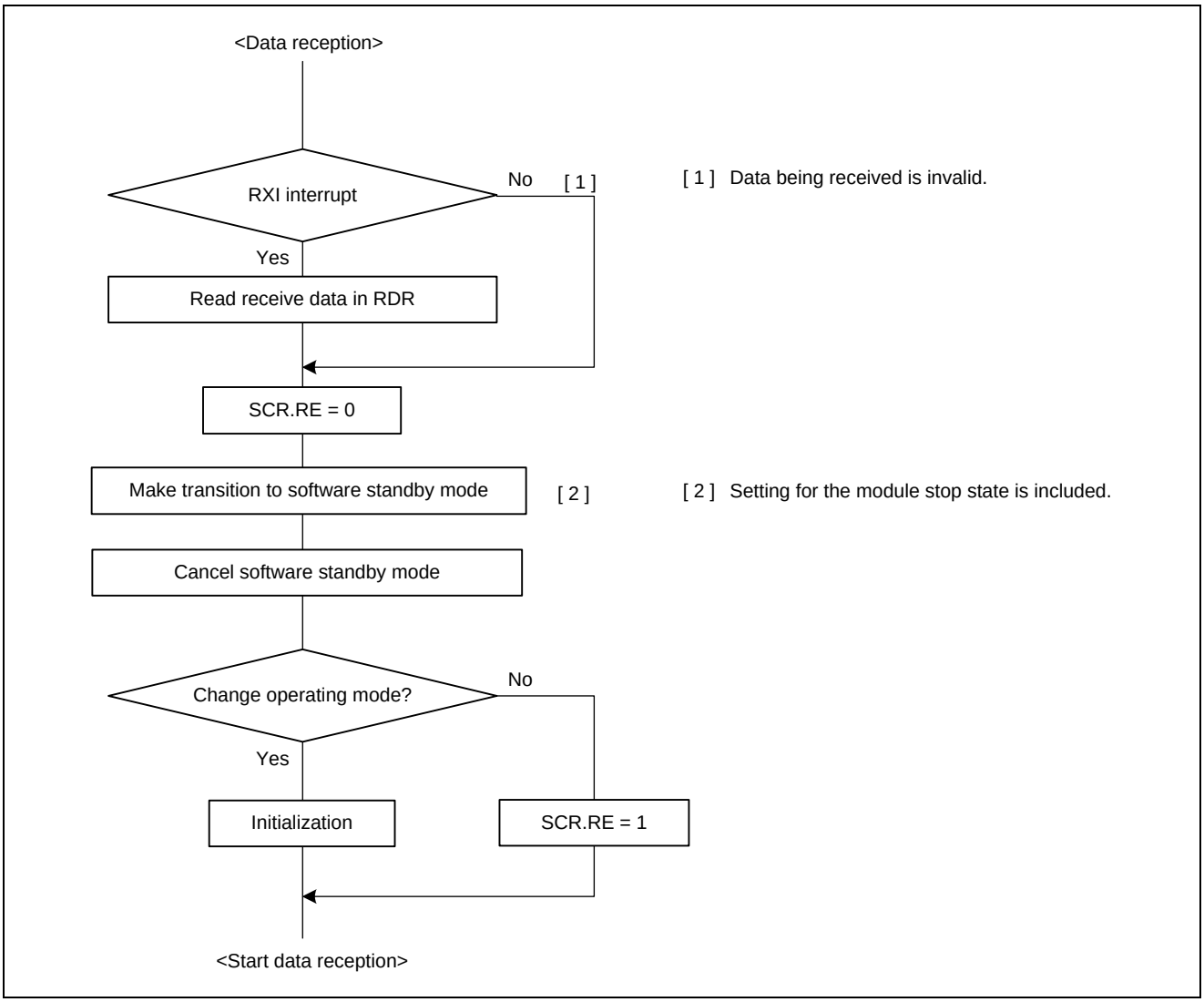



**Figure 20.35 Example of Flowchart for Transition to Software Standby Mode during Reception**



Downloaded from Arrow.com.

### 20.7.9 External Clock Input in Clock Synchronous Mode

In clock synchronous mode, the external clock SCKn must be input as follows:

High-pulse period, low-pulse period = 2 clock cycles or more, period = 6 clock cycles or more



Downloaded from Arrow.com.

# 21. CRC Calculator (CRC)

The CRC (Cyclic Redundancy Check) calculator generates CRC codes of data blocks.

## 21.1 Overview

Table 21.1 lists the specifications of the CRC calculator, and figure 21.1 shows a block diagram of the CRC calculator.

**Table 21.1 Specifications of CRC**

| Item | Description |
|---|---|
| Data for CRC calculation* | CRC code generated for any desired data in 8n-bit units (where n is a whole number) |
| Data block size | 8 bits |
| CRC processor unit | Operation executed on eight bits in parallel |
| CRC generating polynomial | One of three generating polynomials selectable<br>• 8-bit CRC<br>$X^8 + X^2 + X + 1$<br>• 16-bit CRC<br>$X^{16} + X^{15} + X^2 + 1$<br>$X^{16} + X^{12} + X^5 + 1$ |
| CRC calculation switching | CRC code generation for LSB-first or MSB-first communication selectable |
| Power-down function | Module stop state can be set |

Note: * The circuit does not have functionality to divide data for calculation into a data-block size. Write data in 8-bit units.

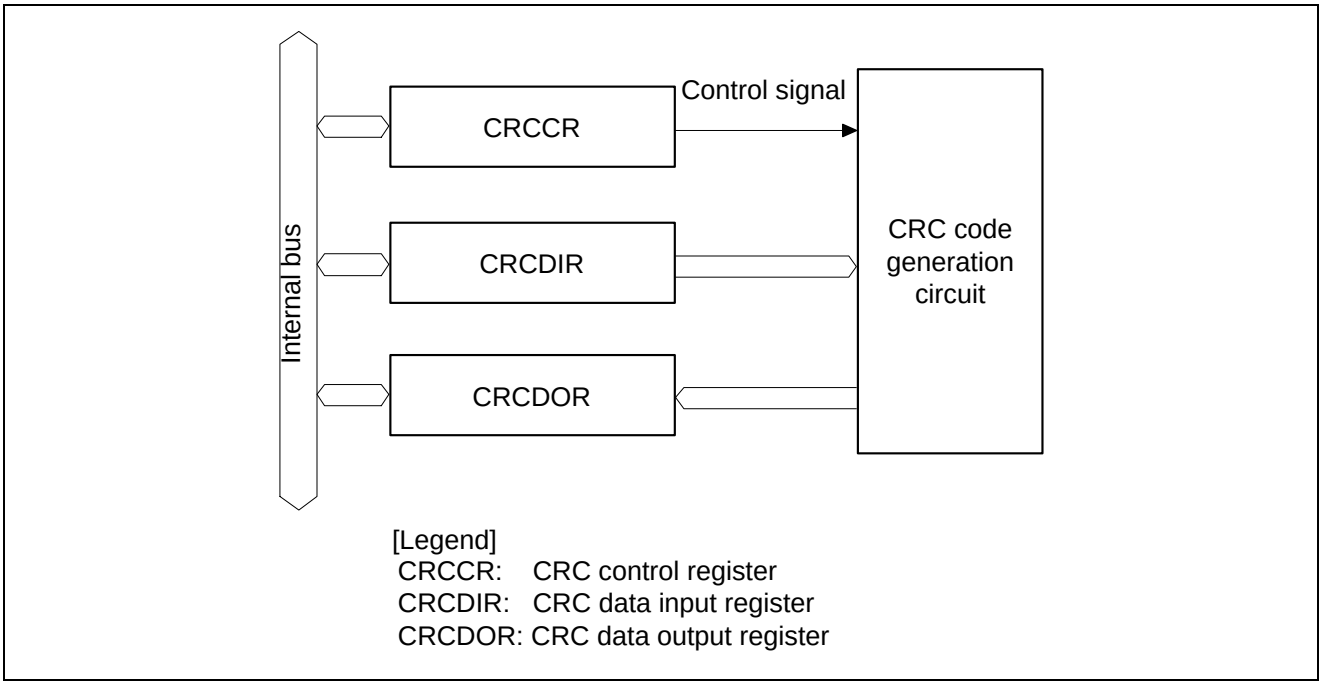



**Figure 21.1 Block Diagram of CRC Calculator**



Downloaded from Arrow.com.

## 21.2 Register Descriptions

Table 21.2 lists the registers of the CRC calculator.

**Table 21.2 Registers of CRC Calculator**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| CRC control register | CRCCR | 00h | 0008 8280h | 8 |
| CRC data input register | CRCDIR | 00h | 0008 8281h | 8 |
| CRC data output register | CRCDOR | 0000h | 0008 8282h | 16 |

### 21.2.1 CRC Control Register (CRCCR)

Address: 0008 8280h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DORCLR | — | — | — | — | LMS | GPS[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | GPS[1:0] | CRC Generating Polynomial Switching | b0 b1<br>0 0: No calculation is executed.*<br>0 1: $X^8 + X^2 + X + 1$<br>1 0: $X^{16} + X^{15} + X^2 + 1$<br>1 1: $X^{16} + X^{12} + X^5 + 1$ | R/W |
| b2 | LMS | CRC Calculation Switching | 0: Performs CRC operation for LSB-first communication. The lower-order byte (bits 7 to 0) is the first to be transmitted when the value of the CRCDOR (CRC code) are divided into bytes.<br>1: Performs CRC operation for MSB-first communication. The higher-order byte (bits 15 to 8) is first to be transmitted when the value of the CRCDOR (CRC code) are divided into bytes. | R/W |
| b6 to b3 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DORCLR | CRCDOR Register Clear | 0: No effect on the operation<br>1: Clear the CRCDOR register<br>This bit is always read as 0. | R/W |

CRCCR initializes the CRC calculator, switches the operation mode, and selects the generating polynomial.

#### GPS[1:0] Bits (CRC Generating Polynomial Switching)

These bits select the CRC code generating polynomial.

#### LMS Bit (CRC Calculation Switching)

Selects LSB-first or MSB-first communication for the CRC code generation.



Downloaded from Arrow.com.

**DORCLR Bit (CRCDOR Register Clear)**

Write 1 to this bit so that the CRCDOR register is cleared to 0000h.

This bit is always read as 0.

## 21.2.2 CRC Data Input Register (CRCDIR)

Address: 0008 8281h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CRCDIR is an 8-bit readable/writable register, to which the bytes to be CRC-operated are written.

## 21.2.3 CRC Data Output Register (CRCDOR)

Address: 0008 8282h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CRCDOR is a 16-bit readable/writable register that contains the result of CRC calculation.

In general, the value will be 0 if there is no CRC error when the calculated CRC code matches the CRC code that continues on, for verification, from the transferred data.

When an 8-bit CRC ($X^8 + X^2 + X + 1$ polynomial) is in use, the valid CRC code is obtained in the lower-order byte (b7 to b0). The higher-order byte (b15 to b8) is not be updated.



Downloaded from Arrow.com.

## 21.3 Operation

The CRC calculator generates CRC codes for use in LSB-first or MSB-first transfer.

The following figures show examples in which the CRCCR.GPS[1:0] bits are set to 11b so the CRC code is calculated by using a 16-bit CRC (with the polynomial $X^{16} + X^{12} + X^5 + 1$), and the CRC code is calculated for the value "F0h".

When an 8-bit CRC (with the polynomial $X^8 + X^2 + X + 1$) is in use, the valid bits of the CRC code are obtained in the lower-order byte of CRCDOR.

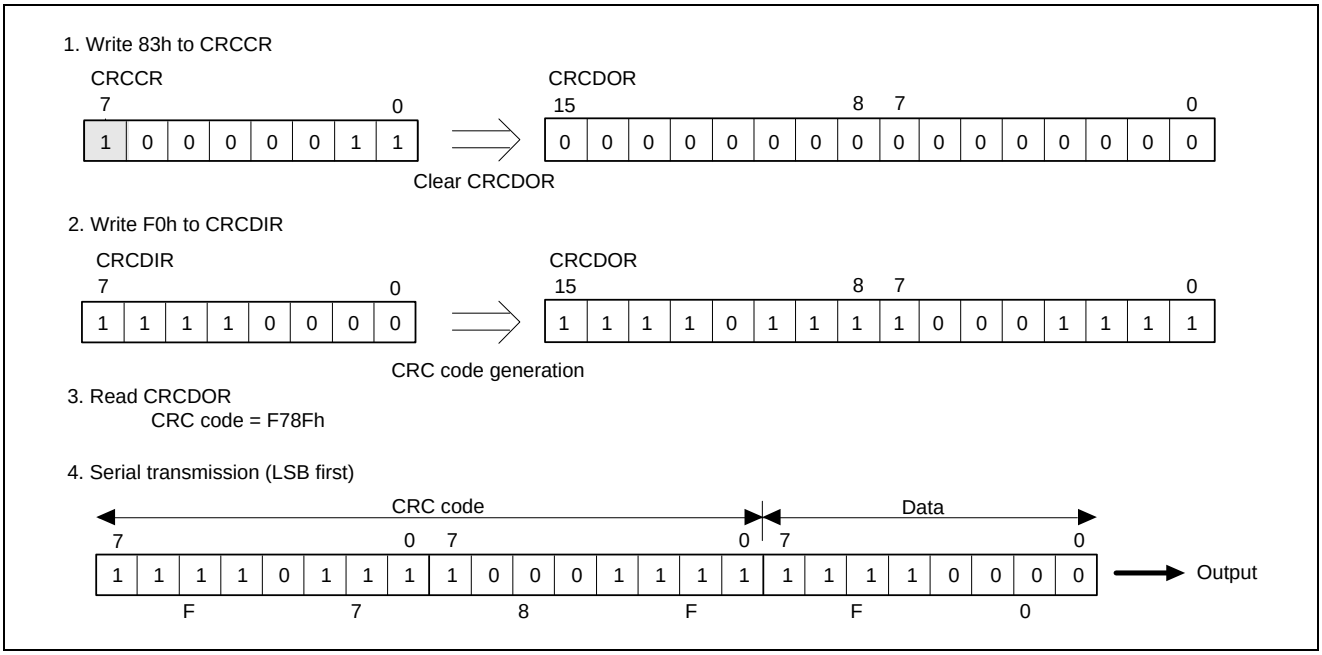



**Figure 21.2 LSB-First Data Transmission**

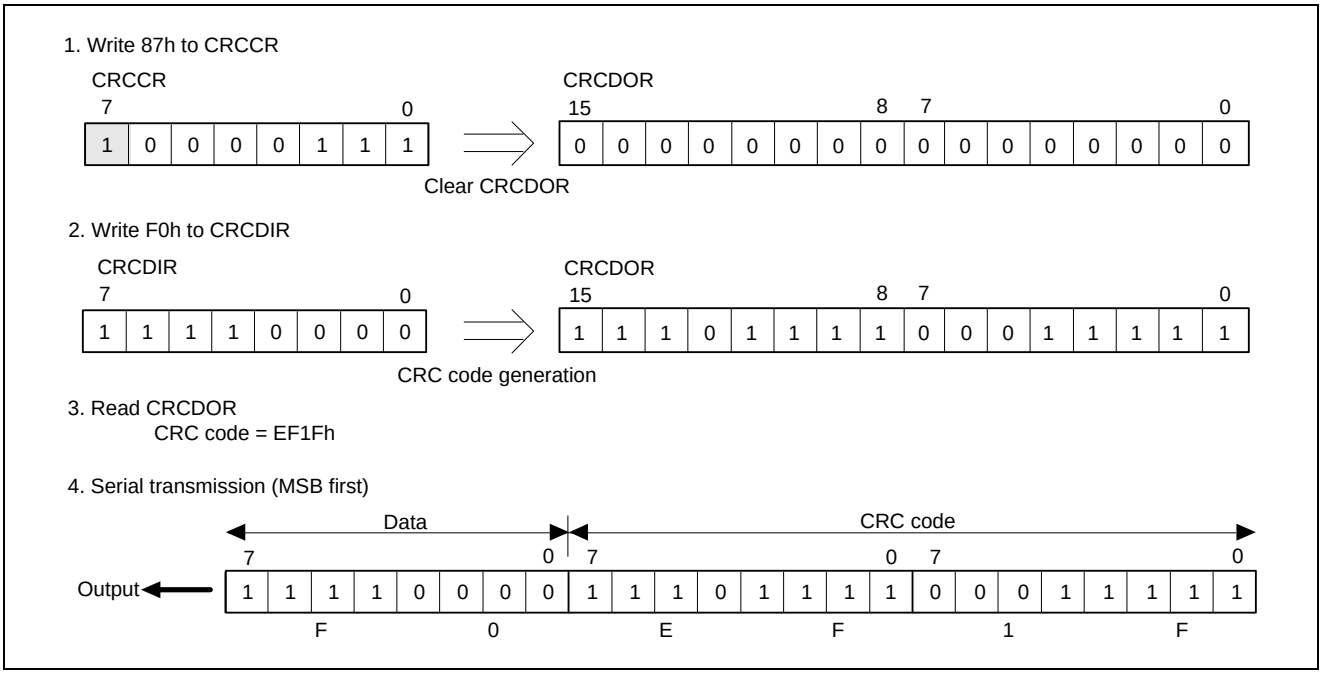



**Figure 21.3 MSB-First Data Transmission**



Downloaded from Arrow.com.

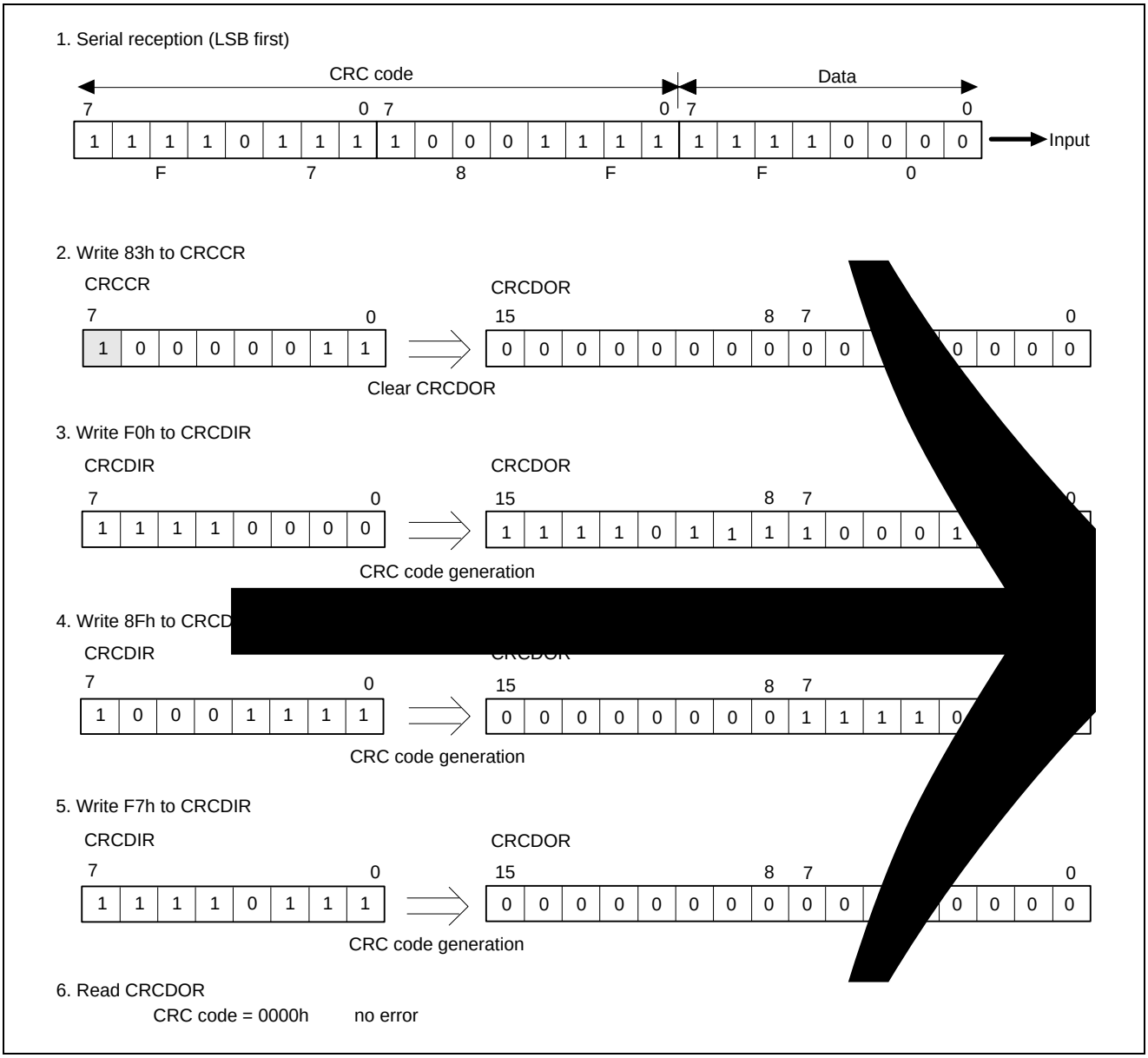



**Figure 21.4 LSB-First Data Reception**



Downloaded from Arrow.com.

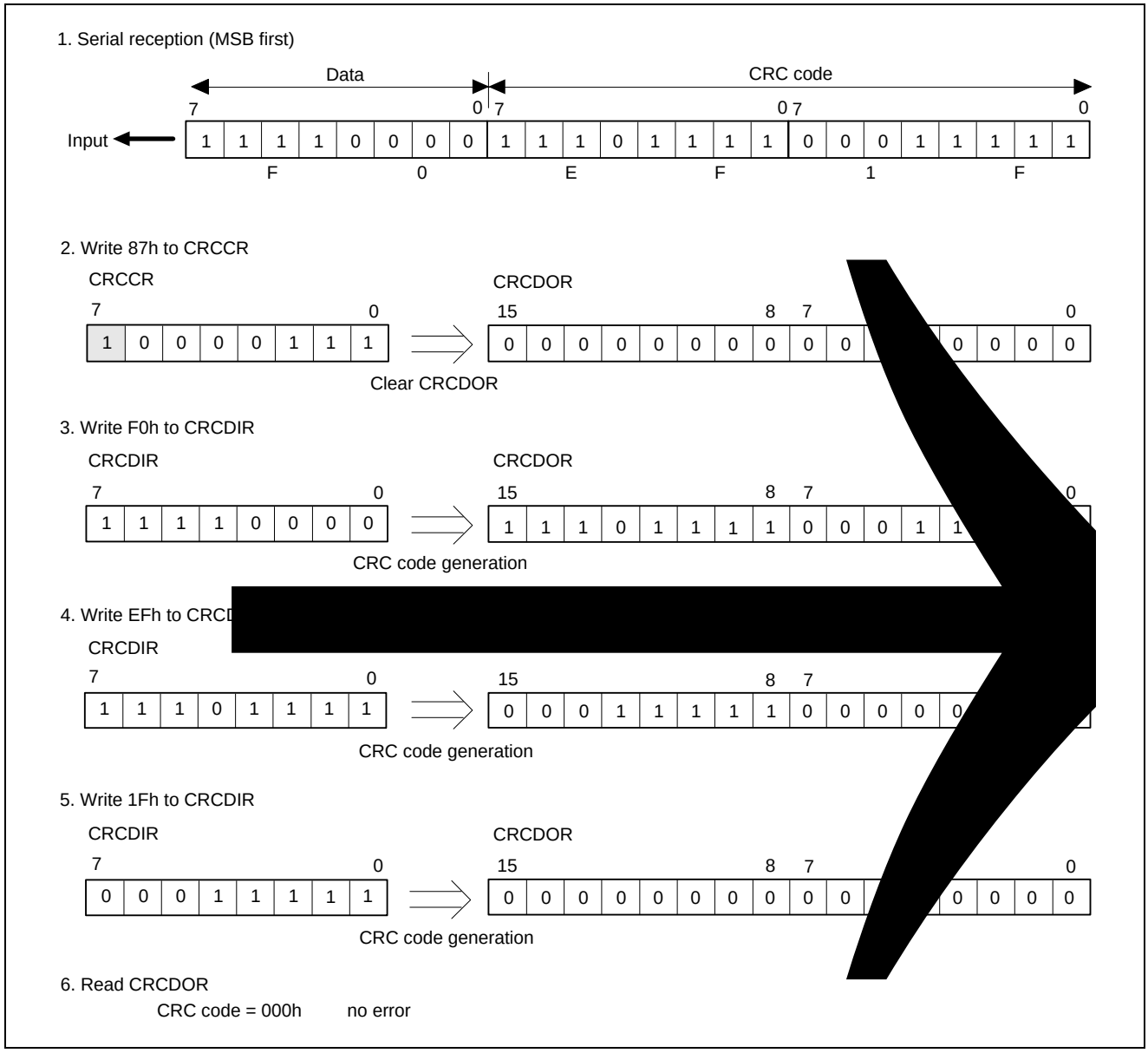



**Figure 21.5 MSB-First Data Reception**



Downloaded from Arrow.com.

## 21.4 Usage Notes

### 21.4.1 Module Stop Function Setting

Operation of the CRC calculator can be disabled or enabled using the module stop control register B (MSTPCRB). The initial setting is for operation of the CRC calculator to be halted. Register access is enabled by clearing the module stop state. For details, see section 8, Low Power Consumption.

## 21.5 Note on Transmission

Note that the sequence of transmission for the CRC code differs according to whether transmission is LSB-first or MSB-first.

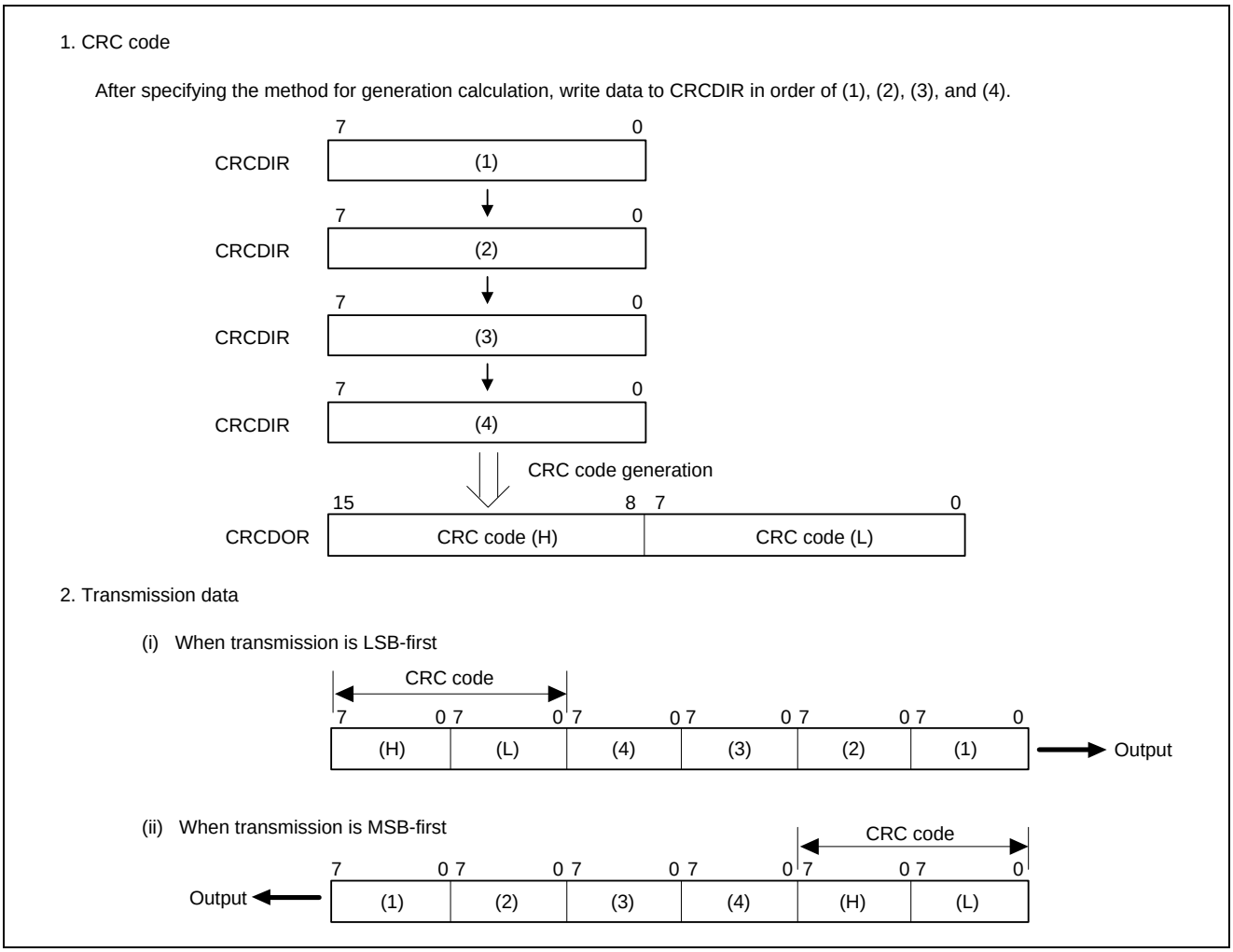



**Figure 21.6 LSB-First and MSB-First Data Transmission**



Downloaded from Arrow.com.

# 22. I²C Bus Interface (RIIC)

The RX610 Group has two I²C bus interfaces (RIIC modules).

The RIIC module conforms with and provides a subset of the NXP I²C bus (inter-IC bus) interface functions.

## 22.1 Overview

Table 22.1 lists the specifications of the RIIC, figure 22.1 shows a block diagram of the RIIC, and figure 22.2 shows an example of I/O pin connections to external circuits (I²C bus configuration example). Table 22.2 shows the input/output pins of the RIIC.

**Table 22.1 RIIC Specifications**

| Item | Specifications |
|---|---|
| Communications format | • I²C bus format or SMBus format<br>• Master mode or slave mode selectable<br>• Automatic securing of the various set-up times, hold times, and bus-free times for the transfer rate |
| Transfer rate | Up to 1M bps |
| SCL clock | For master operation, the duty cycle of the SCL clock is selectable in the range from 4% to 96%. |
| Issuing and detecting conditions | Start, restart, and stop conditions are automatically generated.<br>Start conditions (including restart conditions) and stop conditions are detectable. |
| Slave address | • Up to three slave-address settings can be made.<br>• Seven- and ten-bit address formats are supported (along with the use of both at once).<br>• General call addresses, device ID addresses, and SMBus host addresses are detectable. |
| Acknowledgement | • For transmission, the acknowledge bit is automatically loaded.<br>Transfer of the next data for transmission can be automatically suspended on detection of a not-acknowledge bit.<br>• For reception, the acknowledge bit is automatically transmitted.<br>If a wait between the eighth and ninth clock cycles has been selected, software control of the value in the acknowledge field in response to the received value is possible (i.e. the return of ACK or NACK is selectable). |
| Wait function | • In reception, the following periods of waiting can be obtained by holding the clock signal (SCL) at the low level:<br>waiting between the eighth and ninth clock cycles; and<br>waiting between the ninth clock cycle and the first clock cycle of the next transfer (WAIT function) |
| SDA output delay function | Timing of the output of transmitted data, including the acknowledge bit, can be delayed. |
| Arbitration | • For multi-master operation<br>Operation to synchronize the SCL (clock) signal in cases of conflict with the SCL signal from another master is possible.<br>When issuing the start condition would create conflict on the bus, loss of arbitration is detected by testing for non-matching between the internal signal for the SDA line and the level on the SDA line.<br>In master operation, loss of arbitration is detected by testing for non-matching between the signal on the SDA line and the internal signal for the SDA line.<br>• Loss of arbitration due to detection of the start condition while the bus is busy is detectable (to prevent the issuing of double start conditions).<br>• Loss of arbitration in transfer of a not-acknowledge bit due to the internal signal for the SDA line and the level on the SDA line not matching is detectable.<br>• Loss of arbitration due to non-matching of internal and line levels for data is detectable in slave transmission. |



Downloaded from Arrow.com.

| Item | Specifications |
|---|---|
| Timeout detection function | • The internal timeout detection function is capable of detecting long-interval stoppages of the SCL (clock signal). |
| Noise removal | • The interface incorporates digital noise filters for both the SCL and SDA signals, and the width for noise cancellation by the filters is adjustable. |
| Interrupt sources | • Four sources:<br>Error in transfer or occurrence of events (detection of AL, NACK, timeout, a start condition including a restart condition, or a stop condition)<br>Receive-data-full (including matching with a slave address)<br>Transmit-data-empty (including matching with a slave address)<br>Transmission complete |



Downloaded from Arrow.com.

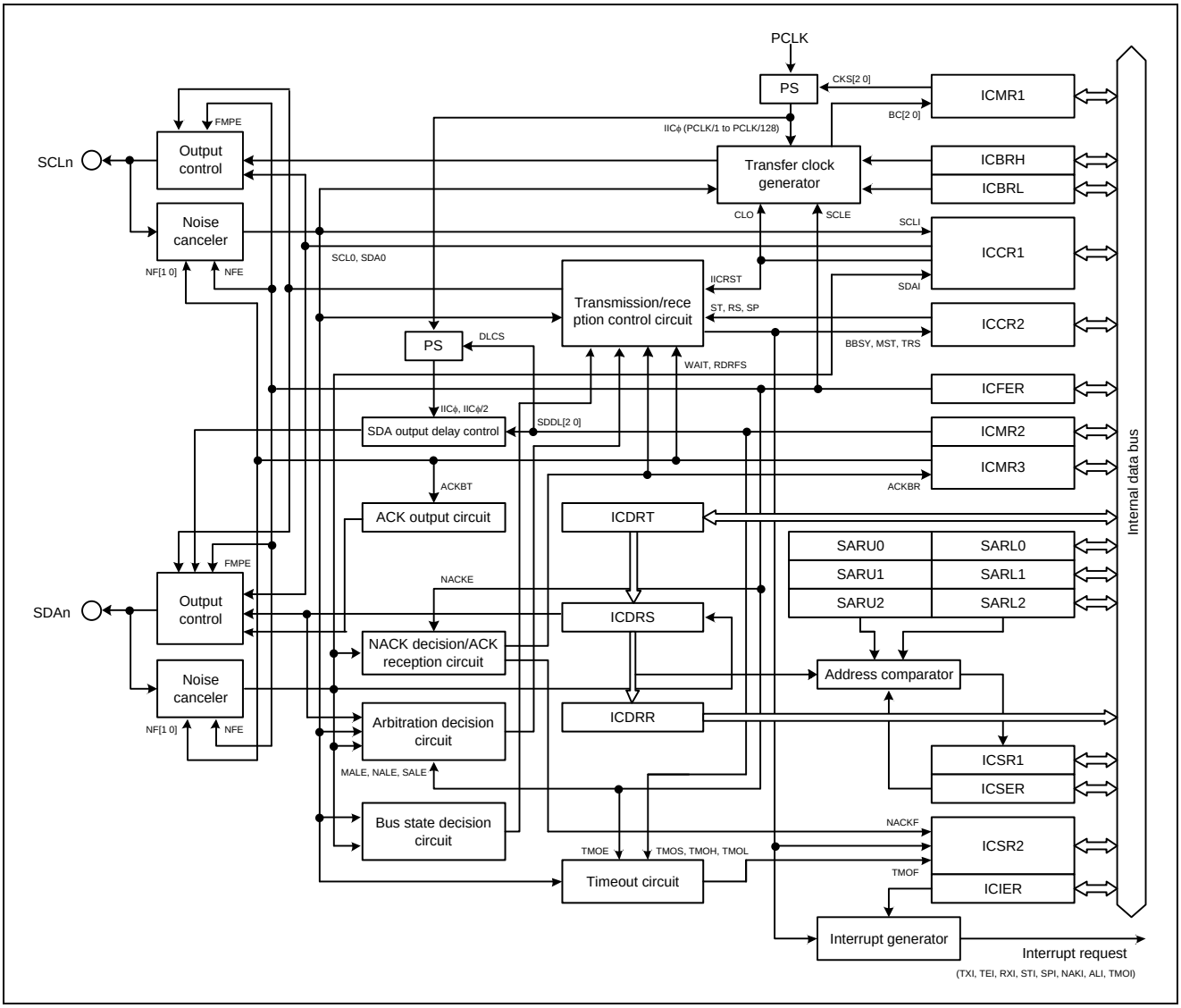



**Figure 22.1 Block Diagram of RIIC**



Downloaded from Arrow.com.

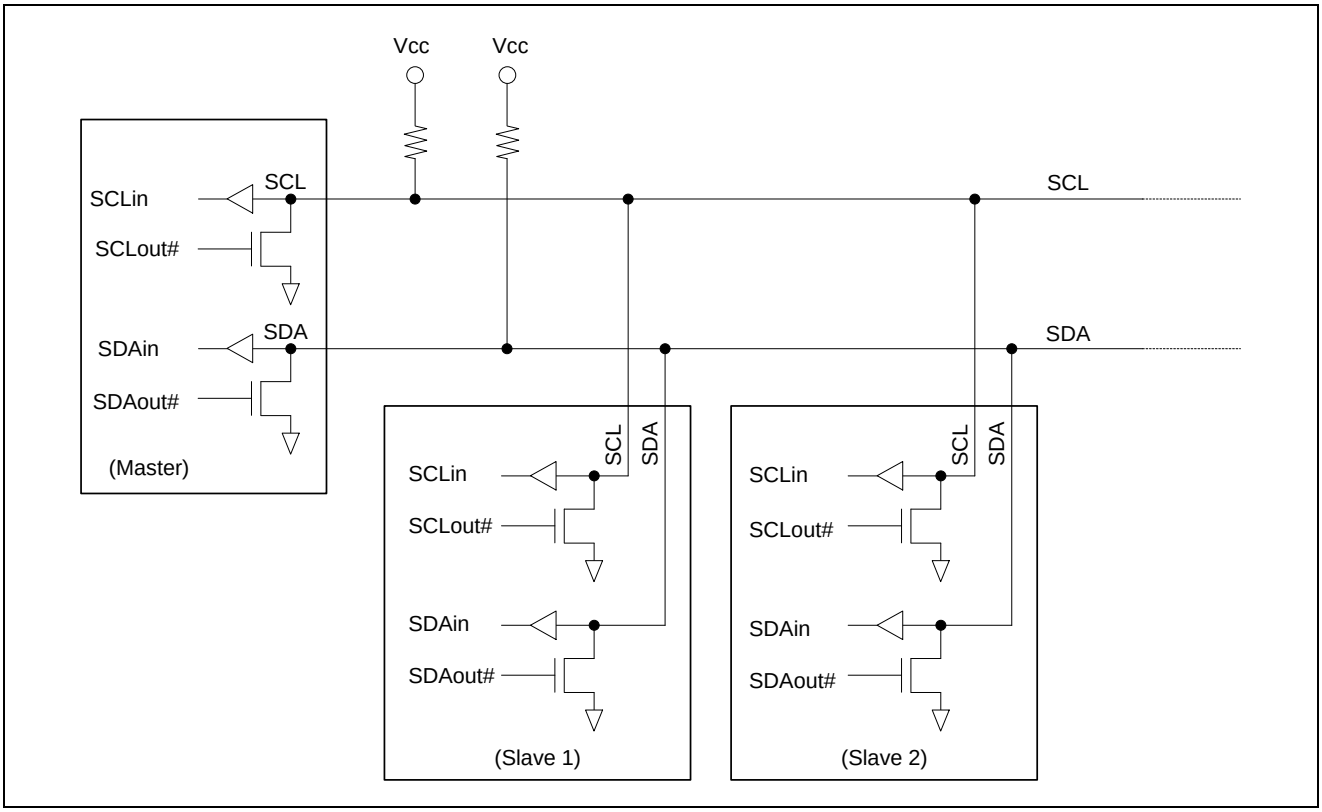



**Figure 22.2 Connections to the External Circuit by the I/O Pins (I²C Bus Configuration Example)**

**Table 22.2 Pin Configuration**

| Channel | Pin Name | I/O | Function |
|---|---|---|---|
| RIIC0 | SCL0 | I/O | RIIC0 serial clock I/O pin |
| | SDA0 | I/O | RIIC0 serial data I/O pin |
| RIIC1 | SCL1 | I/O | RIIC1 serial clock I/O pin |
| | SDA1 | I/O | RIIC1 serial data I/O pin |



Downloaded from Arrow.com.

## 22.2 Register Descriptions

Table 22.3 lists the registers of the RIIC.

**Table 22.3 Registers of the RIIC**

| Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| RIIC0 | I²C bus control register 1 | ICCR1 | 1Fh | 0008 8300h | 8 |
| | I²C bus control register 2 | ICCR2 | 00h | 0008 8301h | 8 |
| | I²C bus mode register 1 | ICMR1 | 08h | 0008 8302h | 8 |
| | I²C bus mode register 2 | ICMR2 | 06h | 0008 8303h | 8 |
| | I²C bus mode register 3 | ICMR3 | 00h | 0008 8304h | 8 |
| | I²C bus function enable register | ICFER | 72h | 0008 8305h | 8 |
| | I²C bus status enable register | ICSER | 09h | 0008 8306h | 8 |
| | I²C bus interrupt enable register | ICIER | 00h | 0008 8307h | 8 |
| | I²C bus status register 1 | ICSR1 | 00h | 0008 8308h | 8 |
| | I²C bus status register 2 | ICSR2 | 00h | 0008 8309h | 8 |
| | Slave address register L0 | SARL0 | 00h | 0008 830Ah | 8 |
| | Internal Counter for Timeout L | TMOCNTL | 0000h | 0008 830Ah | 16 |
| | Slave address register U0 | SARU0 | F0h | 0008 830Bh | 8 |
| | Internal Counter for Timeout U | TMOCNTU | 0000h | 0008 830Bh | 16 |
| | Slave address register L1 | SARL1 | 00h | 0008 830Ch | 8 |
| | Slave address register U1 | SARU1 | F0h | 0008 830Dh | 8 |
| | Slave address register L2 | SARL2 | 00h | 0008 830Eh | 8 |
| | Slave address register U2 | SARU2 | F0h | 0008 830Fh | 8 |
| | I²C bus bit rate low-level register | ICBRL | FFh | 0008 8310h | 8 |
| | I²C bus bit rate high-level register | ICBRH | FFh | 0008 8311h | 8 |
| | I²C bus transmit data register | ICDRT | FFh | 0008 8312h | 8 |
| | I²C bus receive data register | ICDRR | 00h | 0008 8313h | 8 |
| | I²C bus shift register | ICDRS | — | — | 8 |
| RIIC1 | I²C bus control register 1 | ICCR1 | 1Fh | 0008 8320h | 8 |
| | I²C bus control register 2 | ICCR2 | 00h | 0008 8321h | 8 |
| | I²C bus mode register 1 | ICMR1 | 08h | 0008 8322h | 8 |
| | I²C bus mode register 2 | ICMR2 | 06h | 0008 8323h | 8 |
| | I²C bus mode register 3 | ICMR3 | 00h | 0008 8324h | 8 |
| | I²C bus function enable register | ICFER | 72h | 0008 8325h | 8 |
| | I²C bus status enable register | ICSER | 09h | 0008 8326h | 8 |
| | I²C bus interrupt enable register | ICIER | 00h | 0008 8327h | 8 |
| | I²C bus status register 1 | ICSR1 | 00h | 0008 8328h | 8 |
| | I²C bus status register 2 | ICSR2 | 00h | 0008 8329h | 8 |
| | Slave address register L0 | SARL0 | 00h | 0008 832Ah | 8 |
| | Internal Counter for Timeout L | TMOCNTL | 0000h | 0008 832Ah | 16 |
| | Slave address register U0 | SARU0 | F0h | 0008 832Bh | 8 |
| | Internal Counter for Timeout U | TMOCNTU | 0000h | 0008 832Bh | 16 |
| | Slave address register L1 | SARL1 | 00h | 0008 832Ch | 8 |
| | Slave address register U1 | SARU1 | F0h | 0008 832Dh | 8 |
| | Slave address register L2 | SARL2 | 00h | 0008 832Eh | 8 |
| | Slave address register U2 | SARU2 | F0h | 0008 832Fh | 8 |
| | I²C bus bit rate low-level register | ICBRL | FFh | 0008 8330h | 8 |
| | I²C bus bit rate high-level register | ICBRH | FFh | 0008 8331h | 8 |
| | I²C bus transmit data register | ICDRT | FFh | 0008 8332h | 8 |
| | I²C bus receive data register | ICDRR | 00h | 0008 8333h | 8 |



Downloaded from Arrow.com.

| Channel | Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|
| RIIC1 | I²C bus shift register | ICDRS | — | — | 8 |



Downloaded from Arrow.com.

## 22.2.1 I²C Bus Control Register 1 (ICCR1)

Addresses: RIIC0.ICCR1 0008 8300h, RIIC1.ICCR1 0008 8320h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ICE | IICRST | CLO | SOWP | SCLO | SDAO | SCLI | SDAI |
| Value after reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SDAI | SDA Bus Input Monitor | 0: SDAn pin input is at a low level<br>1: SDAn pin input is at a high level | R |
| b1 | SCLI | SCL Bus Input Monitor | 0: SCLn pin input is at a low level<br>1: SCLn pin input is at a high level | R |
| b2 | SDAO | SDA Output Control*1*2 | Read:<br>0: SDAn pin output is at a low level<br>1: SDAn pin is in a high-impedance state<br>Write:<br>0: Changes the SDAn pin output to a low level<br>1: Changes the SDAn pin in a high-impedance state<br>(High level output is achieved through an external pull-up resistor.) | R/W *1, *2 |
| b3 | SCLO | SCL Output Control*1*2 | Read:<br>0: SCLn pin output is at a low level<br>1: SCLn pin is in a high-impedance state<br>Write:<br>0: Changes the SCLn pin output to a low level<br>1: Changes the SCLn pin in a high-impedance state<br>(High level output is achieved through an external pull-up resistor.) | R/W *1, *2 |
| b4 | SOWP | SCLO/SDAO Write Protect*2 | 0: Allows the SCLO and SDAO bits to be rewritten<br>(This bit is always read as 1.) | R/W *2 |
| b5 | CLO | Extra SCL Clock Cycle Output | 0: Does not output an extra SCL clock cycle (default)<br>1: Outputs an extra SCL clock cycle<br>(The CLO bit is cleared automatically after one clock cycle is output.) | R/W |
| b6 | IICRST | I²C Bus Interface Internal Reset | 0: Clears the RIIC reset or internal reset<br>1: Initiates the RIIC reset or internal reset<br>(Clears the bit counter and the SCLn/SDAn output latch) | R/W |
| b7 | ICE | I²C Bus Interface Enable | 0: Disables the RIIC (the SCLn pin and SDAn pin function as ports)<br>1: Enables the RIIC transfer function (the SCLn pin and SDAn pin drive the bus) | R/W |

Notes: 1. Do not write to these bits during communication. Changing a value during communication may cause a transmission or reception failure or an AL error.

2. To change the SDAO and SCLO bits, set the SOWP bit to 0 at the same timing to set the SDAO and SCLO bits to 0.



Downloaded from Arrow.com.

ICCR1 enables or disables the RIIC, resets the internal state of the RIIC, outputs an extra SCL clock cycle, and manipulates and monitors the SCLn pin and SDAn pin.

#### SDAI Bit (SDA Bus Input Monitor)

This bit indicates the input level of the SDAn pin.

#### SCLI Bit (SCL Bus Input Monitor)

This bit indicates the input level of the SCLn pin.

#### SDAO Bit (SDA Output Control)

This bit controls the output level of the SDAn pin. This bit also indicates the output state of the SDAn pin.

#### SCLO Bit (SCL Output Control)

This bit controls the output level of the SCLn pin. This bit also indicates the output state of the SCLn pin.

#### SOWP Bit (SCLO/SDAO Write Protect)

This bit controls the modification of the SCLO and SDAO bits.

#### CLO Bit (Extra SCL Clock Cycle Output)

This bit is used to output an extra SCL clock cycle for debugging or error processing.

Normally, set the bit to 0. Setting the bit to 1 in a normal communication state causes a communication error.

For details on this function, see section 22.11.2, Extra SCL Clock Cycle Output Function.

#### IICRST Bit (I²C Bus Interface Internal Reset)

This bit is used to reset the internal states of the RIIC.

Setting this bit to 1 initiates an RIIC reset or internal reset.

Whether an RIIC reset or internal reset is initiated is determined according to the combination with the ICE bit. Table 22.4 lists the resets of the RIIC.

The RIIC reset resets all registers including the BBSY flag in ICCR2 and internal states of the RIIC, and the internal reset resets the bit counter (BC[2:0] bits in ICMR1), the I²C bus shift register (ICDRS), and the I²C bus status registers (ICSR1 and ICSR2) as well as the internal states of the RIIC. For the reset conditions for each register, see section 22.14, Reset States.

An internal reset initiated with the IICRST bit set to 1 during operation (with the ICE bit set to 1) resets the internal states of the RIIC without initializing the port settings and the control and setting registers of the RIIC when the bus or RIIC hangs up due to a communication error.

If the RIIC hangs up in a low level output state, resetting the internal states cancels the low level output state and releases the bus with the SCLn pin and SDAn pin at a high impedance.



Downloaded from Arrow.com.

Note: If an internal reset is initiated using the IICRST bit for a bus hang-up occurred during communication with the master device in slave mode, the states may become different between the slave device and the master device (due to the difference in the bit counter information). For this reason, do not initiate an internal reset in slave mode, but initiate restoration processing from the master device. If an internal reset is necessary because the RIIC hangs up with the SCLn line in a low level output state in slave mode, initiate an internal reset and then issue a restart condition from the master device or resume communication from the start condition issuance after issuing a stop condition. If communication is restarted by initiating a reset solely in the slave device without issuing a start condition or restart condition from the master device, synchronization will be lost because the master and slave devices operate asynchronously.

**Table 22.4 RIIC Resets**

| IICRST | ICE | State | Specifications |
|---|---|---|---|
| 1 | 0 | RIIC reset | Resets all registers and internal states of the RIIC. |
| | 1 | Internal reset | Reset the BC[2:0] bits in ICMR1, and the ICSR1, ICSR2, ICDRS registers and the internal states of the RIIC. |

### ICE Bit (I²C Bus Interface Enable)

This bit is used to enable or disable the transfer operation of the RIIC.

When this bit is set to 0 to disable the RIIC, the SCLn pin and SDAn pin function as ports. An RIIC reset is initiated by setting the IICRST bit to 1 with the ICE bit set to 0, and an internal reset is initiated by setting the IICRST bit to 1 with the ICE bit set to 1.

To prevent unexpected communications, set the RIIC registers with the ICE bit set to 0 (to disable the RIIC), and set the ICE bit to 1 (to enable the transfer operation) after finishing all register settings.

Note: In addition to the I²C bus pin functions, other functions are also multiplexed onto the pins of the RX610 Group. To use the pins as I²C bus pins (SCLn pin and SDAn pin), disable the other multiplexed functions. Since both of the SCLn pin and SDAn pin of the I²C bus pins are I/O pins, set the corresponding Pn.DDR register to 0 (input) and set the Pn.ICR register to 1 (input buffer enabled).



Downloaded from Arrow.com.

### 22.2.2 I²C Bus Control Register 2 (ICCR2)

Addresses: RIIC0.ICCR2 0008 8301h, RIIC1.ICCR2 0008 8321h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | BBSY | MST | TRS | – | SP | RS | ST | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b1 | ST | Start Condition Issuance Request | 0: Does not request to issue a start condition<br>1: Requests to issue a start condition | R/W |
| b2 | RS | Restart Condition Issuance Request | 0: Does not request to issue a restart condition<br>1: Requests to issue a restart condition | R/W |
| b3 | SP | Stop Condition Issuance Request | 0: Does not request to issue a stop condition<br>1: Requests to issue a stop condition | R/W |
| b4 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b5 | TRS | Transmit/Receive Mode | 0: Receive mode<br>1: Transmit mode | R/W* |
| b6 | MST | Master/Slave Mode | 0: Slave mode<br>1: Transmit mode | R/W* |
| b7 | BBSY | Bus Busy Detection Flag | 0: The I²C bus is released (bus free state)<br>1: The I²C bus is occupied (bus busy state or in the bus free state) | R |

Note: * When the MTWP bit in ICMR1 is set to 1, the MST and TRS bits can be written to.

ICCR2 has a flag function that indicates whether or not the I²C bus is occupied and whether the RIIC is in transmit/receive or master/slave mode as well as a function to issue a start or stop condition.

**ST Bit (Start Condition Issuance Request)**

This bit is used to request transition to master mode and issuance of a start condition.

When this bit is set to 1 to request to issue a start condition, a start condition is issued when the BBSY flag is set to 0 (bus free).

For details on the start condition issuance, see section 22.10, Start Condition/Restart Condition/Stop Condition Issuing Function.

[Setting condition]

- When 1 is written to the ST bit

[Clearing conditions]

- When 0 is written to the ST bit
- When a start condition has been issued
- When the AL (arbitration lost) flag in ICSR2 is set to 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

Note: Set the ST bit to 1 (start condition issuance request) when the BBSY flag is set to 0 (bus free).

Note that arbitration may be lost if the ST bit is set to 1 (start condition issuance request) when the BBSY flag is set to 1 (bus busy).



Downloaded from Arrow.com.

### RS Bit (Restart Condition Issuance Request)

This bit is used to request that a restart condition be issued in master mode.

When this bit is set to 1 to request to issue a restart condition, a restart condition is issued when the BBSY flag is set to 1 (bus busy) and the MST bit is set to 1 (master mode).

For details on the restart condition issuance, see section 22.10, Start Condition/Restart Condition/Stop Condition Issuing Function.

[Setting condition]

- When 1 is written to the RS bit with the BBSY flag in ICCR2 set to 1

[Clearing conditions]

- When 0 is written to the RS bit
- When a restart condition has been issued or a start condition is detected
- When the AL (arbitration lost) flag in ICSR2 is set to 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

Notes: 1. Do not set the RS bit to 1 while issuing a stop condition.

2. If the RS bit is set to 1 (restart condition issuance request) in mode other than master mode, the restart condition is not issued in this mode but the restart condition issuance request bit remains set. If the operating mode changes to master mode with the bit not being cleared, the restart condition may be issued: this may hinder communications or cause an unexpected action.

### SP Bit (Stop Condition Issuance Request)

This bit is used to request that a stop condition be issued in master mode.

When this bit is set to 1 to request to issue a stop condition, a stop condition is issued when the BBSY flag is set to 1 (bus busy) and the MST bit is set to 1 (master mode).

For details on the stop condition issuance, see section 22.10, Start Condition/Restart Condition/Stop Condition Issuing Function.

[Setting condition]

- When 1 is written to the SP bit with both the BBSY flag and the MST bit in ICCR2 set to 1

[Clearing conditions]

- When 0 is written to the SP bit after reading SP = 1
- When a stop condition has been issued or a stop condition is detected
- When the AL (arbitration lost) flag in ICSR2 is set to 1
- When a start condition and a restart condition are detected
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

Notes: 1. Writing to the SP bit is not possible while the setting of the BBSY flag is 0 (bus free).

2. Do not set the SP bit to 1 while a restart condition is being issued.



Downloaded from Arrow.com.

### TRS Bit (Transmit/Receive Mode)

This bit indicates transmit or receive mode.

The RIIC is in receive mode when the TRS bit is set to 0 and is in transmit mode when the bit is set to 1. Combination of this bit and the MST bit indicates the operating mode of the RIIC.

The value of the TRS bit is automatically changed to the value for transmission mode or reception mode by detection or issuing of a start condition, setting or clearing of the R/W# bit, etc. Although writing to the TRS bit is possible when the MTWP bit in ICMR1 is set to 1, writing to this bit is not necessary during normal usage.

[Setting conditions]

- When a start condition is issued normally according to the start condition issuance request (when a start condition is detected with the ST bit set to 1)
- When the R/W# bit added to the slave address is set to 0 in master mode
- When the address received in slave mode matches the address enabled in ICSER, with the R/W# bit set to 1
- When 1 is written to the TRS bit with the MTWP bit in ICMR1 set to 1

[Clearing conditions]

- When a stop condition is detected
- The AL (arbitration lost) flag in ICSR2 being set to 1
- In master mode, reception of a slave address to which an R/W# bit with the value 1 is appended
- In slave mode, a match between the received address and the address enabled in ICSER when the value of the received R/W# bit is 0 (including cases where the received address is the general call address)
- In slave mode, a restart condition is detected (a start condition is detected with ICCR2.BBSY = 1 and ICCR2.MST = 0)
- When 0 is written to the TRS bit with the MTWP bit in ICMR1 set to 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

### MST Bit (Master/Salve Mode)

This bit indicates master or slave mode.

The RIIC is in slave mode when the MST bit is set to 0 and is in master mode when the bit is set to 1. Combination of this bit and the TRS bit indicates the operating mode of the RIIC.

The value of the MST bit is automatically changed to the value for master mode or slave mode by detection or issuing of a start condition, etc. Although writing to the MST bit is possible when the MTWP bit in ICMR1 is set to 1, writing to this bit is not necessary during normal usage.

[Setting conditions]

- When a start condition is issued normally according to the start condition issuance request (when a start condition is detected with the ST bit set to 1)
- When 1 is written to the MST bit with the MTWP bit in ICMR1 set to 1

[Clearing conditions]

- When a stop condition is detected
- When the AL (arbitration lost) flag in ICSR2 is set to 1
- When 0 is written to the MST bit with the MTWP bit in ICMR1 set to 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset



Downloaded from Arrow.com.

### BBSY Flag (Bus Busy Detection)

The BBSY flag indicates whether the I²C bus is occupied (bus busy) or released (bus free).

This bit is set to 1 when the SDAn line changes from high to low under the condition of SCLn = high, assuming that a start condition has been issued.

When the SDAn line changes from low to high under the condition of SCLn = high, this bit is cleared to 0 after the bus free time (specified in ICBRL) start condition is not detected, assuming that a stop condition has been issued.

[Setting condition]

- When a start condition is detected

[Clearing conditions]

- When the bus free time (specified in ICBRL) start condition is not detected after detecting a stop condition
- When 1 is written to the IICRST bit in ICCR1 with the ICE bit in ICCR1 set to 0 (RIIC reset)



Downloaded from Arrow.com.

## 22.2.3 I²C Bus Mode Register 1 (ICMR1)

Addresses: RIIC0.ICMR1 0008 8302h, RIIC1.ICMR1 0008 8322h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | MTWP | CKS[2:0] | | | BCWP | BC[2:0] | | |
| Value after reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b2 to b0 | BC[2:0] | Bit Counter* | b2 b1 b0 | R/W* |
| | | | 0 0 0: 9 bits | |
| | | | 0 0 1: 2 bits | |
| | | | 0 1 0: 3 bits | |
| | | | 0 1 1: 4 bits | |
| | | | 1 0 0: 5 bits | |
| | | | 1 0 1: 6 bits | |
| | | | 1 1 0: 7 bits | |
| | | | 1 1 1: 8 bits | |
| b3 | BCWP | BC Write Protect* | 0: Enables a value to be written in the BC[2:0] bits | R/W* |
| | | | (This bit is always read as 1.) | |
| b6 to b4 | CKS[2:0] | Internal Reference Clock Selection | b6 b5 b4 | R/W |
| | | | 0 0 0: PCLK/1 clock | |
| | | | 0 0 1: PCLK/2 clock | |
| | | | 0 1 0: PCLK/4 clock | |
| | | | 0 1 1: PCLK/8 clock | |
| | | | 1 0 0: PCLK/16 clock | |
| | | | 1 0 1: PCLK/32 clock | |
| | | | 1 1 0: PCLK/64 clock | |
| | | | 1 1 1: PCLK/128 clock | |
| b7 | MTWP | MST/TRS Write Protect | 0: Disables writing to the MST and TRS bits in ICCR2 | R/W |
| | | | 1: Enables writing to the MST and TRS bits in ICCR2 | |

Note: * Set the BCWP bit to 0 to rewrite the BC[2:0] bits. The BC[2:0] bits must be rewritten by using the MOV instruction.

ICMR1 specifies the internal reference clock source within the RIIC, indicates the number of bits to be transferred, and protects the MST and TRS bits in ICCR2 from being written.

### BC[2:0] Bits (Bit Counter)

These bits function as a counter that indicates the number of bits remaining to be transferred at the detection of a rising edge on the SCLn line. Although these bits are writable and readable, it is not necessary to access these bits under normal conditions.

To write to these bits, specify the number of bits to be transferred plus one (data is transferred with an additional acknowledge bit) between transferred frames when the SCLn line is at a low level.

The values of the BC[2:0] bits return to 000b at the end of a data transfer including the acknowledge bit or when a start condition including a restart condition is detected.



Downloaded from Arrow.com.

### BCWP Bit (BC Write Protect)

This bit enables a value to be written in the BC[2:0] bits.

### CKS[2:0] Bits (Internal Reference Clock Selection)

These bits select a reference clock source (IICϕ) inside the RIIC.

### MTWP Bit (MST/TRS Write Protect)

This bit controls the modification of the MST and TRS bits in ICCR2.



Downloaded from Arrow.com.

### 22.2.4 I²C Bus Mode Register 2 (ICMR2)

Addresses: RIIC0.ICMR2 0008 8303h, RIIC1.ICMR2 0008 8323h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DLCS | SDDL[2:0] | | | TMWE | TMOH | TMOL | TMOS |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TMOS | Timeout Detection Time Selection | 0: Long mode is selected<br>1: Short mode is selected | R/W |
| b1 | TMOL | Timeout L Count Control | 0: Count is disabled while the SCLn line is at a low level<br>1: Count is enabled while the SCLn line is at a low level | R/W |
| b2 | TMOH | Timeout H Count Control | 0: Count is disabled while the SCLn line is at a high level<br>1: Count is enabled while the SCLn line is at a high level | R/W |
| b3 | TMWE | Enable Writing to the Internal Counter for Timeout | 0: Writing to the internal counter for timeout is disabled<br>1: Writing to the internal counter for timeout is enabled<br>When this bit is set to “1”, the address of timeout internal counter (TMOCNTL/U) is allocated to the address of SARL0/SARU0. | R/W |
| b6 to b4 | SDDL[2:0] | SDA Output Delay Counter | • When ICMR2.DLCS = 0 (IICϕ)<br>b6 b4<br>0 0 0: No output delay<br>0 0 1: 1 IICϕ cycle<br>0 1 0: 2 IICϕ cycles<br>0 1 1: 3 IICϕ cycles<br>1 0 0: 4 IICϕ cycles<br>1 0 1: 5 IICϕ cycles<br>1 1 0: 6 IICϕ cycles<br>1 1 1: 7 IICϕ cycles<br>• When ICMR2.DLCS = 1 (IICϕ/2)<br>b6 b4<br>0 0 0: No output delay<br>0 0 1: 1 or 2 IICϕ cycles<br>0 1 0: 3 or 4 IICϕ cycles<br>0 1 1: 5 or 6 IICϕ cycles<br>1 0 0: 7 or 8 IICϕ cycles<br>1 0 1: 9 or 10 IICϕ cycles<br>1 1 0: 11 or 12 IICϕ cycles<br>1 1 1: 13 or 14 IICϕ cycles | R/W |
| b7 | DLCS | SDA Output Delay Clock Source Selection | 0: The internal reference clock (IICϕ) is selected as the clock source of the SDA output delay counter<br>1: The internal reference clock divided by 2 (IICϕ/2) is selected as the clock source of the SDA output delay counter | R/W |

ICMR2 has a timeout detection function and an SDA output delay function.



Downloaded from Arrow.com.

### TMOS Bit (Timeout Detection Time Selection)

This bit is used to select long mode or short mode for the timeout detection time when the timeout detection function is enabled (TMOE bit = 1 in ICFER). When this bit is set to 0, long mode is selected. When this bit is set to 1, short mode is selected. In long mode, the timeout detection internal counter functions as a 16 bit-counter. In short mode, the counter functions as a 14 bit-counter. While the SCLn line is in the state that enables this counter as specified by bits TMOH and TMOL, the counter counts up in synchronization with the internal reference clock (IICϕ) as a count source.

For details on the timeout detection function, see section 22.11.1, Timeout Detection Function.

### TMOL Bit (Timeout L Count Control)

This bit is used to enable or disable the internal counter of the timeout detection function to count up while the SCLn line is held low when the timeout detection function is enabled (TMOE bit = 1 in ICFER).

### TMOH Bit (Timeout H Count Control)

This bit is used to enable or disable the internal counter of the timeout detection function to count up while the SCLn line is held high when the timeout detection function is enabled (TMOE bit = 1 in ICFER).

### TMWE Bit (Enable Writing to the Internal Counter for Timeout)

This bit is used to select whether the slave address register (SARL0/SARU0) is assigned to the internal counter for timeout (TMOCNTL/TMOCNTU).

### SDDL[2:0] Bits (SDA Output Delay Setup Counter)

The SDA output can be delayed by the SDDL[2:0] setting. This counter works with the clock source selected by the DLCS bit. The setting of this function can be used for all types of SDA output, including the transmission of the acknowledge bit.
For details on this function , see section 22.5, Facility for Delaying SDA Output.

Notes: 1. Set the SDA output delay time to meet the I²C bus standard (within the data enable time/acknowledge enable time*²) or the SMBus standard (within the data hold time: 300 ns or more, and SCL-clock low-level period – the data setup time: 250 ns). Note that, if a value outside the standard is set, communication with communication devices may malfunction or it may seemingly become a start condition or stop condition depending on the bus state.

2. Data enable time/acknowledge enable time
   3,450 ns (up to 100 kbps: standard mode [Sm])
   900 ns (up to 400 kbps: fast mode [fm])
   450 ns (up to 1 Mbps: fast mode plus [fm+])

### DLCS bit (SDA Output Delay Clock Source Selection)

This bit is used to select the internal reference clock (IICϕ) or the internal reference clock divided by 2 (IICϕ/2) as the clock source of the SDA output delay time.



Downloaded from Arrow.com.

## 22.2.5 I²C Bus Mode Register 3 (ICMR3)

Addresses: RIIC0.ICMR3 0008 8304h, RIIC1.ICMR3 0008 8324h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | SMBS | WAIT | RDRFS | ACKWP | ACKBT | ACKBR | NF[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | NF[1:0] | Noise Filter Stage Selection | b1 b0<br>0 0: Noise of up to 1-PCLK range is filtered out (single-stage filter)<br>0 1: Noise of up to 2-PCLK range is filtered out (2-stage filter)<br>1 0: Noise of up to 3-PCLK range is filtered out (3-stage filter)<br>1 1: Noise of up to 4-PCLK range is filtered out (4-stage filter) | R/W |
| b2 | ACKBR | Receive Acknowledge | 0: A 0 is received as the acknowledge bit (ACK reception)<br>1: A 1 is received as the acknowledge bit (NACK reception) | R |
| b3 | ACKBT | Transmit Acknowledge | 0: A 0 is sent as the acknowledge bit (ACK transmission)<br>1: A 1 is sent as the acknowledge bit (NACK transmission) | R/W *1 |
| b4 | ACKWP | ACKBT Write Protect | 0: Modification of the ACKBT bit is disabled<br>1: Modification of the ACKBT bit is enabled*1 | W*1 |
| b5 | RDRFS | RDRF Flag Set Timing Selection*2 | 0: The RDRF flag is set at the rising edge of the ninth SCL clock cycle<br>(The SCLn line is not held low at the falling edge of the eighth clock cycle.)<br>1: The RDRF flag is set at the rising edge of the eighth SCL clock cycle<br>(The SCLn line is held low at the falling edge of the eighth clock cycle.)<br>• Low-hold is released by writing a value to the ACKBT bit | R/W *2 |
| b6 | WAIT | WAIT | 0: No WAIT<br>(The period between ninth clock cycle and first clock cycle is not held low.)<br>1: WAIT<br>(The period between ninth clock cycle and first clock cycle is held low.)<br>• Low-hold is released by reading ICDRR | R/W *2 |
| b7 | SMBS | SMBus/I²C Bus Selection | 0: The I²C bus is selected<br>1: The SMBus is selected | R/W |

Notes: 1. If it is attempted to write 1 to both ACKWP and ACKBT bits, the ACKBT bit cannot be set to 1.

2. The WAIT and RDRFS bits are valid only in receive mode (invalid in transmit mode).

ICMR3 has functions to send/receive acknowledge and to select the RDRF set timing in RIIC receive operation, WAIT operation, and the SCLn pin and SDAn pin input level of the RIIC.



Downloaded from Arrow.com.

### NF[1:0] Bits (Noise Filter Stage Selection)

These bits are used to select the number of stages of the digital noise filter.

Note: Set the noise range to be filtered out by the noise filter within a range less than the SCLn line high-level period or low-level period. If the noise range is set to a value of (SCL clock width: high-level period or low-level period, whichever is shorter) – [1.5 internal reference clock (IICϕ) cycles + analog noise filter: 120 ns (reference value)] or more, the SCL clock is regarded as noise by the noise filter function of the RIIC, which may prevent the RIIC from operating normally.

### ACKBR Bit (Receive Acknowledge)

This bit is used to store the acknowledge bit information received from the receive device in transmit mode.

[Setting condition]

- When a 1 is received as the acknowledge bit with the TRS bit in ICCR2 set to 1

[Clearing conditions]

- When a 0 is received as the acknowledge bit with the TRS bit in ICCR2 set to 1
- When 1 is written to the IICRST bit in ICCR1 while the ICE bit in ICCR1 is 0 (RIIC reset)

### ACKBT Bit (Transmit Acknowledge)

This bit is used to set the bit to be sent at the acknowledge timing in receive mode.

[Setting condition]

- When 1 is written to this bit with the ACKWP bit set to 1

[Clearing conditions]

- When 0 is written to this bit with the ACKWP bit set to 1
- When stop condition issuance is detected (when a stop condition is detected with the SP bit in ICCR2 set to 1)
- When 1 is written to the IICRST bit in ICCR1 while the ICE bit in ICCR1 is 0 (RIIC reset)

Note: The ACKBT bit must be modified while the ACKWP bit is 1. If the ACKBT bit is modified with the ACKWP bit cleared to 0, writing to the ACKBT bit is disabled.

### ACKWP Bit (ACKBT Write Protect)

This bit is used to control the modification of the ACKBT bit.

### RDRFS Bit (RDRF Flag Set Timing Selection)

This bit is used to select the RDRF flag set timing in receive mode and also to select whether to hold the SCLn line low at the falling edge of the eighth SCL clock cycle.

When the RDRFS bit is 0, the SCLn line is not held low at the falling edge of the eighth SCL clock cycle, and the RDRF flag is set to 1 at the rising edge of the ninth SCL clock cycle.

When the RDRFS bit is 1, the RDRF flag is set to 1 at the rising edge of the eighth SCL clock cycle and the SCLn line is held low at the falling edge of the eighth SCL clock cycle. The low-hold of the SCLn line is released by writing a value to the ACKBT bit.

After data is received with this setting, the SCLn line is automatically held low before the acknowledge bit is sent. This enables processing to send ACK (ACKBT = 0) or NACK (ACKBT = 1) according to receive data.



Downloaded from Arrow.com.

### WAIT Bit (WAIT)

This bit is used to control whether to hold the period between the ninth SCL clock cycle and the first SCL clock cycle low until the receive data buffer (ICDRR) is completely read each time single-byte data is received in receive mode.

When the WAIT bit is 0, the receive operation is continued without holding the period between the ninth and the first SCL clock cycle low. When both the RDRFS and WAIT bits are 0, continuous receive operation is enabled with the double buffer.

When the WAIT bit is 1, the SCLn line is held low from the falling edge of the ninth clock cycle until the ICDRR value is read each time single-byte data is received. This enables receive operation in byte units.

Note: When the value of the WAIT bit is to be read, be sure to read the ICDRR beforehand.

### SMBS Bit (SMBus/I²C Bus Selection)

Setting this bit to 1 selects the SMBus and enables the HOAE bit in ICSER.



Downloaded from Arrow.com.

## 22.2.6 I²C Bus Function Enable Register (ICFER)

Addresses: RIIC0.ICFER 0008 8305h, RIIC1.ICFER 0008 8325h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FMPE | SCLE | NFE | NACKE | SALE | NALE | MALE | TMOE |
| Value after reset: | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TMOE | Timeout Detection Function Enable | 0: The timeout detection function is disabled<br>1: The timeout detection function is enabled | R/W |
| b1 | MALE | Master Arbitration Lost Detection Enable | 0: Master arbitration lost detection is disabled<br>(Disables the arbitration lost detection function and does not clear the MST and TRS bits in ICCR2 automatically when arbitration is lost.)<br>1: Master arbitration lost detection is enabled<br>(Enables the arbitration lost detection function and clears the MST and TRS bits in ICCR2 automatically when arbitration is lost.) | R/W |
| b2 | NALE | NACK Transmission Arbitration Lost Detection Enable | 0: NACK transmission arbitration lost detection is disabled<br>1: NACK transmission arbitration lost detection is enabled | R/W |
| b3 | SALE | Slave Arbitration Lost Detection Enable | 0: Slave arbitration lost detection is disabled<br>1: Slave arbitration lost detection is enabled | R/W |
| b4 | NACKE | NACK Reception Transfer Suspension Enable | 0: Transfer operation is not suspended during NACK reception (transfer suspension disabled)<br>1: Transfer operation is suspended during NACK reception (transfer suspension enabled) | R/W |
| b5 | NFE | Digital Noise Filter Circuit Enable | 0: No digital noise filter circuit is used<br>1: A digital noise filter circuit is used | R/W |
| b6 | SCLE | SCL Synchronous Circuit Enable | 0: No SCL synchronous circuit is enabled<br>1: An SCL synchronous circuit is enabled | R/W |
| b7 | FMPE | Fast-mode Plus Enable | 0: No fm+ slope control circuit is used for the SCLn pin and SDAn pin<br>1: An fm+ slope control circuit is used for the SCLn pin and SDAn pin | R/W |

ICFER enables or disables the timeout detection function, the arbitration lost detection function, and the receive operation suspension function during NACK reception, and selects the use of a digital noise filter circuit and SCL synchronous circuit.

### TMOE Bit (Timeout Detection Function Enable)

This bit is used to enable or disable the timeout detection function.

For details on the timeout detection function, see section 22.11.1, Timeout Detection Function.

### MALE Bit (Master Arbitration Lost Detection Enable)

This bit is used to specify whether to use the arbitration lost detection function in master mode. Normally, set this bit to 1.



Downloaded from Arrow.com.

### NALE Bit (NACK Transmission Arbitration Lost Detection Enable)

This bit is used to specify whether to cause arbitration to be lost when ACK is detected during transmission of NACK in receive mode (such as when slaves with the same address exist on the bus or when two or more masters select the same slave device simultaneously with different number of receive bytes).

### SALE Bit (Slave Arbitration Lost Detection Enable)

This bit is used to specify whether to cause arbitration to be lost when a value different from the value being transmitted is detected on the bus in slave transmit mode (such as when slaves with the same address exist on the bus or when a mismatch with the transmit data occurs due to noise).

### NACKE Bit (NACK Reception Transfer Suspension Enable)

This bit is used to specify whether to continue or discontinue the transfer operation when NACK is received from the slave device in transmit mode. Normally, set this bit to 1.

When NACK is received with the NACKE bit set to 1, the next transfer operation is suspended.

When the NACKE bit is 0, the next transfer operation is continued regardless of the received acknowledge content.

For details on the NACK reception transfer suspension function, see section 22.8.2, NACK Reception Transfer Suspension Function.

### NFE Bit (Digital Noise Filter Circuit Enable)

This bit is used to specify whether to use a digital noise filter circuit.

### SCLE Bit (SCL Synchronous Circuit Enable)

This bit is used to specify whether to synchronize the SCL clock with the SCL input clock. Normally, set this bit to 1.

When the SCLE bit is cleared to 0 (SCL synchronous circuit is invalid), the RIIC does not synchronize the SCL clock with the SCL input clock (by detecting the SCLn line level) for the SCL clock output operation in master mode, and the RIIC outputs the SCL clock with the transfer rate set in ICBRH and ICBRL regardless of the SCLn line state. For this reason, if the bus load of the I²C bus line is much larger than the specification value or if the SCL clock output overlaps in multiple masters, the short-cycle SCL clock that does not meet the specification may be output. When SCL synchronous circuit is invalid, it also affects the issuance of a start condition, restart condition, and stop condition, and the continuous output of extra SCL clock cycles.

This bit must not be cleared to 0 except for checking the output of the transfer rate that was set during debugging.

### FMPE Bit (Fast-mode Plus Enable)

This bit is used to specify whether to use a slope control circuit for Fast-mode Plus[fm+].

When this bit is set to 1, a slope control circuit conforming to the Fast-mode Plus[fm+] slope control standard (tof) of the I²C bus is selected. When this bit is cleared to 0, a slope control circuit conforming to the Standard-mode[Sm] and Fast-mode[fm] slope control standard (tof) of the I²C bus is selected.

Set this bit to 1 when using the transmission rate within a range up to 1 Mbps (Fast-mode Plus[fm+]) of the I²C bus standard. Clear this bit to 0 when using the transmission rate at other rates (up to 100 kbps[Sm], up to 400 kbps[fm]) or for SMBus (10 to 100 kbps).



Downloaded from Arrow.com.

## 22.2.7 I²C Bus Status Enable Register (ICSER)

Addresses: RIIC0.ICSER 0008 8306h, RIIC1.ICSER 0008 8326h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | HOAE | – | DIDE | – | GCAE | SAR2E | SAR1E | SAR0E |
| Value after reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SAR0E | Slave Address Register 0 Enable | 0: Slave address in SARL0 and SARU0 is disabled<br>1: Slave address in SARL0 and SARU0 is enabled | R/W |
| b1 | SAR1E | Slave Address Register 1 Enable | 0: Slave address in SARL1 and SARU1 is disabled<br>1: Slave address in SARL1 and SARU1 is enabled | R/W |
| b2 | SAR2E | Slave Address Register 2 Enable | 0: Slave address in SARL2 and SARU2 is disabled<br>1: Slave address in SARL2 and SARU2 is enabled | R/W |
| b3 | GCAE | General Call Address Enable | 0: General call address detection is disabled<br>1: General call address detection is enabled | R/W |
| b4 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b5 | DIDE | Device-ID Address Detection Enable | 0: Device-ID address detection is disabled<br>1: Device-ID address detection is enabled | R/W |
| b6 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b7 | HOAE | Host Address Enable | 0: Host address detection is disabled<br>1: Host address detection is enabled | R/W |

ICSER enables or disables comparison of slave addresses, general call address detection, device-ID command detection, and host address detection.

### SARyE Bit (Slave Address Register y Enable) (y = 0 to 2)

This bit is used to enable or disable the received slave address and the slave address set in SARLy and SARUy.

When this bit is set to 1, the slave address set in SARLy and SARUy is enabled and is compared with the received slave address.

When this bit is cleared to 0, the slave address set in SARLy and SARUy is disabled and is ignored even if it matches the received slave address.



Downloaded from Arrow.com.

### GCAE Bit (General Call Address Enable)

This bit is used to specify whether to ignore the general call address (0000 000b + 0 [W]: All 0) when it is received.

When this bit is set to 1, if the received slave address matches the general call address, the RIIC recognizes the received slave address as the general call address independently of the slave addresses set in SARLy and SARUy (y = 0 to 2) and performs data receive operation.

When this bit is cleared to 0, the received slave address is ignored even if it matches the general call address.

### DIDE Bit (Device-ID Address Detection Enable)

This bit is used to specify whether to recognize and execute the Device-ID address when a device ID (1111 100b) is received in the first frame after a start condition or restart condition is detected.

When this bit is set to 1, if the received first frame matches the device ID, the RIIC recognizes that the Device-ID address has been received. When the following R/W# bit is 0 [W], the RIIC recognizes the second and the following frames as slave addresses and continues the receive operation.

When this bit is cleared to 0, the RIIC ignores the received first frame even if it matches the device ID address and recognizes the first frame as a normal slave address.

For details on the device-ID address detection, see section 22.7.3, Device-ID Address Detection.

### HOAE Bit (Host Address Enable)

This bit is used to specify whether to ignore received host address (0001 000b) when the SMBS bit in ICMR3 is 1.

When this bit is set to 1 while the SMBS bit in ICMR3 is 1, if the received slave address matches the host address, the RIIC recognizes the received slave address as the host address independently of the slave addresses set in SARLy and SARUy (y = 0 to 2) and performs the receive operation.

When the SMBS bit in ICMR3 or the HOAE bit is cleared to 0, the received slave address is ignored even if it matches the host address.



Downloaded from Arrow.com.

## 22.2.8 I²C Bus Interrupt Enable Register (ICIER)

Addresses: RIIC0.ICIER 0008 8307h, RIIC1.ICIER 0008 8327h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | TEIE | RIE | NAKIE | SPIE | STIE | ALIE | TMOIE |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TMOIE | Timeout Interrupt Enable | 0: Timeout interrupt request (TMOI) is disabled<br>1: Timeout interrupt request (TMOI) is enabled | R/W |
| b1 | ALIE | Arbitration Lost Interrupt Enable | 0: Arbitration lost interrupt request (ALI) is disabled<br>1: Arbitration lost interrupt request (ALI) is enabled | R/W |
| b2 | STIE | Start Condition Detection Interrupt Enable | 0: Start condition detection interrupt request (STI) is disabled<br>1: Start condition detection interrupt request (STI) is enabled | R/W |
| b3 | SPIE | Stop Condition Detection Interrupt Enable | 0: Stop condition detection interrupt request (SPI) is disabled<br>1: Stop condition detection interrupt request (SPI) is enabled | R/W |
| b4 | NAKIE | NACK Reception Interrupt Enable | 0: NACK reception interrupt request (NAKI) is disabled<br>1: NACK reception interrupt request (NAKI) is enabled | R/W |
| b5 | RIE | Receive Data Full Interrupt Enable | 0: Receive data full interrupt request (ICRXI) is disabled<br>1: Receive data full interrupt request (ICRXI) is enabled | R/W |
| b6 | TEIE | Transmit End Interrupt Enable | 0: Transmit end interrupt request (ICTEI) is disabled<br>1: Transmit end interrupt request (ICTEI) is enabled | R/W |
| b7 | TIE | Transmit Data Empty Interrupt Enable | 0: Transmit data empty interrupt request (ICTXI) is disabled<br>1: Transmit data empty interrupt request (ICTXI) is enabled | R/W |

ICIER enables or disables various interrupt sources.

### TMOIE Bit (Timeout Interrupt Enable)

This bit is used to enable or disable timeout interrupt requests (TMOI) when the TMOF flag in ICSR2 is set to 1. A TMOI interrupt request is canceled by clearing the TMOF flag or the TMOIE bit to 0.

### ALIE Bit (Arbitration Lost Interrupt Enable)

This bit is used to enable or disable arbitration lost interrupt requests (ALI) when the AL flag in ICSR2 is set to 1. An ALI interrupt request is canceled by clearing the AL flag or the ALIE bit to 0.

### STIE Bit (Start Condition Detection Interrupt Enable)

This bit is used to enable or disable start condition detection interrupt requests (STI) when the START flag in ICSR2 is set to 1. An STI interrupt request is canceled by clearing the START flag or the STIE bit to 0.



Downloaded from Arrow.com.

#### SPIE Bit (Stop Condition Detection Interrupt Enable)

This bit is used to enable or disable stop condition detection interrupt requests (SPI) when the STOP flag in ICSR2 is set to 1. An SPI interrupt request is canceled by clearing the STOP flag or the SPIE bit to 0.

#### NAKIE Bit (NACK Reception Interrupt Enable)

This bit is used to enable or disable NACK reception interrupt requests (NAKI) when the NACKF flag in ICSR2 is set to 1. An NAKI interrupt request is canceled by clearing the NACKF flag or the NAKIE bit to 0.

#### RIE Bit (Receive Data Full Interrupt Enable)

This bit is used to enable or disable receive data full interrupt requests (ICRXI) when the RDRF flag in ICSR2 is set to 1.

#### TEIE Bit (Transmit End Interrupt Enable)

This bit is used to enable or disable transmit end interrupts (ICTEI) when the TDRE flag in ICSR2 is set to 1. An ICTEI interrupt request is canceled by clearing the TEND flag or the TEIE bit to 0.

#### TIE Bit (Transmit Data Empty Interrupt Enable)

This bit is used to enable or disable transmit data empty interrupts (ICTXI) when the TDRE flag in ICSR2 is set to 1.



Downloaded from Arrow.com.

## 22.2.9 I²C Bus Status Register 1 (ICSR1)

Addresses: RIIC0.ICSR1 0008 8308h, RIIC1.ICSR1 0008 8328h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | HOA | — | DID | — | GCA | AAS2 | AAS1 | AAS0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | AAS0 | Slave Address 0 Detection Flag | 0: Slave address 0 is not detected<br>1: Slave address 0 is detected<br>• This bit is set to 1 when the received slave address matches the SVA[7:1] value in SARL0 while the FS bit in SARU0 is 0 (7-bit address format selected).<br>• This bit is set to 1 when the received slave address matches a value of (1111 0b + SVA [9:8] in SARU0) and the following address matches the SARL0 value while the FS bit in SARU0 is 1 (10-bit address format selected).<br>(This bit is set at the rising edge of the ninth SCL clock cycle in the SARL0 match determination frame.) | R/(W) * |
| b1 | AAS1 | Slave Address 1 Detection Flag | 0: Slave address 1 is not detected<br>1: Slave address 1 is detected<br>• This bit is set to 1 when the received slave address matches the SVA[7:1] value in SARL1 while the FS bit in SARU1 is 0 (7-bit address format selected).<br>• This bit is set to 1 when the received slave address matches a value of (1111 0b + SVA [9:8] in SARU1) and the following address matches the SARL1 value while the FS bit in SARU1 is 1 (10-bit address format selected).<br>(This bit is set at the rising edge of the ninth SCL clock cycle in the SARL1 match determination frame.) | R/(W) * |
| b2 | AAS2 | Slave Address 2 Detection Flag | 0: Slave address 2 is not detected<br>1: Slave address 2 is detected<br>• This bit is set to 1 when the received slave address matches the SVA[7:1] value in SARL2 while the FS bit in SARU2 is 0 (7-bit address format selected).<br>• This bit is set to 1 when the received slave address matches a value of (1111 0b + SVA [9:8] in SARU2) and the following address matches the SARL2 value while the FS bit in SARU2 is 1 (10-bit address format selected).<br>(This bit is set at the rising edge of the ninth SCL clock cycle in the SARL2 match determination frame.) | R/(W) * |
| b3 | GCA | General Call Address Detection Flag | 0: General call address is not detected<br>1: General call address is detected<br>• This bit is set to 1 when the received slave address matches the general call address (all 0). | R/(W) * |
| b4 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b5 | DID | Device-ID Command Detection Flag | 0: Device-ID command is not detected<br>1: Device-ID command is detected<br>• This bit is set to 1 when the first frame received immediately after a start condition is detected matches a value of (device ID (1111 100b) + 0 [W]). | R/(W) * |
| b6 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b7 | HOA | Host Address Detection Flag | 0: Host address is not detected<br>1: Host address is detected<br>• This bit is set to 1 when the received slave address matches the host address (0001 000b). | R/(W) * |

Note: * Only 0 can be written to clear the flag.

ICSR1 indicates various address detection statuses.

### AASy Flag (Slave Address y Detection) (ym = 0 to 2)

[Setting conditions]

<For 7-bit address format: SARUy.FS = 0>

- When the received slave address matches the SVA[7:1] value in SARLy with the SARyE bit in ICSER set to 1 (slave address m detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

<For 10-bit address format: SARUy.FS = 1>

- When the received slave address matches a value of (1111 0b + SVA [9:8] in SARUy) and the following address matches the SARLy value with the SARyE bit in ICSER set to 1 (slave address m detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the AASy bit after reading AASy = 1
- When a stop condition is detected
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

<For 7-bit address format: SARUy.FS = 0>

- When the received slave address does not match the SVA[7:1] value in SARLy with the SARyE bit in ICSER set to 1 (slave address m detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.

<For 10-bit address format: SARUy.FS = 1>

- When the received slave address does not match a value of (1111 0b + SVA [9:8] in SARUy) with the SARyE bit in ICSER set to 1 (slave address m detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When the received slave address matches a value of (1111 0b + SVA [9:8] in SARUy) and the following address does not match the SARLy value with the SARyE bit in ICSER set to 1 (slave address m detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.



Downloaded from Arrow.com.

### GCA Flag (General Call Address Detection)

[Setting condition]

- When the received slave address matches the general call address (0000 000b + 0 [W]) with the GCAE bit in ICSER set to 1 (general call address detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the GCA bit after reading GCA = 1
- When a stop condition is detected
- When the received slave address does not match the general call address (0000 000b + 0 [W]) with the GCAE bit in ICSER set to 1 (general call address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

### DID Flag (Device-ID Address Detection)

[Setting condition]

- When the first frame received immediately after a start condition or restart condition is detected matches a value of (device ID (1111 100b) + 0 [W]) with the DIDE bit in ICSER set to 1 (Device-ID address detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the DID bit after reading DID = 1
- When a stop condition is detected
- When the first frame received immediately after a start condition or restart condition is detected does not match a value of (device ID (1111 100b)) with the DIDE bit in ICSER set to 1 (Device-ID address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When the first frame received immediately after a start condition or restart condition is detected matches a value of (device ID (1111 100b) + 0 [W]) and the second frame does not match any of slave addresses 0 to 2 with the DIDE bit in ICSER set to 1 (Device-ID address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset



Downloaded from Arrow.com.

### HOA Flag (Host Address Detection)

[Setting condition]

- When the received slave address matches the host address (0001 000b) with the HOAE bit in ICSER set to 1 (host address detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the HOA bit after reading HOA = 1
- When a stop condition is detected
- When 0 is written to the SMBS bit in ICMR3 or the HOAE bit in ICSER
- When the received slave address does not match the host address (0001 000b) with the HOAE bit in ICSER set to 1 (host address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset



Downloaded from Arrow.com.

## 22.2.10 I²C Bus Status Register 2 (ICSR2)

Addresses: RIIC0.ICSR2 0008 8309h, RIIC1.ICSR2 0008 8329h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | TEND | RDRF | NACKF | STOP | START | AL | TMOF |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | TMOF | Timeout Detection Flag | 0: Timeout is not detected | R/(W) |
| | | | 1: Timeout is detected | * |
| b1 | AL | Arbitration Lost Flag | 0: Arbitration is not lost | R/(W) |
| | | | 1: Arbitration is lost | * |
| b2 | START | Start Condition Detection Flag | 0: Start condition is not detected | R/(W) |
| | | | 1: Start condition is detected | * |
| b3 | STOP | Stop Condition Detection Flag | 0: Stop condition is not detected | R/(W) |
| | | | 1: Stop condition is detected | * |
| b4 | NACKF | NACK Detection Flag | 0: NACK is not detected | R/(W) |
| | | | 1: NACK is detected | * |
| b5 | RDRF | Receive Data Full Flag | 0: ICDRR contains no receive data | R/(W) |
| | | | 1: ICDRR contains receive data | * |
| b6 | TEND | Transmit End Flag | 0: Data is being transmitted | R/(W) |
| | | | 1: Data has been transmitted | * |
| b7 | TDRE | Transmit Data Empty Flag | 0: ICDRT contains transmit data | R |
| | | | 1: ICDRT contains no transmit data | |

Note: * Only 0 can be written to clear the flag.

ICSR2 indicates various interrupt request flags and statuses.

### TMOF Flag (Timeout Detection)

This flag is set to 1 when the RIIC recognizes timeout after the SCLn line state remains unchanged for a certain period.

[Setting condition]

- When the SCLn line state remains unchanged for the period specified by bits TMOH, TMOL, and TMOS in ICMR2 with the TMOE bit in ICFER set to 1 (timeout detection function enabled) in master mode or in the slave specification state.

[Clearing conditions]

- When 0 is written to the TMOF bit after reading TMOF = 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset



Downloaded from Arrow.com.

### AL Flag (Arbitration Lost Flag)

This flag shows that bus mastership has been lost (loss in arbitration) due to a bus conflict or some other reason when a start condition is issued or an address and data are transmitted. The RIIC monitors the level on the SDAn line during transmission and, if the level on the line does not match the value of the bit being output, sets the value of the AL bit to 1 to indicate that the bus is occupied by another device.

The RIIC can also set the flag to indicate the detection of loss of arbitration during NACK transmission in master mode or during data transmission in slave mode.

[Setting conditions]

<When master arbitration lost detection is enabled: ICFER.MALE = 1>

- When the internal SDA output state does not match the SDAn line level at the rising edge of SCL clock except for the ACK period during data (including slave address) transmission in master transmit mode (when the SDAn line is driven low while the internal SDA output is 1 (the SDAn pin is in the high-impedance state))
- When a start condition is detected while the ST bit in ICCR2 is 1 (start condition issuance request) or the internal SDA output state does not match the SDAn line level
- When the ST bit in ICCR2 is set to 1 (start condition issuance request) with the BBSY flag in ICCR2 set to 1.

<When NACK arbitration lost detection is enabled: ICFER.MALE = 1>

- When the internal SDA output state does not match the SDAn line level at the rising edge of SCL clock in the ACK period during NACK transmission in receive mode

<When slave arbitration lost detection is enabled>

- When the internal SDA output state does not match the SDAn line level at the rising edge of SCL clock except for the ACK period during data transmission in slave transmit mode

[Clearing conditions]

- When 0 is written to the AL bit after reading AL = 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

**Table 22.5 Relationship between Arbitration Lost Generation Sources and Arbitration Lost Enable Functions**

| ICFER | | | ICSR2 | | |
|---|---|---|---|---|---|
| MALE | NALE | SALE | AL | Error | Arbitration Lost Generation Source |
| 1 | * | * | 1 | Start condition issuance error | When internal SDA output state does not match SDAn line level when a start condition is detected while the ST bit in ICCR2 is 1 |
| | | | | | When ST in ICCR2 is set to 1 with BBSY in ICCR2 set to 1 |
| | | | 1 | Transmit data mismatch | When transmit data (including slave address) does not match the bus state in master transmit mode |
| * | 1 | * | 1 | NACK transmission mismatch | When ACK is detected during transmission of NACK in master receive mode or slave receive mode |
| * | * | 1 | 1 | Transmit data mismatch | When transmit data does not match the bus state in slave transmit mode |

[Legend] *: Don't care



Downloaded from Arrow.com.

### START Flag (Start Condition Detection)

[Setting condition]

- When a start condition (or a restart condition) is detected

[Clearing conditions]

- When 0 is written to the START bit after reading START = 1
- When a stop condition is detected
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

### STOP Flag (Stop Condition Detection)

[Setting condition]

- When a stop condition is detected

[Clearing conditions]

- When 0 is written to the STOP bit after reading STOP = 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

### NACKF Flag (NACK Detection)

[Setting condition]

- When acknowledge is not received (NACK is received) from the receive device in transmit mode with the NACKE bit in ICFER set to 1 (transfer suspension enabled)

[Clearing conditions]

- When 0 is written to the NACKF bit after reading NACKF = 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

Note: When the NACKF flag is set to 1, the RIIC suspends data transmission/reception. Writing to ICDRT in transmit mode or reading from ICDRR in receive mode with the NACKF flag set to 1 does not enable data transmit/receive operation. To restart data transmission/reception, clear the NACKF flag to 0.

### RDRF Flag (Receive Data Full)

[Setting conditions]

- When receive data has been transferred from ICDRS to ICDRR
  This flag is set to 1 at the rising edge of the eighth or ninth SCL clock cycle (selected by the RDRFS bit in ICMR3)
- When the received slave address matches after a start condition (or a restart condition) is detected with the TRS bit in ICCR2 cleared to 0

[Clearing conditions]

- When 0 is written to the RDRF bit after reading RDRF = 1
- When data is read from ICDRR
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset



Downloaded from Arrow.com.

### TEND Flag (Transmit End)

[Setting condition]

- At the rising edge of the ninth SCL clock cycle while the TDRE flag is 1

[Clearing conditions]

- When 0 is written to the TEND bit after reading TEND = 1
- When data is written to ICDRT
- When a stop condition is detected
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

### TDRE Flag (Transmit Data Empty)

[Setting conditions]

- When data has been transferred from ICDRT to ICDRS and ICDRT becomes empty
- When the TRS bit in ICCR2 is set to 1
  a. When the MST bit in ICCR2 is set to 1 after a start condition (or a restart condition) is detected
  b. When the RIIC enters transmit mode from receive mode
  c. When 1 is written to while the ICMR1.MTWP bit is 1
- When the received slave address matches while the TRS bit is 1

[Clearing conditions]

- When data is written to ICDRT
- When the TRS bit in ICCR2 is cleared to 0
  a. When a stop condition is detected
  b. When the RIIC enters receive mode from transmit mode
  c. When 0 is written to while the ICMR1.MTWP bit is 1
- When 1 is written to the IICRST bit in ICCR1 to apply an RIIC reset or an internal reset

Note: When the NACKF flag is set to 1 while the NACKE bit in ICFER is 1, the RIIC suspends data transmission/reception. Here, if the TDRE flag is 0 (next transmit data has been written), data is transferred to the ICDRS register and the ICDRT register becomes empty at the rising edge of the ninth clock cycle, but the TDRE flag is not set to 1.



Downloaded from Arrow.com.

## 22.2.11 Slave Address Register Ly (SARLy) (y = 0 to 2)

Addresses: RIIC0.SARL0 0008 830Ah, RIIC1.SARL0 0008 832Ah
RIIC0.SARL1 0008 830Ch, RIIC1.SARL1 0008 832Ch
RIIC0.SARL2 0008 830Eh, RIIC1.SARL2 0008 832Eh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | SVA[7:1] | | | | | | | SVA0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | SVA0 | 10-Bit Address LSB | The least significant bit (LSB) of a 10-bit slave address is set<br>• When the FS bit in SARUy is 0 (7-bit address format), this bit is invalid<br>• When the FS bit in SARUy is 1 (10-bit address format), this bit is the LSB of the lower 8-bit address (combined with the SVA[7:1] bits) of a 10-bit slave address | R/W |
| b7 to b1 | SVA[7:1] | 7-Bit Address/10-Bit Address Lower Bits | A slave address is set<br>• When the FS bit in SARUy is 0 (7-bit address format), these bits form a 7-bit slave address<br>• When the FS bit in SARUy is 1 (10-bit address format), these bits form the lower 8-bit address (combined with the SVA0 bit) of a 10-bit slave address | R/W |

SARLy sets slave address y (7-bit address or lower eight bits of 10-bit address).

### SVA0 Bit (10-Bit Address LSB)

When the 10-bit address format is selected (SARUy.FS = 1), this bit functions as the LSB of a 10-bit address and forms the lower eight bits of a 10-bit address in combination with the SVA[7:1] bits.

When the SARyE bit in ICSER is set to 1 (SARLy and SARUy enabled) and the SARUy.FS bit is 1, this bit is valid. While the SARUy.FS bit or SARyE bit is 0, the setting of this bit is ignored.

### SVA[7:1] Bits (7-Bit Address/10-Bit Address Lower Bits)

When the 7-bit address format is selected (SARUy.FS = 0), these bits function as a 7-bit address. When the 10-bit address format is selected (SARUy.FS = 1), these bits function as the lower eight bits of a 10-bit address in combination with the SVA0 bit.

While the SARyE bit in ICSER is 0, the setting of these bits is ignored.



Downloaded from Arrow.com.

## 22.2.12 Slave Address Register Uy (SARUy) (y = 0 to 2)

Addresses: RIIC0.SARU0 0008 830Bh, RIIC1.SARU0 0008 832Bh
RIIC0.SARU1 0008 830Dh, RIIC1.SARU1 0008 832Dh
RIIC0.SARU2 0008 830Fh, RIIC1.SARU2 0008 832Fh

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SVA[9:8] | | FS |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FS | 7-Bit/10-Bit Address Format Selection | 0: The 7-bit address format is selected<br>1: The 10-bit address format is selected | R/W |
| b2, b1 | SVA[9:8] | 10-Bit Address Upper Bits | A slave address is set<br>• When the SARUy.FS bit is 0 (7-bit address format), these bits are invalid<br>• When the SARUy.FS bit is 1 (10-bit address format), these bits form the upper two bits of a 10-bit slave address | R/W |
| b7 to b3 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

SARUy selects 7-bit address format or 10-bit address format and sets the upper bits of a 10-bit slave address.

### FS Bit (7-Bit/10-Bit Address Format Selection)

This bit is used to select 7-bit address or 10-bit address for slave address m (in SARLy and SARUy).

When the SARyE bit in ICSER is set to 1 (SARLy and SARUy enabled) and the SARUy.FS bit is 0, the 7-bit address format is selected for slave address m, the SVA[7:1] setting in SARLy is valid, and the settings of the SVA[9:8] bits and the SVA0 bit in SARLy are ignored.

When the SARyE bit in ICSER is set to 1 (SARLy and SARUy enabled) and the SARUy.FS bit is 1, the 10-bit address format is selected for slave address m and the settings of the SVA[9:8] bits and SARLy are valid.

While the SARyE bit in ICSER is 0 (SARLy and SARUy disabled), the setting of the SARUy.FS bit is invalid.

### SVA[9:8] Bits (10-Bit Address Upper Bits)

When the 10-bit address format is selected (FS = 1), these bits function as the upper two bits of a 10-bit address.

When the SARyE bit in ICSER is set to 1 (SARLy and SARUy enabled) and the SARUy.FS bit is 1, these bits are valid. While the SARUy.FS bit or SARyE bit is 0, the setting of these bits is ignored.



Downloaded from Arrow.com.

## 22.2.13 I²C Bus Bit Rate Low-Level Register (ICBRL)

Addresses: RIIC0.ICBRH 0008 8310h, RIIC1.ICBRH 0008 8330h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | BRL[4:0] | | | | |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b4 to b0 | BRL[4:0] | Bit Rate Low-Level Period | Low-level period of SCL clock | R/W |
| b7 to b5 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |

ICBRL is a 5-bit register to set the low-level period of SCL clock. It also works to generate the data setup time for automatic SCL low-hold operation (see section 22.8, Function to Automatically Hold SCLn Clock Low); when the RIIC is used only in slave mode, this register needs to be set to a value longer than the data setup time*.

ICBRL counts the low-level period with the internal reference clock source (IICϕ) specified by the CKS[2:0] bits in ICMR1.

Note: Data setup time (tSU: DAT)
250 ns (up to 100 kbps: standard mode [Sm])
100 ns (up to 400 kbps: fast mode [fm])
50 ns (up to 1 Mbps: fast mode plus [fm+])



Downloaded from Arrow.com.

## 22.2.14 I²C Bus Bit Rate High-Level Register (ICBRH)

Addresses: RIIC0.ICBRL 0008 8311h, RIIC1.ICBRL 0008 8331h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | BRH[4:0] | | | | |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b4 to b0 | BRH[4:0] | Bit Rate High-Level Period | High-level period of SCL clock | R/W |
| b7 to b5 | — | Reserved | These bits are always read as 1. The write value should always be 1. | R/W |

ICBRH is a 5-bit register to set the high-level period of SCL clock. ICBRH is valid in master mode. If the RIIC is used only in slave mode, this register need not to set the high-level period.

ICBRH counts the high-level period with the internal reference clock source (IICϕ) specified by the CKS[2:0] bits in ICMR1.

The I²C transfer rate and the SCL clock duty ratio are calculated using the following expression.

Transfer rate = 1 / { [(ICBRH + 1) + (ICBRL + 1)] / IICϕ*1 + SCLn line rising time [tr] + SCLn line falling time [tf]}

Duty cycle = { SCLn line rising time [tr]*2 + (ICBRH + 1) / IICϕ} / {SCLn line falling time [tf]*2 + (ICBRL + 1) / IICϕ}

Notes: 1. IICϕ = PCLK × 106 × Division ratio

2. The SCLn line rising time [tr] and SCLn line falling time [tf] depend on the total bus line capacitance [Cb] and the pull-up resistor [Rp]. For details, see the I²C bus standard from NXP Semiconductors.



Downloaded from Arrow.com.

Table 22.6 shows examples of ICBRH/ICBRL settings.

**Table 22.6 Examples of ICBRH/ICBRL Settings for Transfer Rate**

| Transfer Rate (kbps) | Operating Frequency PCLK (MHz) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | | | 10 | | | 12.5 | | |
| | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL |
| 10 | 100b | 22 (F6h) | 25 (F9h) | 101b | 13 (EDh) | 15 (EFh) | 101b | 16 (F0h) | 20 (F4h) |
| 50 | 010b | 16 (F0h) | 19 (F3h) | 010b | 21 (F5h) | 24 (F8h) | 011b | 12 (ECh) | 15 (EFh) |
| 100 | 001b | 15 (EFh) | 18 (F2h) | 001b | 19 (F3h) | 23 (F7h) | 001b | 24 (F8h) | 29 (FDh) |
| 400 | 000b | 4 (E4h) | 10 (EAh) | 000b | 5 (E5h) | 12 (ECh) | 000b | 7 (E7h) | 16 (F0h) |
| 1000 | 000b | 2 (E2h) | 3 (E3h) | 000b | 2 (E2h) | 4 (E4h) | 000b | 3 (E3h) | 6 (E6h) |

| Transfer Rate (kbps) | Operating Frequency PCLK (MHz) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 16 | | | 20 | | | 25 | | |
| | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL |
| 10 | 101b | 22 (F6h) | 25 (F9h) | 110b | 13 (EDh) | 15 (EFh) | 110b | 16 (F0h) | 20 (F4h) |
| 50 | 011b | 16 (F0h) | 19 (F3h) | 011b | 21 (F5h) | 24 (F8h) | 100b | 12 (ECh) | 15 (EFh) |
| 100 | 010b | 15 (EFh) | 18 (F2h) | 010b | 19 (F3h) | 23 (F7h) | 010b | 24 (F8h) | 29 (FDh) |
| 400 | 000b | 9 (E9h) | 20 (F4h) | 000b | 11 (EBh) | 25 (F9h) | 001b | 7 (E7h) | 16 (F0h) |
| 1000 | 000b | 4 (E4h) | 7 (E7h) | 000b | 5 (E5h) | 9 (E9h) | 000b | 6 (E6h) | 12 (ECh) |

| Transfer Rate (kbps) | Operating Frequency PCLK (MHz) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 30 | | | 33 | | | 50 | | |
| | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL | CKS [2:0] | ICBRH | ICBRL |
| 10 | 110b | 20 (F4h) | 24 (F8h) | 110b | 22 (F6h) | 26 (FAh) | 111b | 16 (F0h) | 20 (F4h) |
| 50 | 100b | 15 (EFh) | 18 (F2h) | 100b | 17 (F1h) | 20 (F4h) | 100b | 26 (FAh) | 31 (FFh) |
| 100 | 010b | 2 (E2h) | 3 (E3h) | 011b | 16 (F0h) | 19 (F3h) | 011b | 24 (F8h) | 29 (FDh) |
| 400 | 001b | 8 (E8h) | 19 (F3h) | 001b | 9 (E9h) | 21 (F5h) | 010b | 7 (E7h) | 16 (F0h) |
| 1000 | 000b | 7 (E7h) | 14 (EEh) | 000b | 8 (E8h) | 16 (F0h) | 000b | 12 (ECh) | 24 (F8h) |

Note: ICBRH/ICBRL settings in these tables are calculated using the following values:

SCLn line rising time (tr): 100 kbps or less, [Sm]: 1000 ns, 400 kbps or less, [Fm]: 300 ns, 1 Mbps or less, [Fm+]: 120 ns
SCLn line falling time (tf): 400 kbps or less, [Sm/Fm]: 300 ns, 1 Mbps or less, [Fm+]: 120 ns
For the specified values of SCLn line rising time (tr) and SCLn line falling time (tf), see the I²C bus standard from NXP Semiconductors.



Downloaded from Arrow.com.

### 22.2.15 I²C Bus Transmit Data Register (ICDRT)

Addresses: RIIC0.ICDRT 0008 8312h, RIIC1.ICDRT 0008 8332h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

ICDRT is an 8-bit readable/writable register that stores transmit data. When ICDRT detects a space in the I²C bus shift register (ICDRS), it transfers the transmit data that has been written to ICDRT to ICDRS and starts transmitting data in transmit mode.

The double-buffer structure of ICDRT and ICDRS allows continuous transmit operation if the next transmit data has been written to ICDRT while the ICDRS data is being transmitted.

ICDRT can always be read and written. Write transmit data to ICDRT once when a transmit data empty interrupt (ICTXI) request is generated.

### 22.2.16 I²C Bus Receive Data Register (ICDRR)

Addresses: RIIC0.ICDRR 0008 8313h, RIIC1.ICDRR 0008 8333h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ICDRR is an 8-bit read-only register that stores receive data. When one byte of data has been received, the received data is transferred from the I²C bus shift register (ICDRS) to ICDRR to enable the next data to be received.

The double-buffer structure of ICDRS and ICDRR allows continuous receive operation if the received data has been read from ICDRR while ICDRS is receiving data.

ICDRR cannot be written. Read data from ICDRR once when a receive data full interrupt (ICRXI) request is generated.

If ICDRR receives the next receive data before the current data is read from ICDRR (while the RDRF flag in ICSR2 is 1), the RIIC automatically holds the SCL clock low one cycle before the RDRF flag is set to 1 next.

### 22.2.17 I²C Bus Shift Register (ICDRS)

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| Value after reset: | – | – | – | – | – | – | – | – |

ICDRS is an 8-bit shift register to transmit and receive data.

During transmission, transmit data is transferred from ICDRT to ICDRS and is sent from the SDAn pin. During reception, data is transferred from ICDRS to ICDRR after one byte of data has been received.

ICDRS cannot be accessed directly.

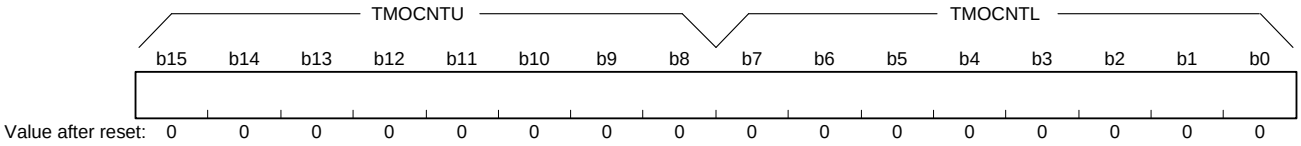


Downloaded from Arrow.com.

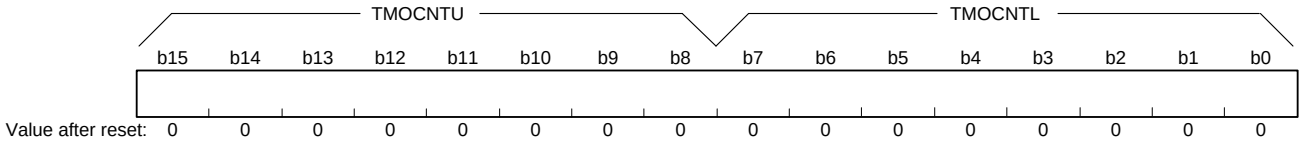
## 22.2.18 Internal Counter for Timeout (TMOCNT)

Addresses: RIIC0.TMOCNTL 0008 830Ah, RIIC0.TMOCNTU 0008 830Bh*1
RIIC1.TMOCNTL 0008 832Ah, RIIC1.TMOCNTU 0008 832Bh*1

- ICMR2.TMOS = 0 (when long mode is selected)




- ICMR2.TMOS = 1 (when short mode is selected)




Note 1. Care is required because these are the same registers as SARL0 and SARU0.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b15 to b8 | TMOCNTU | Internal Counter for Timeout | Higher-order bits of the internal counter for timeout*1 | W*2 |
| b7 to b0 | TMOCNTL | | Lower-order bits of the internal counter for timeout | W*2 |

Notes: 1. Bits 15 to 12 become reserved bits when TMOS = 1 (short mode). Although they are still writable, values written have no effect.

2. The value of the internal counter for timeout is not readable. If reading is attempted, the value read will be FFFFh.

Timeout internal counter (TMOCNTL/TMOCNTU) is initialized (0000h) after a reset, while ICCR1.IICRST = 1 or ICFER.TMOE = 1 and PCLK/1 is selected with ICMR1.CKS[2:0] = 000b setting, and when counter clear conditions specified by TMOH/TMOL of ICMR2 (SCL rising/falling edge detection) are satisfied.



Downloaded from Arrow.com.

## 22.3 Operation

### 22.3.1 Communication Data Format

The I²C bus format consists of 8-bit data and 1-bit acknowledge. The frame following a start condition or restart condition is an address frame used to specify a slave device with which the master device communicates. The specified slave is valid until a new slave is specified or a stop condition is issued.

Figure 22.3 shows the I²C bus format, and figure 22.4 shows the I²C bus timing.

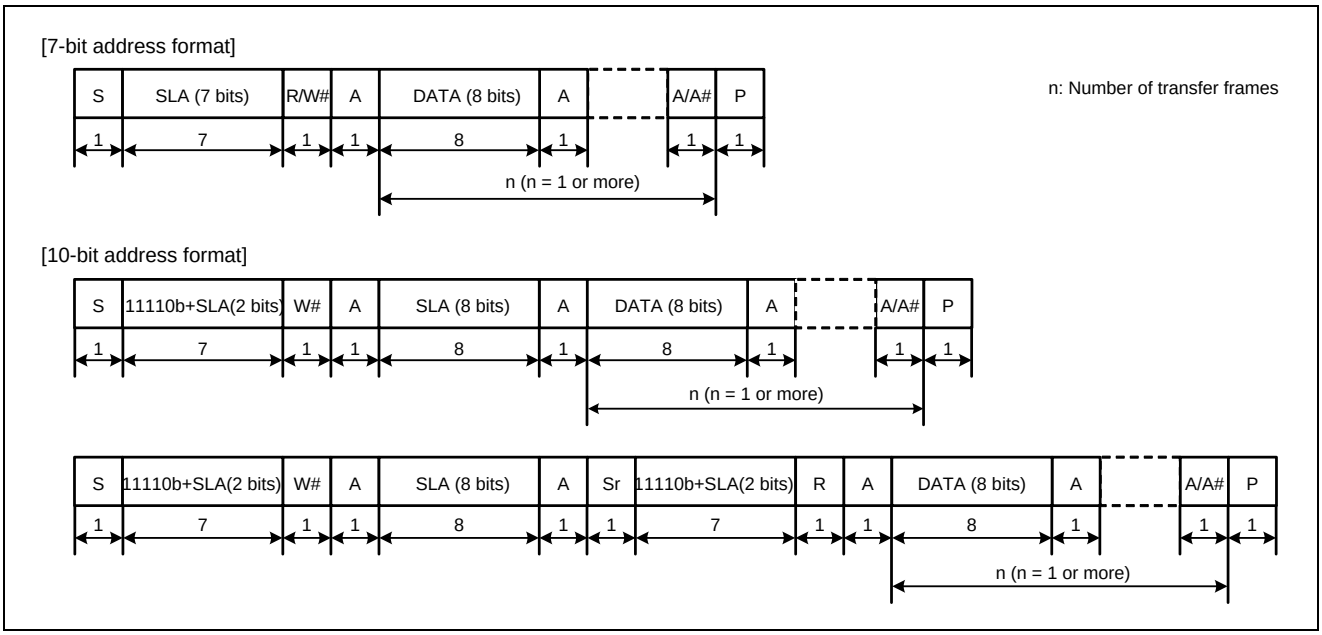



**Figure 22.3 I²C Bus Format**

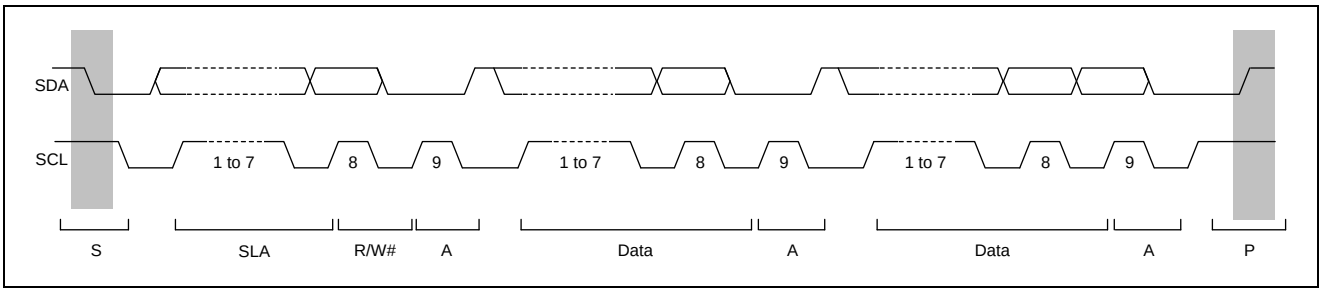



**Figure 22.4 I²C Bus Timing (SLA = 7 Bits)**

**[Legend]**

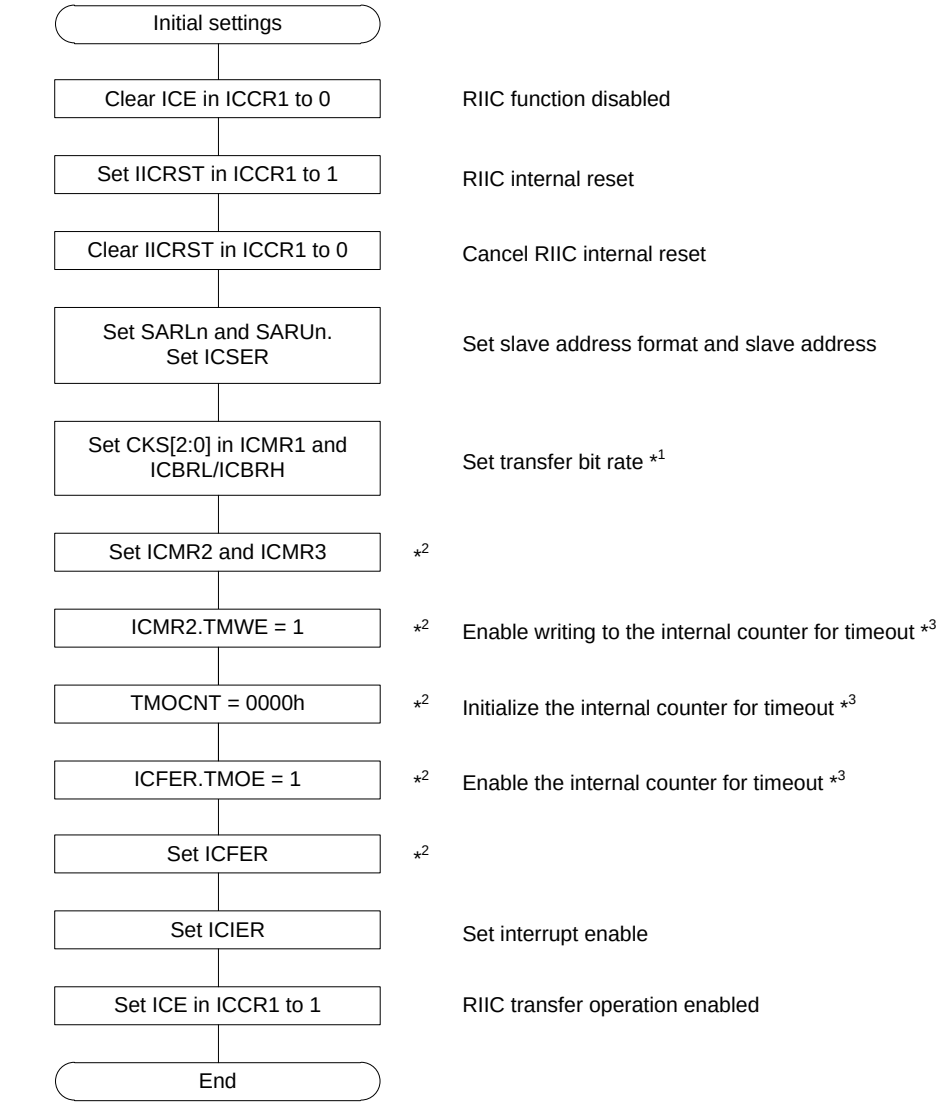
| | |
|---|---|
| S: | Start condition. The master device drives the SDAn line low from high level while the SCLn line is at a high level. |
| SLA: | Slave address, by which the master device selects a slave device. |
| R/W#: | Indicates the direction of data transfer: from the slave device to the master device when R/W is 1, or from the master device to the slave device when R/W is 0. |
| A: | Acknowledge. The receive device drives the SDAn line low. (In master transmit mode, the slave device returns acknowledge. In master receive mode, the master device returns acknowledge.) |
| Sr: | Restart condition. The master device drives the SDAn line low from the high level after the setup time has elapsed with the SCLn line at the high level. |
| DATA: | Transmitted or received data |
| P: | Stop condition. The master device drives the SDAn line high from low level while the SCLn line is at a high level. |



Downloaded from Arrow.com.

## 22.3.2 Initial Settings

Before starting data transmission and reception, initialize the RIIC according to the procedure in figure 22.5.




[Legend] n = 0 to 2

Notes: 1. When the RIIC is used only in slave mode, set the ICBRL register to a value longer than the data setup time.
2. Set these registers as necessary.
3. This is the case when the timeout function is in use (this processing is not required if the function is not in use).

**Figure 22.5 Example of RIIC Initialization Flow**



Downloaded from Arrow.com.

### 22.3.3 Master Transmitter Operation

In master transmitter operation, the RIIC outputs the SCL (clock) and transmitted data signals as the master device, and the slave device returns acknowledgements. Figure 22.6 shows an example of usage of master transmission and figures 22.7 to 22.9 show the timing of operations in master transmission.

The following describes the procedure and operations for master transmission.

1. Set the IICRST bit in ICCR1 to 1 (internal reset) and then clear the IICRST bit to 0 (canceling reset) with the ICE bit in ICCR1 cleared to 0 (disabling the interface). This initializes the internal state and the various flags of ICSR1. After that, set registers SARLy, SARUy, ICSER, ICMR1, ICBRH, and ICBRL (y = 0 to 2), and set the other registers as necessary (for initial settings of the RIIC, see figure 22.5). When the necessary register settings have been completed, set the ICE bit to 1 (to enable transfer). This step is not necessary if initialization of the RIIC has already been completed.

2. Read the BBSY flag in ICCR2 to check that the bus is open, and then set the ST bit in ICCR2 to 1 (start condition issuance request). Upon receiving the request, the RIIC issues a start condition. At the same time, the BBSY flag and the START flag in ICSR2 are automatically set to 1 and the ST bit is automatically cleared to 0. At this time, if the start condition is detected and the internal levels for the SDA output state and the levels on the SDAn line have matched while the ST bit is 1, the RIIC recognizes that issuing of the start condition as requested by the ST bit has been successfully completed, and the MST and TRS bits in ICCR2 are automatically set to 1, placing the RIIC in master transmitter mode. The TDRE flag in ICSR2 is also automatically set to 1 in response to setting of the TRS bit to 1.

3. Check that the TDRE flag in ICSR2 is 1, and then write the value for transmission (the slave address and the R/W# bit) to ICDRT. Once the data for transmission are written to ICDRT, the TDRE flag is automatically cleared to 0, the data are transferred from ICDRT to ICDRS, and the TDRE flag is again set to 1. After the byte containing the slave address and R/W# bit has been transmitted, the value of the TRS bit is automatically updated to select master transmitter or master receiver mode in accord with the value of the transmitted R/W# bit. If the value of the R/W# bit was 0, the RIIC continues in master transmitter mode.
   Since the ICSR2.NACKF flag being 1 at this time indicates that no slave device recognized the address or there was an error in communications, write 1 to the ICCR2.SP bit to issue a stop condition.
   For data transmission with an address in the 10-bit format, start by writing 1111 0b, the two higher-order bits of the slave address, and W to ICDRT as the first address transmission. Then, as the second address transmission, write the eight lower-order bits of the slave address to ICDRT.

4. After confirming that the TDRE flag in ICSR2 is 1, write the data for transmission to the ICDRT register. The RIIC automatically holds the SCLn line low until the data for transmission are ready or a stop condition is issued.

5. After all bytes of data for transmission have been written to the ICDRT register, wait until the value of the TEND flag in ICSR2 returns to 1, and then set the SP bit in ICCR2 to 1 (stop condition issuance request). Upon receiving a stop condition issuance request, the RIIC issues the stop condition.

6. Upon detecting the stop condition, the RIIC automatically clears the MST and TRS bits in ICCR2 to 00b and enters slave receiver mode. Furthermore, it automatically clears the TDRE and TEND flags to 0, and sets the STOP flag in ICSR2 to 1.

7. After checking that the ICSR2.STOP flag is 1, clear the ICSR2.NACKF and STOP flags to 0 for the next transfer operation.



Downloaded from Arrow.com.

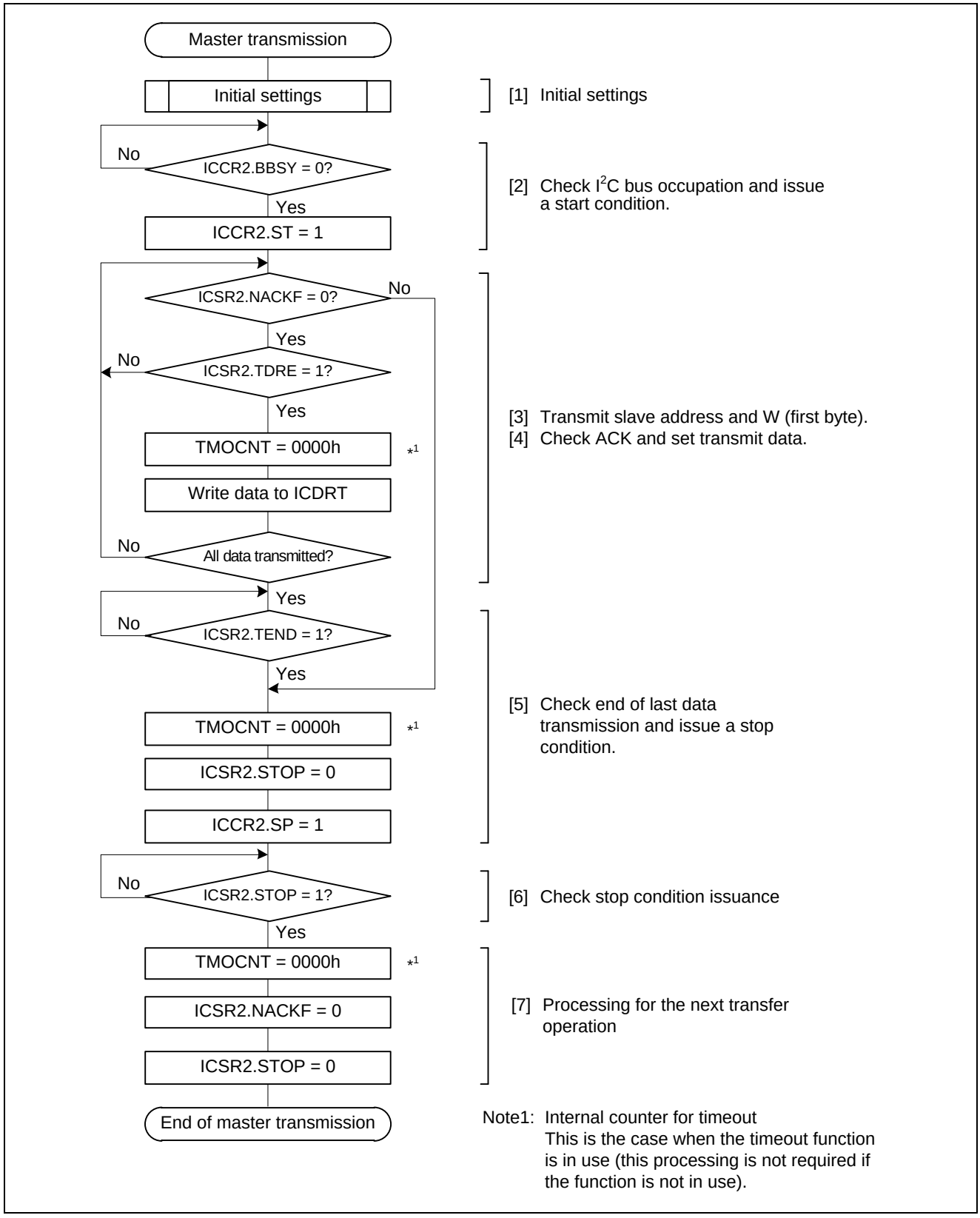



**Figure 22.6 Example of Master Transmission Flowchart**



Downloaded from Arrow.com.

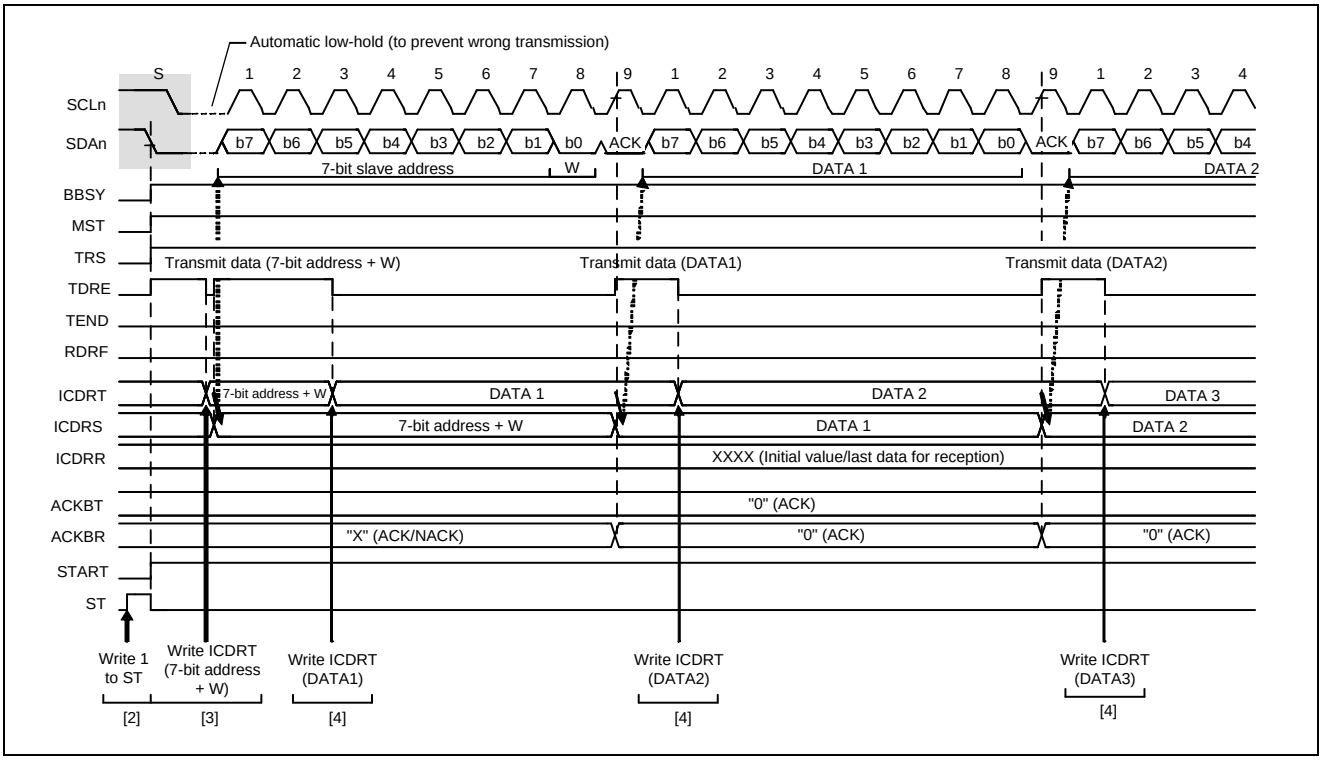



**Figure 22.7 Master Transmit Operation Timing (1) (7-Bit Address Format)**

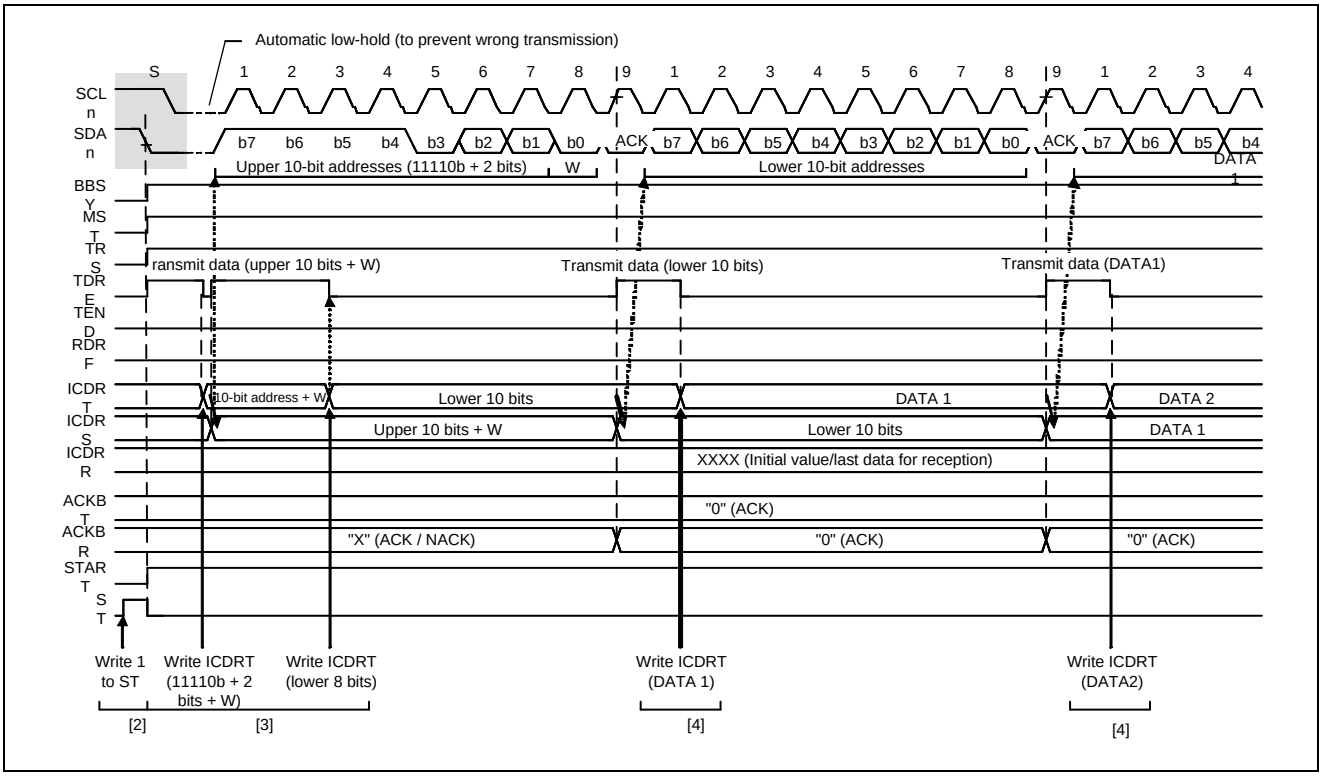



**Figure 22.8 Master Transmit Operation Timing (2) (10-Bit Address Format)**



Downloaded from Arrow.com.

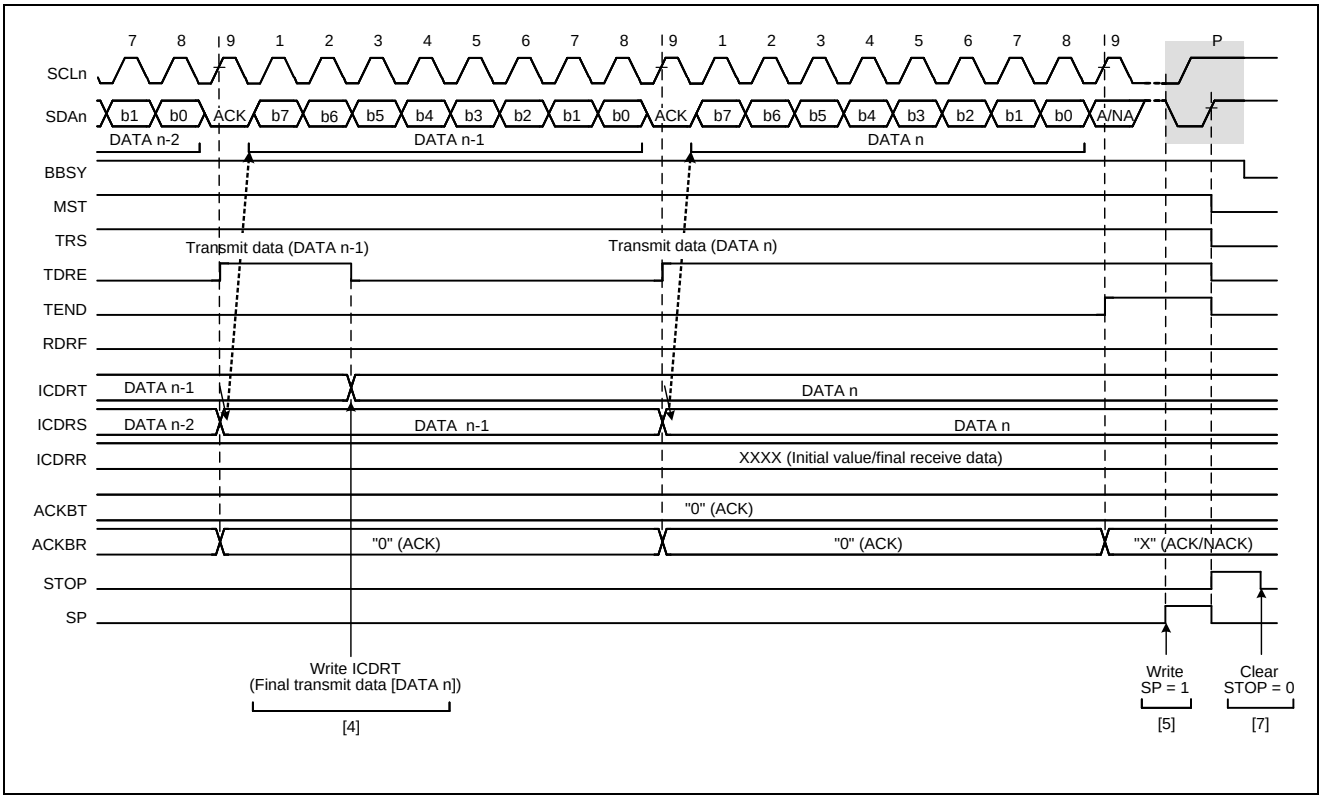



**Figure 22.9 Master Transmit Operation Timing (3)**



Downloaded from Arrow.com.

### 22.3.4 Master Receiver Operation

In master receiver operation, the RIIC as a master device outputs the SCL (clock) signal, receives data from the slave device, and returns acknowledgements. Since the RIIC must start by sending a slave address to the corresponding slave device, this part of the procedure is performed in master transmitter mode, but the subsequent steps are in master receiver mode.

Figure 22.10 shows an example of usage of master reception and figures 22.11 and 22.13 show the timing of operations in master reception.

The following describes the procedure and operations for master reception.

1. Set the IICRST bit in ICCR1 to 1 (internal reset) and then clear the IICRST bit to 0 (canceling reset) with the ICE bit in ICCR1 cleared to 0 (disabling the interface). This initializes the internal state and the various flags of ICSR1. After that, set registers SARLy, SARUy, ICSER, ICMR1, ICBRH, and ICBRL (y = 0 to 2), and set the other registers as necessary (for initial settings of the RIIC, see figure 22.5). When the necessary register settings have been completed, set the ICE bit to 1 (to enable transfer). This step is not necessary if initialization of the RIIC has already been completed.
2. Read the BBSY flag in ICCR2 to check that the bus is open, and then set the ST bit in ICCR2 to 1 (start condition issuance request). Upon receiving the request, the RIIC issues a start condition. When the RIIC detects the start condition, the BBSY flag and the START flag in ICSR2 are automatically set to 1 and the ST bit is automatically cleared to 0. At this time, if the start condition is detected and the levels for the SDA output and the levels on the SDAn line have matched while the ST bit is 1, the RIIC recognizes that issuing of the start condition as requested by the ST bit has been successfully completed, and the MST and TRS bits in ICCR2 are automatically set to 1, placing the RIIC in master transmitter mode. The TDRE flag in ICSR2 is also automatically set to 1 in response to setting of the TRS bit to 1.
3. Check that the TDRE flag in ICSR2 is 1, and then write the value for transmission (the first byte indicates the slave address and value of the R/W# bit) to ICDRT. Once the data for transmission are written to ICDRT, the TDRE flag is automatically cleared to 0, the data are transferred from ICDRT to ICDRS, and the TDRE flag is again set to 1. Once the byte containing the slave address and R/W# bit has been transmitted, the value of the ICCR2.TRS bit is automatically updated to select master transmitter or master receiver mode in accord with the value of the transmitted R/W# bit. If the value of the R/W# bit was 1, the TRS bit is cleared to 0 on the rising edge of the ninth cycle of SCLn (the clock signal), placing the RIIC in master receiver mode. At this time, the TDRE flag is automatically cleared to 0 and the ICSR2.RDRF flag is automatically set to 1.
   Since the ICSR2.NACKF flag being 1 at this time indicates that no slave device recognized the address or there was an error in communications, write 1 to the ICCR2.SP bit to issue a stop condition.
   For master reception from a device with a 10-bit address, start by using master transmission to issue the 10-bit address, and then issue a restart condition. After that, transmitting 1111 0b, the two higher-order bits of the slave address, and the R bit places the RIIC in master receiver mode.
4. Dummy read ICDRR after confirming that the RDRF flag in ICSR2 is 1; this makes the RIIC start output of the SCL (clock) signal and start data reception.



Downloaded from Arrow.com.

5. After one byte of data has been received, the RDRF flag in ICSR2 is set to 1 on the rising edge of the eighth or ninth cycle of SCL clock (the clock signal) as selected by the RDRFS bit in ICMR3. Reading out ICDRR at this time will produce the received data, and the RDRF flag is automatically cleared to 0 at the same time. Furthermore, the value of the acknowledgement field received during the ninth cycle of SCL clock is returned as the value set in the ACKBT bit of ICMR3.Furthermore, if the next byte to be received is the next to last byte, set the ICMR3.WAIT bit to 1 (for wait insertion) before reading the ICDRR (containing the second byte from last). As well as enabling NACK output even in the case of delays in processing to set the ICMR3.ACKBT bit to 1 (NACK) in step (6), due to other interrupts etc., this fixes the SCLn line to the low level on the rising edge of the ninth clock cycle in reception of the last byte, so the state is such that issuing a stop condition is possible.
6. When the ICMR3.RDRFS bit is 0 and the slave device must be notified that it is to end transfer for data reception after transfer of the next (final) byte, set the ACKBT bit in ICMR3 to 1 (NACK).
7. After reading out the byte before last from the ICDRR register, if the value of the ICSR2.RDRF flag is confirmed to be 1, write 1 to the SP bit in ICCR2 (stop condition issuance request) and then read the last byte from ICDRR. When ICDRR is read, the RIIC is released from the wait state and issues the stop condition after low-level output in the ninth clock cycle is completed or the SCL line is released from the low-hold state.
8. Upon detecting the stop condition, the RIIC automatically clears the MST and TRS bits in ICCR2 to "00b" and enters slave receiver mode. Furthermore, detection of the stop condition leads to setting of the STOP flag in ICSR2 to 1.
9. After checking that the ICSR2.STOP flag is 1, clear the ICSR2.NACKF and STOP flags to 0 for the next transfer operation.



Downloaded from Arrow.com.

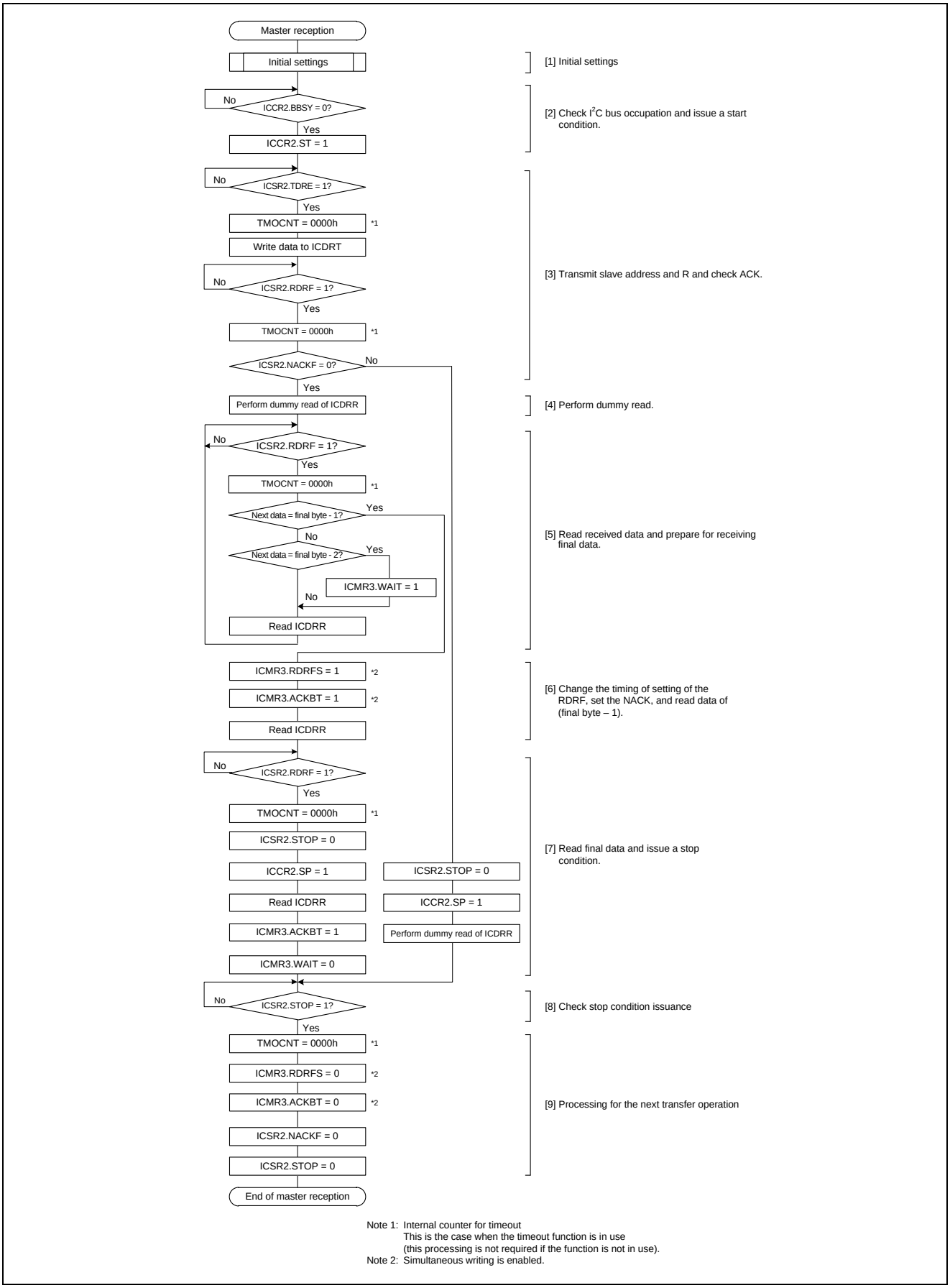



**Figure 22.10 Example of Master Reception Flowchart (7-Bit Address Format)**



Downloaded from Arrow.com.

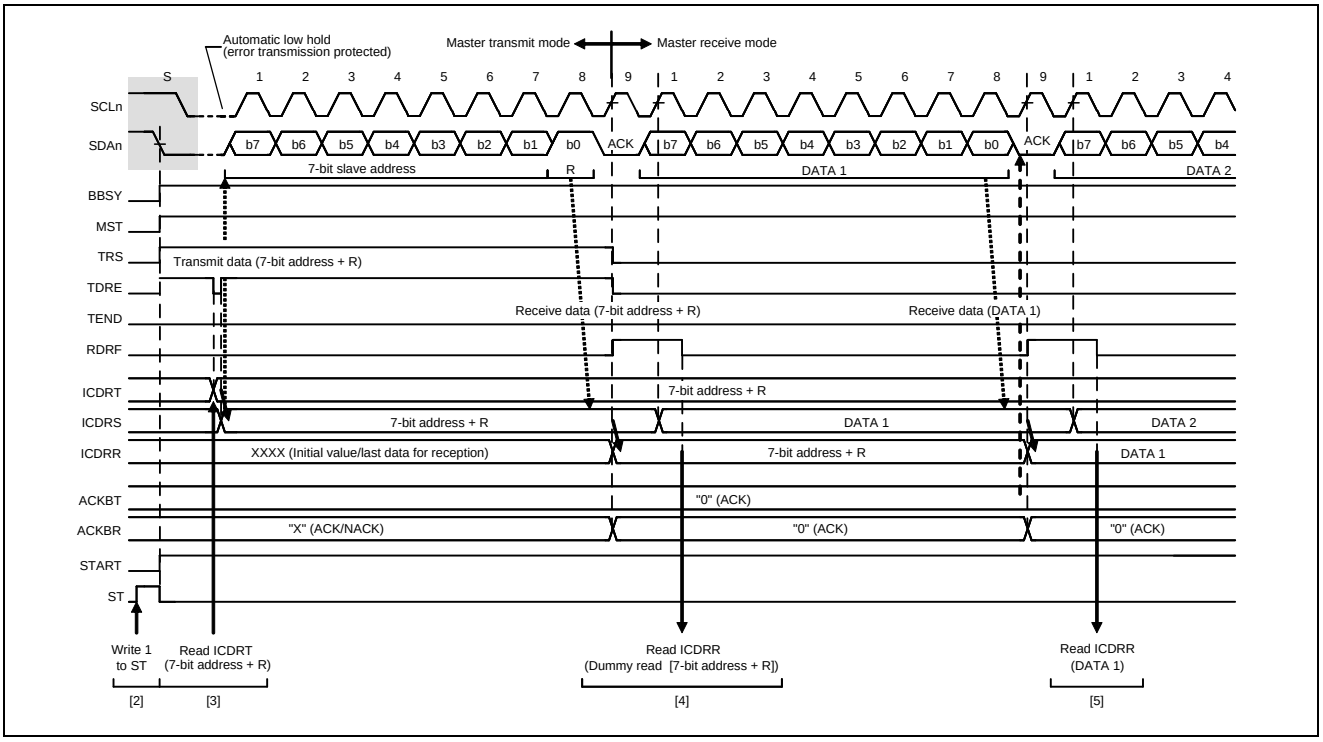



**Figure 22.11 Master Receive Operation Timing (1) (7-Bit Address Format, when RDRFS=0)**

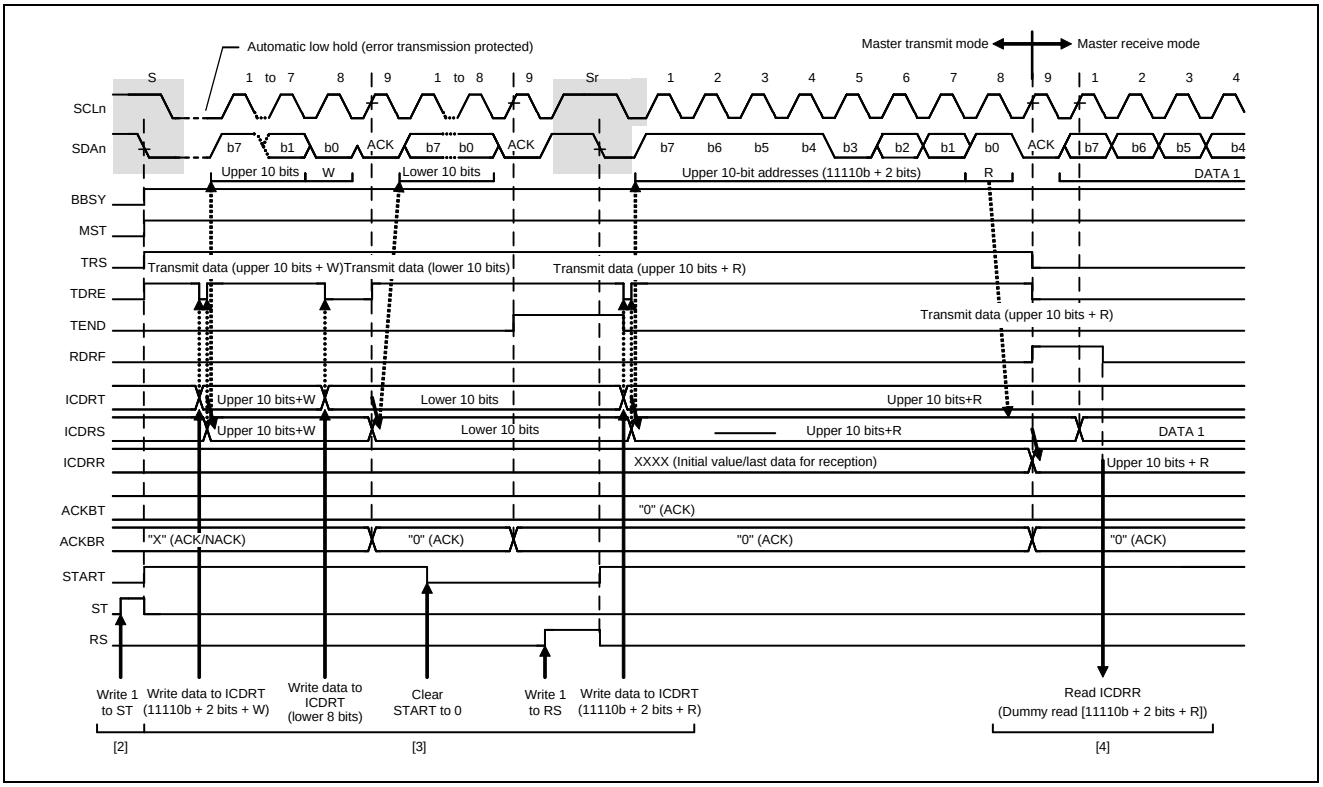



**Figure 22.12 Master Receive Operation Timing (2) (10-Bit Address Format, when RDRFS=0)**



Downloaded from Arrow.com.

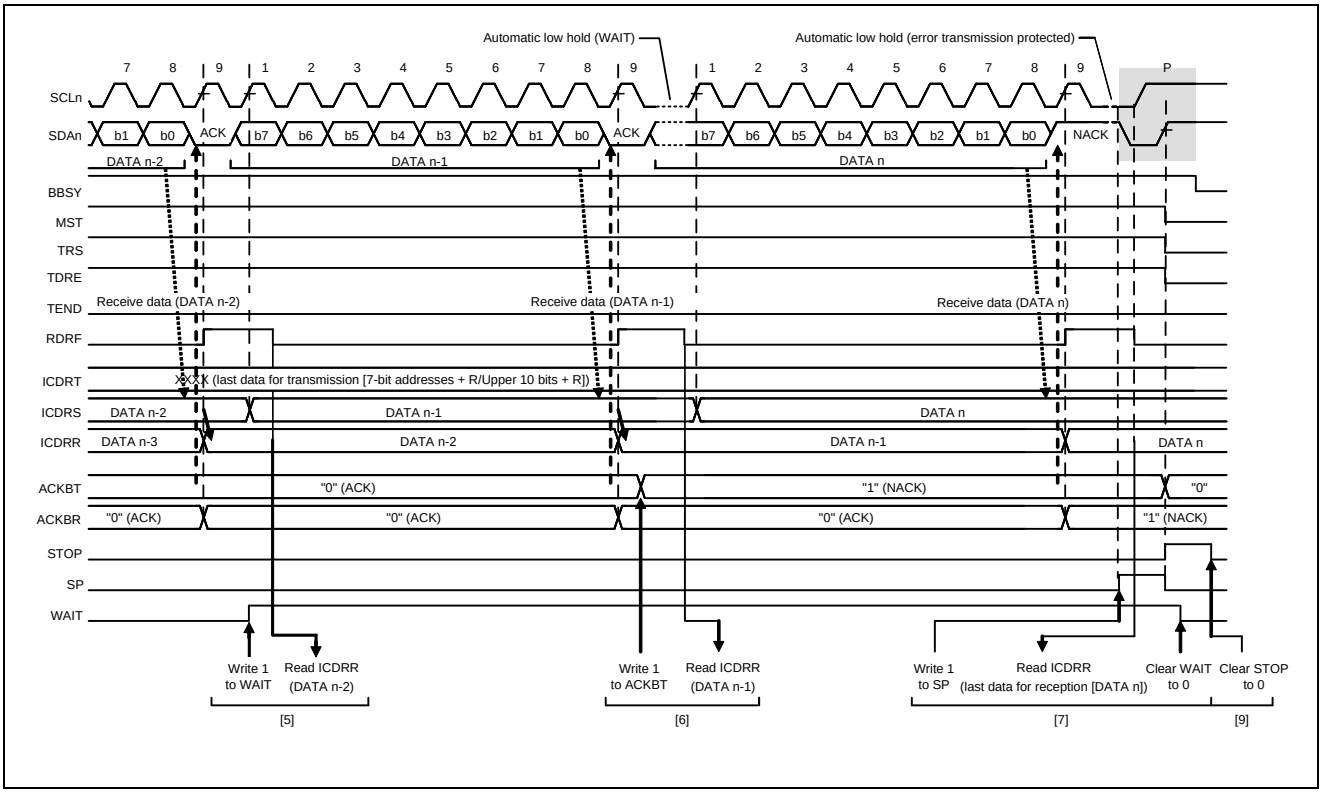



**Figure 22.13 Master Receive Operation Timing (3) (when RDRFS=0)**



Downloaded from Arrow.com.

### 22.3.5 Slave Transmitter Operation

In slave transmitter operation, the master device outputs the SCL (clock) signal, the RIIC transmits data as a slave device, and the master device returns acknowledgements.

Figure 22.14 shows an example of usage of slave transmission and figures 22.15 and 22.16 show the timing of operations in slave transmission.

The following describes the procedure and operations for slave transmission.

1. Follow the procedure in figure 22.5 to make initial settings for the RIIC. This step is not necessary if initialization of the RIIC has already been completed. After initial settings, the RIIC will stay in the standby state until it receives a slave address that it matches.
2. After receiving a matching slave address, the RIIC sets one of the corresponding bits ICSR1.HOA, GCA, and AASy (y = 0 to 2) to 1 on the rising edge of the ninth cycle of SCL clock (the clock signal) and outputs the value set in the ICMR3.ACKBT bit to the acknowledge bit on the ninth cycle of SCL clock. If the value of the R/W# bit that was also received at this time is 1, the RIIC automatically places itself in slave transmitter mode by setting both the TRS bit and the TDRE flag in ICSR2 to 1.
3. After the ICSR2.TEND flag is confirmed to be 1, write the data for transmission to the ICDRT register. At this time, if the RIIC receives no acknowledge from the master device (receives an NACK signal) while the ICFER.NACKE bit is 1, the RIIC suspends transfer of the next data.
4. Wait unit the ICSR2.TEND flag is set to 1 while the ICSR2.TDRE flag is 1, after the ICSR2.NACKF flag is set to 1or the last byte for transmission is written to the ICDRT register. When the ICSR2.NACKF flag or the TEND flag is 1, the RIIC drives the SCLn line low on the ninth falling edge of SCL clock.
5. When the ICSR2.NACKF flag or the ICSR2.TEND flag is 1, dummy read ICDRR to complete the processing. This releases the SCLn line.
6. Upon detecting the stop condition, the RIIC automatically clears bits ICSR1.HOA, GCA, and AASy (y = 0 to 2), flags ICSR2.TDRE and TEND, and the ICCR2.TRS bit to 0, and enters slave receiver mode.
7. After checking that the ICSR2.STOP flag is 1, clear the ICSR2.NACKF and STOP flags to 0 for the next transfer operation.



Downloaded from Arrow.com.

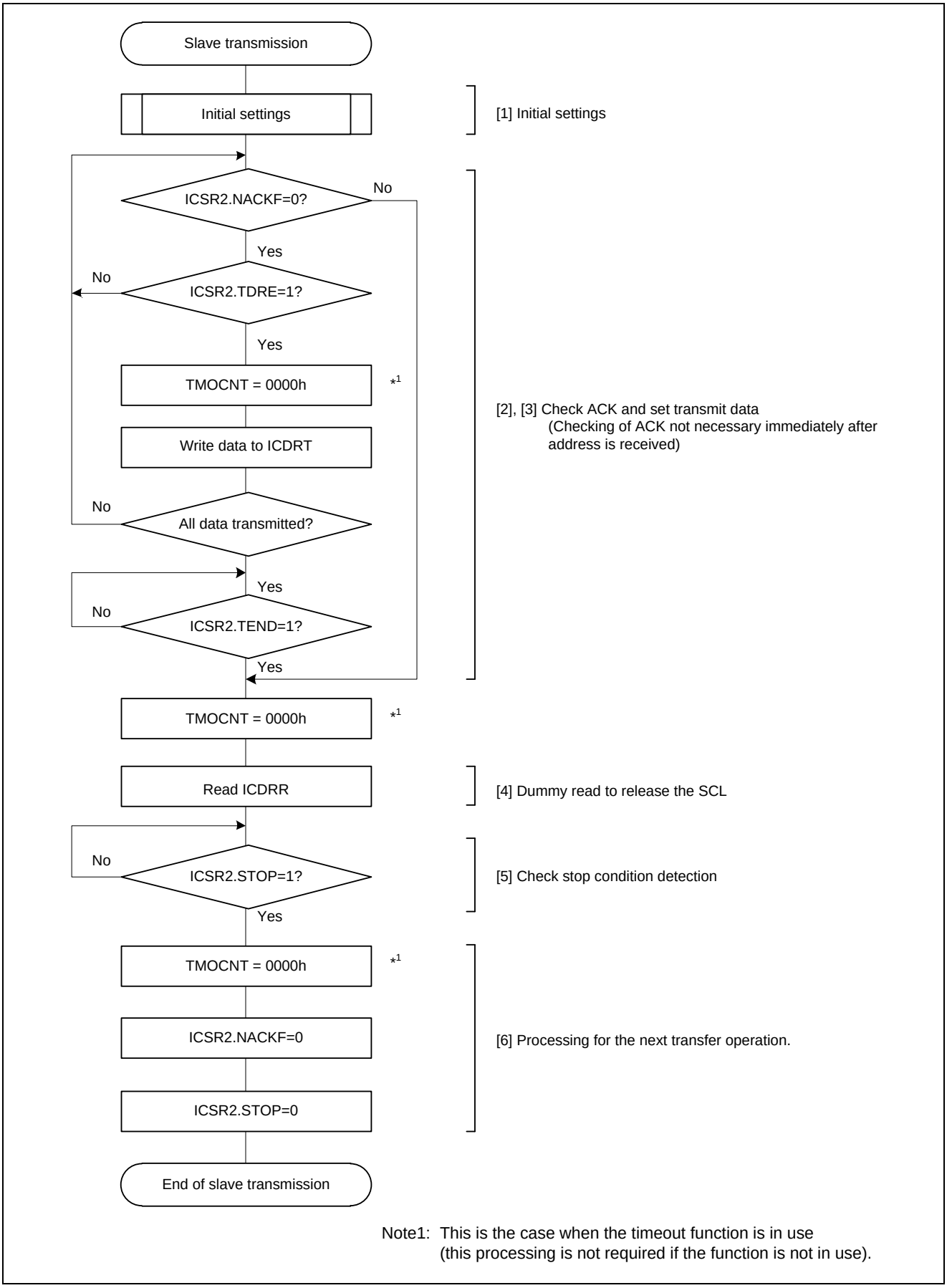



**Figure 22.14 Example of Slave Transmission Flowchart**



Downloaded from Arrow.com.

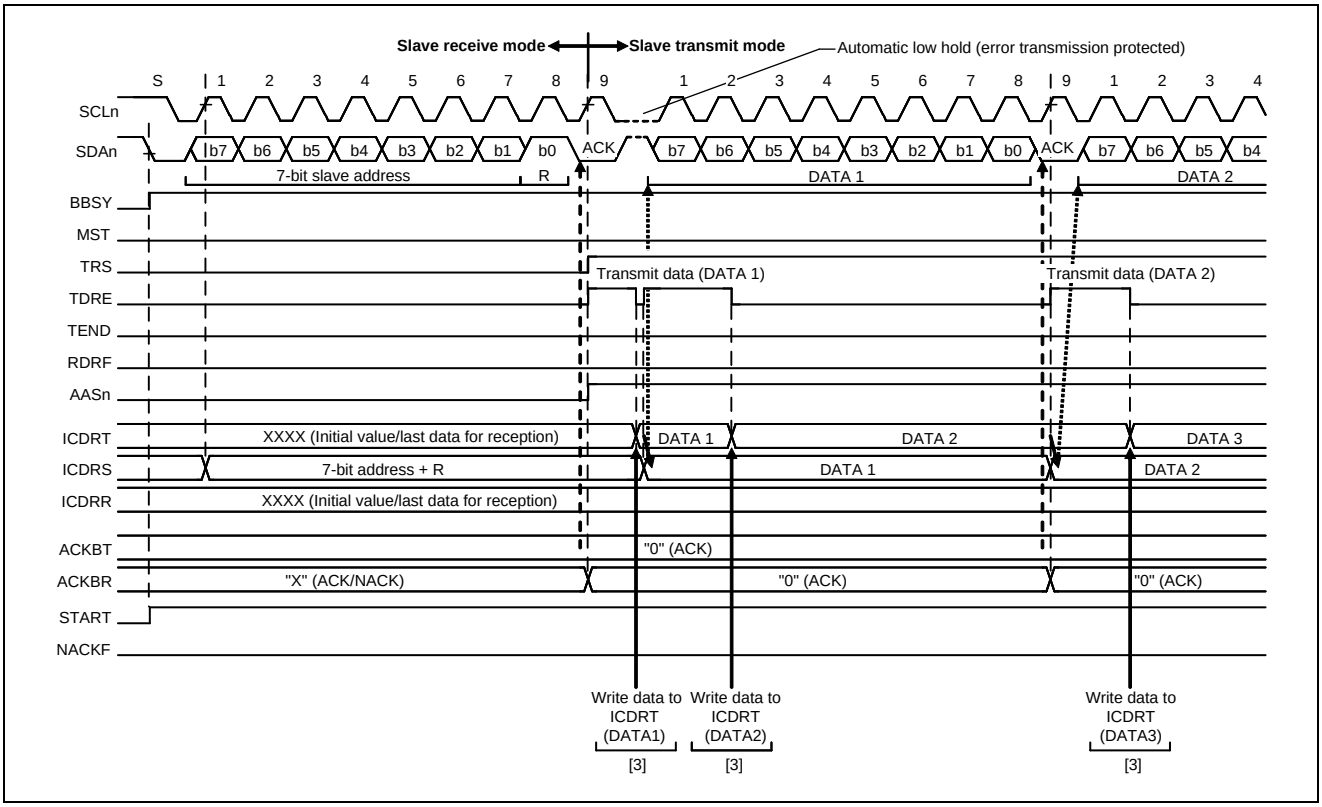



**Figure 22.15 Slave Transmit Operation Timing (1) (7-Bit Address Format)**

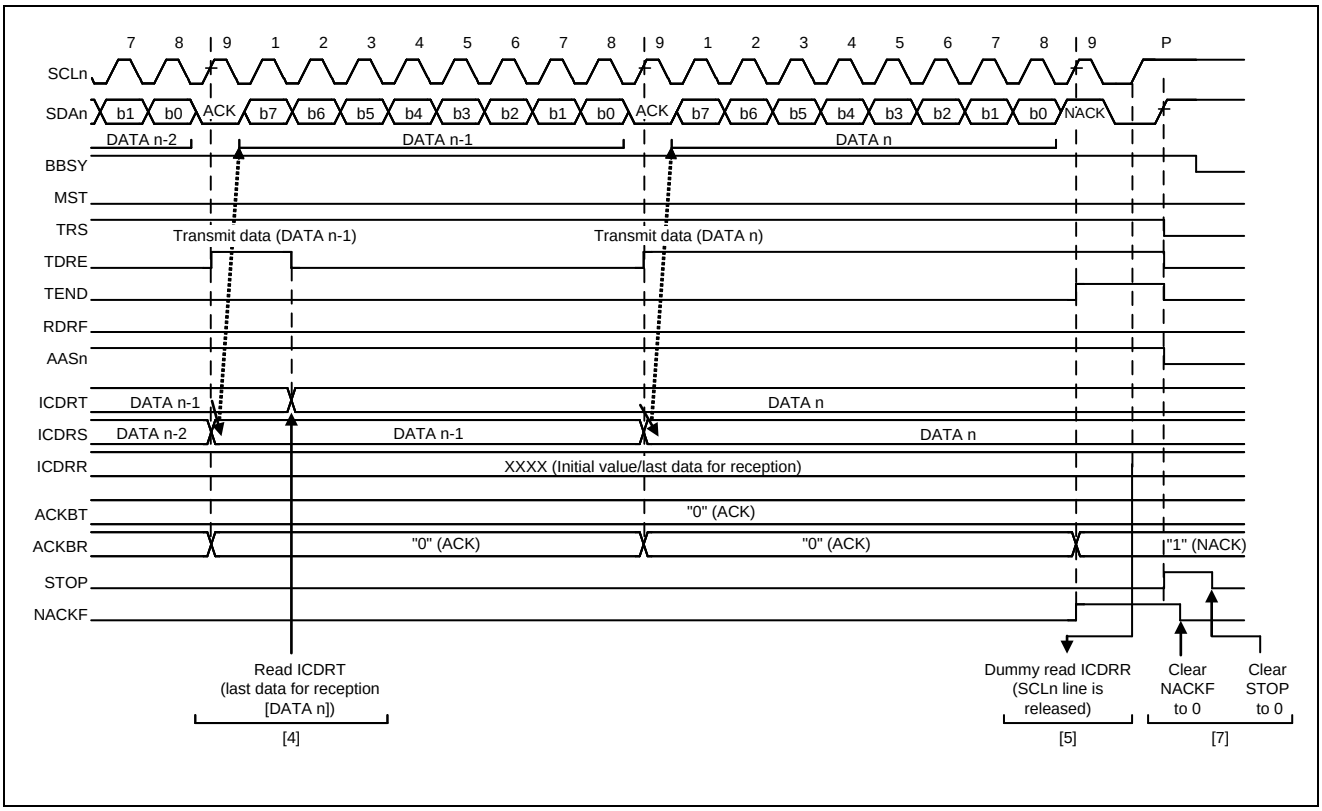



**Figure 22.16 Slave Transmit Operation Timing (2)**



Downloaded from Arrow.com.

### 22.3.6 Slave Receiver Operation

In slave receiver operation, the master device outputs the SCL clock and data, and the RIIC returns acknowledgements as a slave device.

Figure 22.17 shows an example of usage of slave reception and figures 22.18 and 22.19 show the timing of operations in slave reception.

The following describes the procedure and operations for slave reception.

1. Follow the procedure in figure 22.5 to make initial settings for the RIIC. This step is not necessary if initialization of the RIIC has already been completed. After initial settings, the RIIC will stay in the standby state until it receives a slave address that it matches.
2. After receiving a matching slave address, the RIIC sets one of the corresponding bits ICSR1.HOA, GCA, and AASy (y = 0 to 2) to 1 on the rising edge of the ninth cycle of SCL clock (the clock signal) and outputs the value set in the ICMR3.ACKBT bit to the acknowledge bit on the ninth cycle of SCL clock. If the value of the R/W# bit that was also received at this time is 1, the RIIC continues to place itself in slave receiver mode and sets the RDRF flag in ICSR2 to 1.
3. After the ICSR2.STOP flag is confirmed to be 0 and the ICSR2.RDRF flag to be 1, dummy read ICDRR (the dummy value consists of the slave address and R/W# bit when the 7-bit address format is selected, or the lower eight bits when the 10-bit address format is selected).
4. When ICDRR is read, the RIIC automatically clears the ICSR2.RDRF flag to 0. If reading of ICDRR is delayed and a next byte is received while the RDRF flag is still set to 1, the RIIC holds the SCLn line low from one SCL cycle before the timing with which RDRF should be set. In this case, reading ICDRR releases the SCLn line from being held at the low level.
   When the ICSR2.STOP flag is 1 and the ICSR2.RDRF flag is also 1, read ICDRR until all the data is completely received.
5. Upon detecting the stop condition, the RIIC automatically clears bits ICSR1.HOA, GCA, and AASy (y = 0 to 2) to 0.
6. After checking that the ICSR2.STOP flag is 1, clear the ICSR2. STOP flag to 0 for the next transfer operation.



Downloaded from Arrow.com.

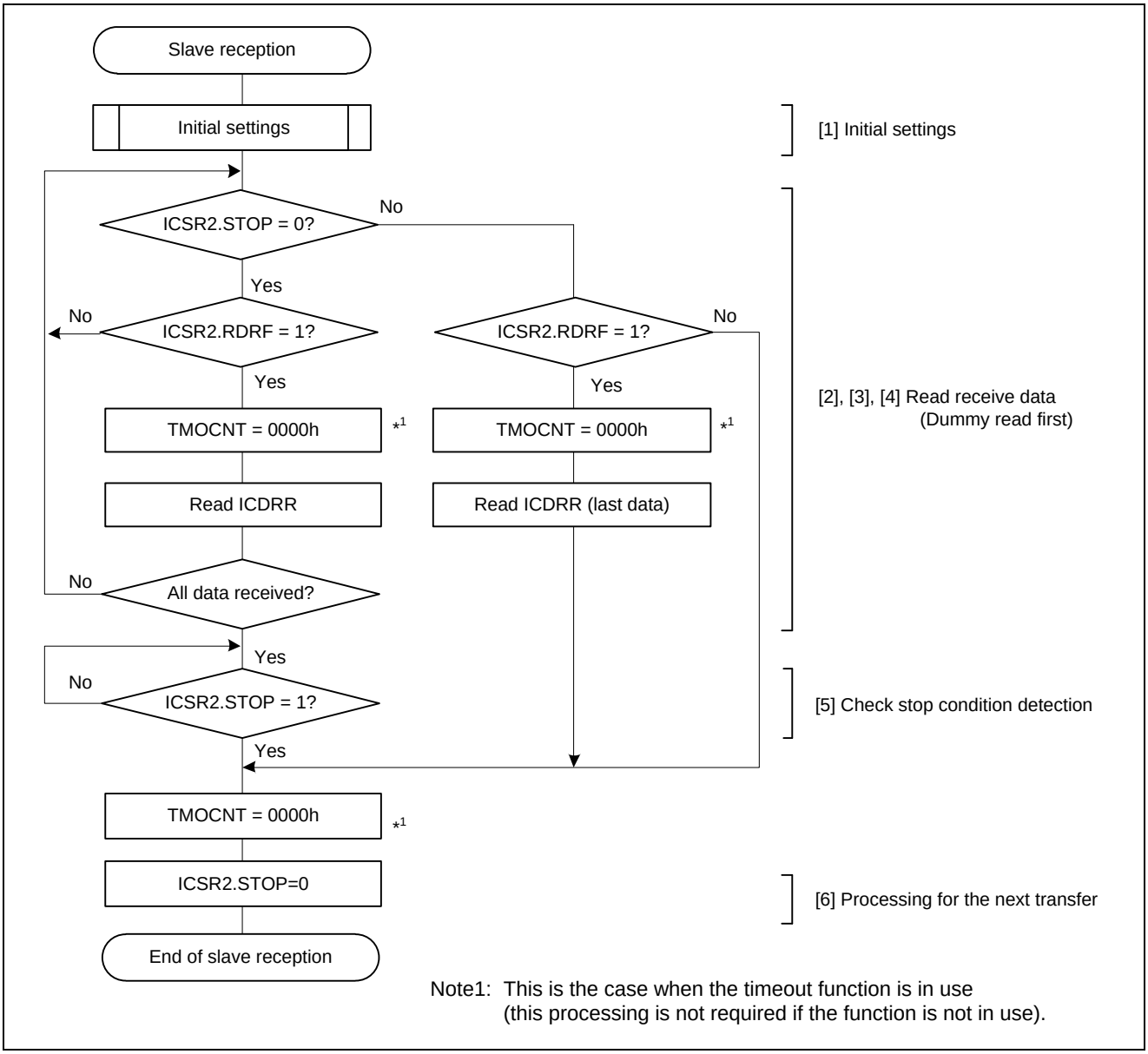



**Figure 22.17 Example of Slave Reception Flowchart**



Downloaded from Arrow.com.

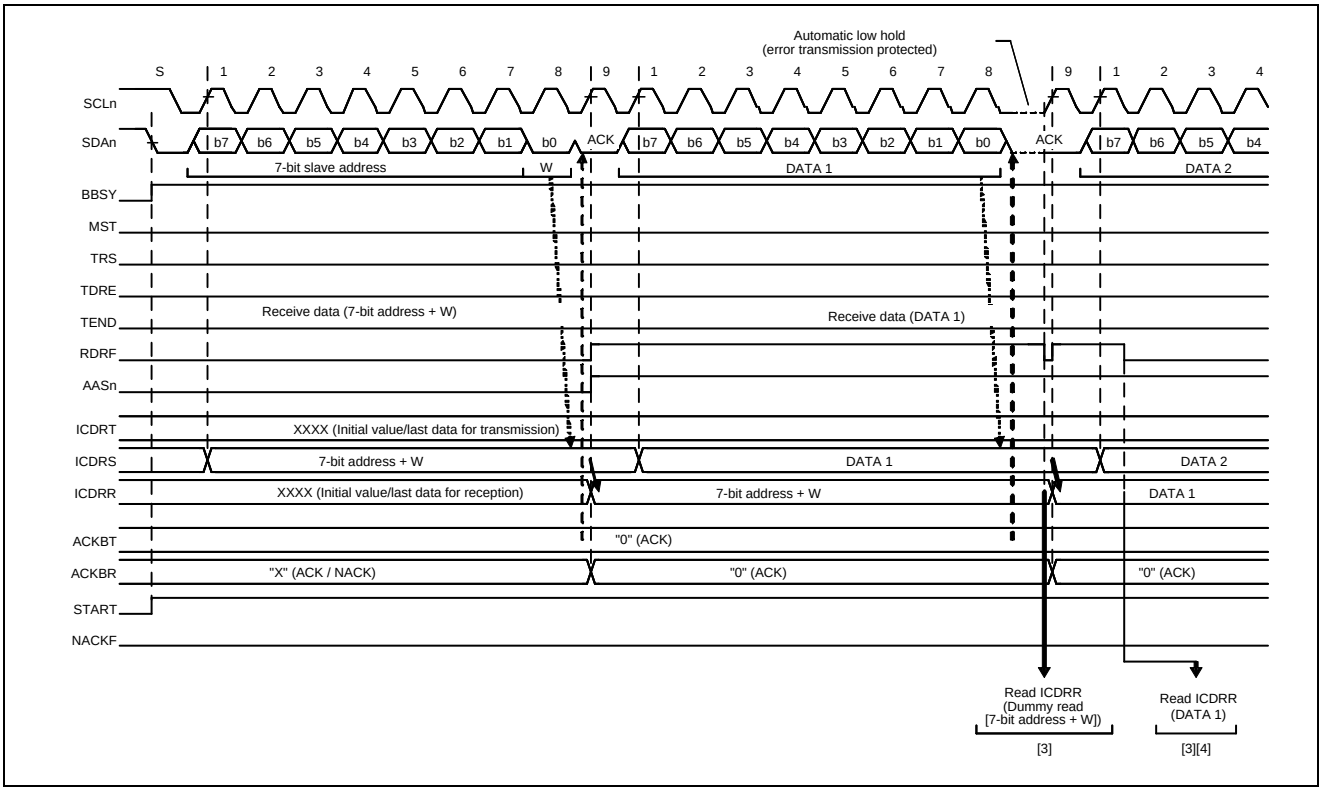



**Figure 22.18 Slave Receive Operation Timing (1) (7-Bit Address Format, when RDRFS=0)**

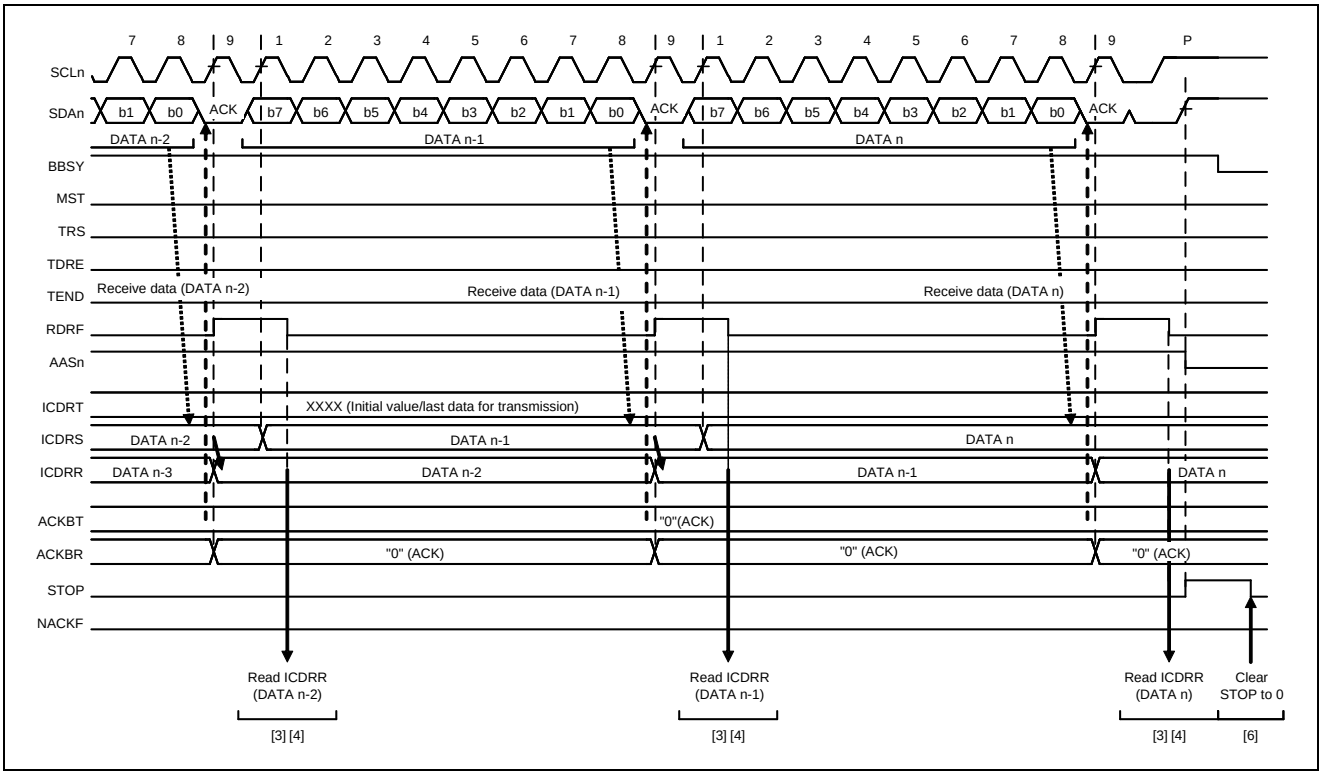



**Figure 22.19 Slave Receive Operation Timing (2) (when RDRFS=0)**



Downloaded from Arrow.com.

## 22.4 SCL Synchronization Circuit

In generation of the SCL (clock) signal, the RIIC starts counting out the value for width at high level specified in ICBRH when it detects a rising edge on the SCLn line and drives the SCLn line low once counting of the width at high level is complete. When the RIIC detects the falling edge of the SCLn line, it starts counting out the width at low level period specified in ICBRL, and then stops driving the SCLn line (releases the line) once counting of the width at low level is complete. The SCL (clock) signal is thus generated.

If multiple master devices are connected to the I²C bus, a collision of SCL signals may arise due to contention with another master device. In such cases, the master devices have to synchronize their SCL signals. Since this synchronization of SCL signals must be bit by bit, the RIIC is equipped with a facility (the SCL synchronization circuit) to obtain bit-by-bit synchronization of the SCL clock signals by monitoring the SCLn line while in master mode.

When the RIIC has detected a rising edge on the SCLn line and thus started counting out the width at high level specified in ICBRH, and the level on the SCL line falls because an SCL signal is being generated by another master device, the RIIC stops counting when it detects the falling edge, drives the level on the SCL line low, and starts counting out the width at low level specified in ICBRL. When the RIIC finishes counting out the width at low level, it stops driving the SCLn line to the low level (i.e. releases the line). At this time, if the width at low level of the SCL clock signal from the other master device is longer than the width at low level set in the RIIC, the width at low level of the SCL signal will be extended. Once the width at low level for the other master device has ended, the SCL signal rises because the SCL line has been released. When the RIIC finishes outputting the low-level period of the SCL clock of, the SCLn line is released and the SCL clock rises. That is, in cases of contention of SCL signals from more than one master, the width at high level of the SCL signal is synchronized with that of the clock having the narrower width, and the width at low level of the SCL signal is synchronized with that of the clock having the broader width. However, such synchronization of the SCL signal is only enabled when the SCLE bit in ICFER is set to 1.

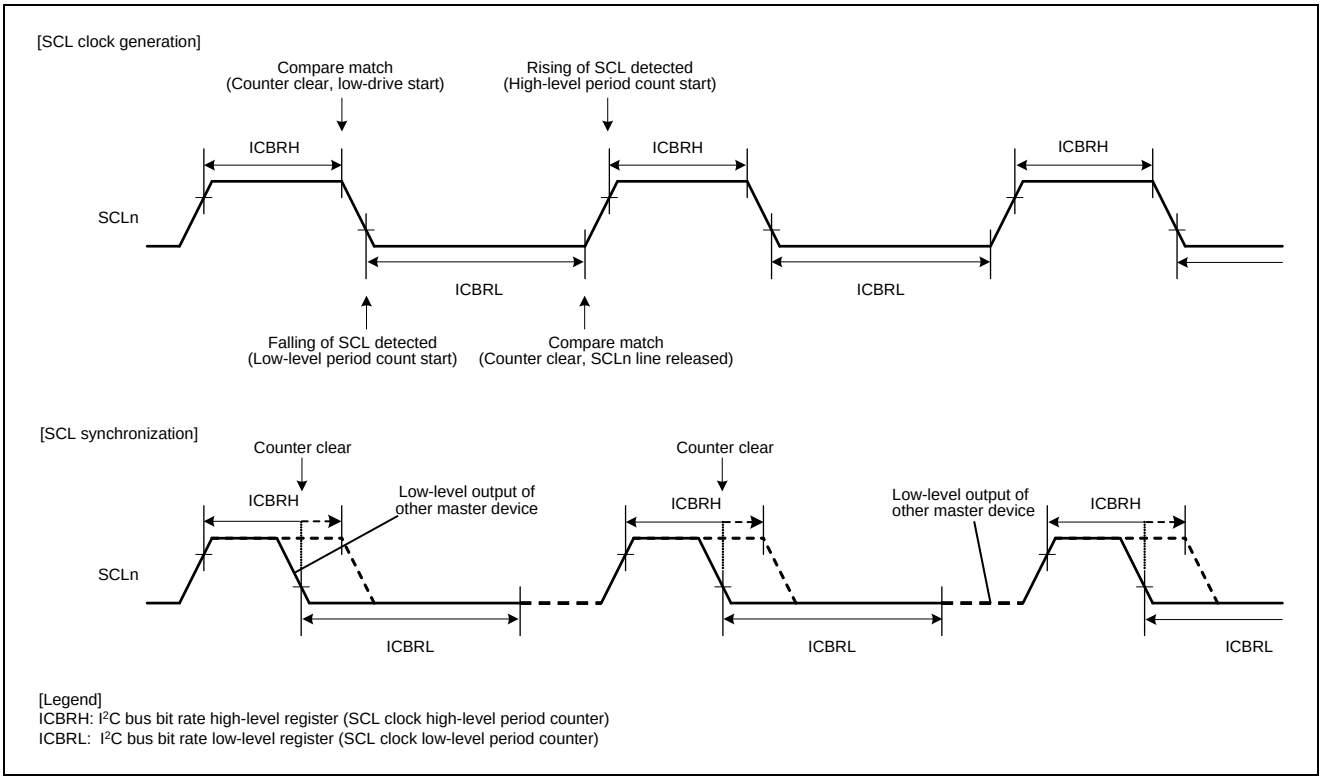



**Figure 22.20 Generation and Synchronization of the SCL Signal from the RIIC**



Downloaded from Arrow.com.

## 22.5 Facility for Delaying SDA Output

The RIIC module incorporates a facility for delaying output on the SDA line. The delay can be applied to all output (issuing of the start, restart, and stop conditions, data, and the ACK and NACK signals) on the SDA line.

With the SDA output delay facility, SDA output is delayed from detection of a falling edge of the SCL signal to ensure that the SDA signal is output within the interval over which the SCL (clock) signal is at the low level. Doing this leads to usage with the aim of preventing erroneous operation of communications devices, with the aim of satisfying the 300-ns (min.) data-hold time requirement of the SMBus specification.

The output delay facility is enabled by setting the SDDL[2:0] bits in IMCR2 to any value other than 000b, and disabled by setting the same bits to 000b.

While the SDA output delay facility is enabled (i.e. while the SDDL[2:0] bits in IMCR2 are set to any value other than 000b), the DLCS bit in IMCR2 selects the clock source for counting by the SDA output delay counter as the internal base clock (IICϕ) for the IIC module or as a clock signal derived by dividing the frequency of the internal base clock by two (IICϕ/2). The counter counts the number of cycles set in the SDDL[2:0] bits in IMCR2. After counting of the set number of cycles of delay is completed, the RIIC module places the required output (start, restart, or stop condition, data, or an ACK or NACK signal) on the SDA line.

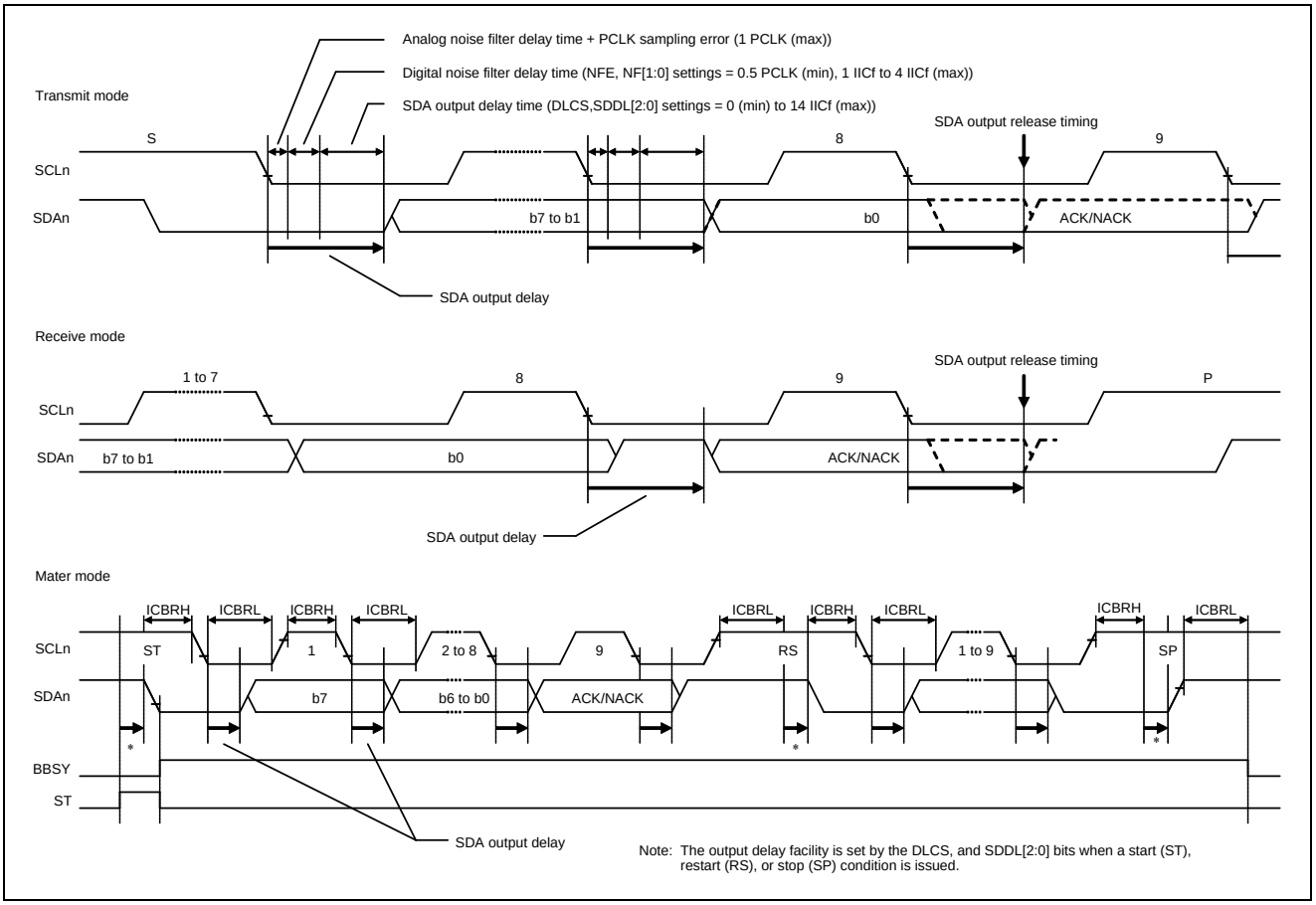



**Figure 22.21 SDA Output Delay Facility**



Downloaded from Arrow.com.

## 22.6 Digital Noise-Filter Circuits

The states of the SCLn and SDAn pins are conveyed to the internal circuitry through analog noise-filter and digital noise-filter circuits. Figure 22.22 is a block diagram of the digital noise-filter circuit.

The on-chip digital noise-filter circuit of the RIIC consists of four flip-flop circuit stages connected in series and a match-detection circuit.

The number of effective stages in the digital noise filter is selected by the NF[1:0] bits in ICMR3. The selected number of effective stages determines the noise-filtering capability as a period from one to four PCLK cycles.

The input signal to the SCLn pin (or SDAn pin) is sampled on falling edges of the PCLK signal. When the input signal level matches the output level of the number of effective flip-flop circuit stages as selected by the NF[1:0] bits in ICMR3, the signal level is conveyed to the subsequent stage. If the signal levels do not match, the previous value is retained.

If the ratio between the frequency of the internal operating clock (PCLK) and the transfer rate is small (e.g. data transfer at 400 kbps with PCLK = 4 MHz), the characteristics of the digital noise filter may lead to the elimination of needed signals as noise. In such cases, it is possible to disable the digital noise-filter circuit (by clearing the NFE bit in ICFER) and use only the analog noise-filter circuit.

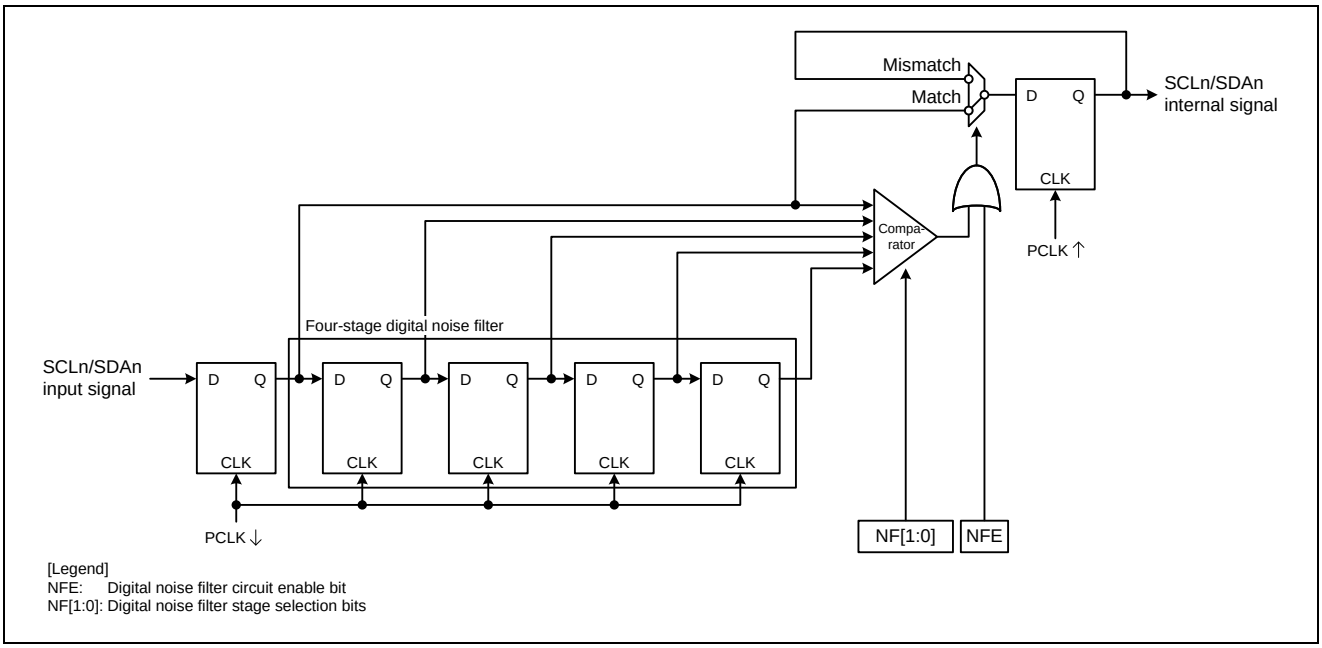



**Figure 22.22 Block Diagram of Digital Noise Filter Circuit**



Downloaded from Arrow.com.

## 22.7 Address Match Detection

The RIIC can set three unique slave addresses in addition to the general call address and host address, and also can set 7-bit or 10-bit slave addresses.

### 22.7.1 Slave-Address Match Detection

The RIIC can set three unique slave addresses, and has a slave address detection function for each unique slave address. When the SARyE bit (y = 0 to 2) in ICSER is set to 1, the slave addresses set in SARUy and SARLy (y = 0 to 2) can be detected.

When the RIIC detects a match of the set slave address, the corresponding AASy flag (y = 0 to 2) in ICSR1 is set to 1 at the falling edge of the ninth SCL clock cycle, and the RDRF flag in ICSR2 or the TDRE flag in ICSR2 is set to 1 by the following R/W# bit. This causes a receive data full interrupt (ICRXI) or transmit data empty interrupt (ICTXI) to be generated. The AASy flag is used to identify which slave address has been specified.

Figures 22.23 to 22.25 show the AASy flag set timing in three cases.

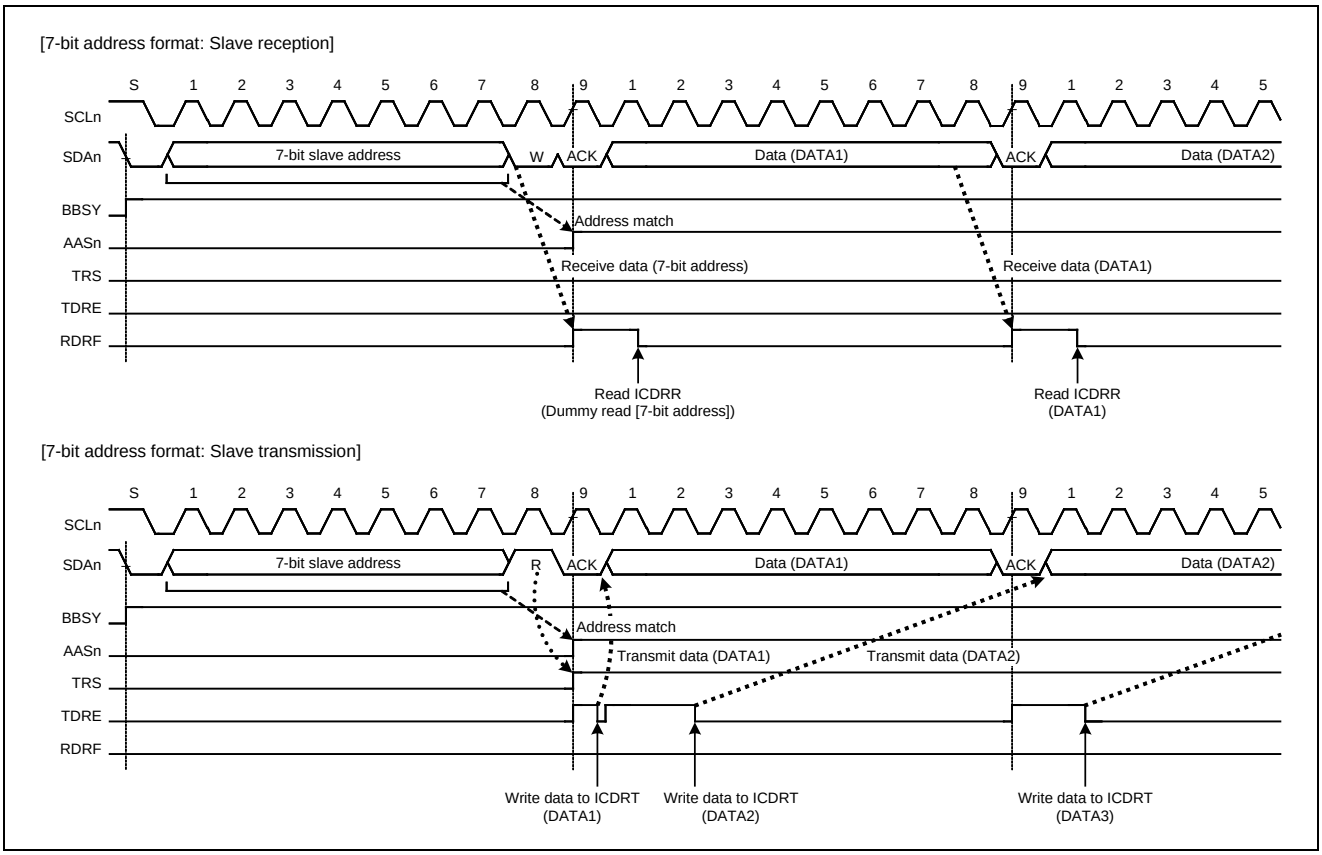



**Figure 22.23 Timing of AASy Flag Setting to 1 with 7-Bit Address Format Selected**



Downloaded from Arrow.com.

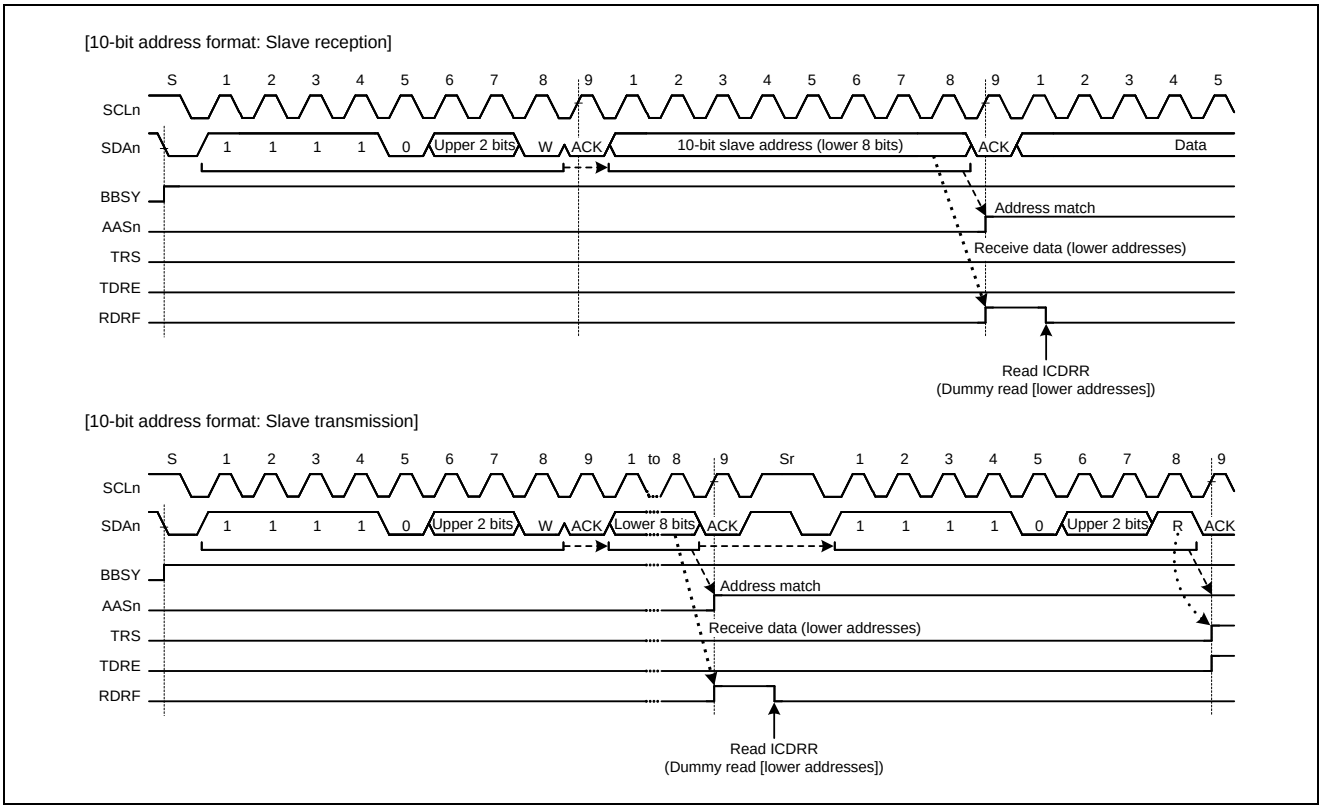



**Figure 22.24 Timing of AASy Flag Setting to 1 with 10-Bit Address Format Selected**

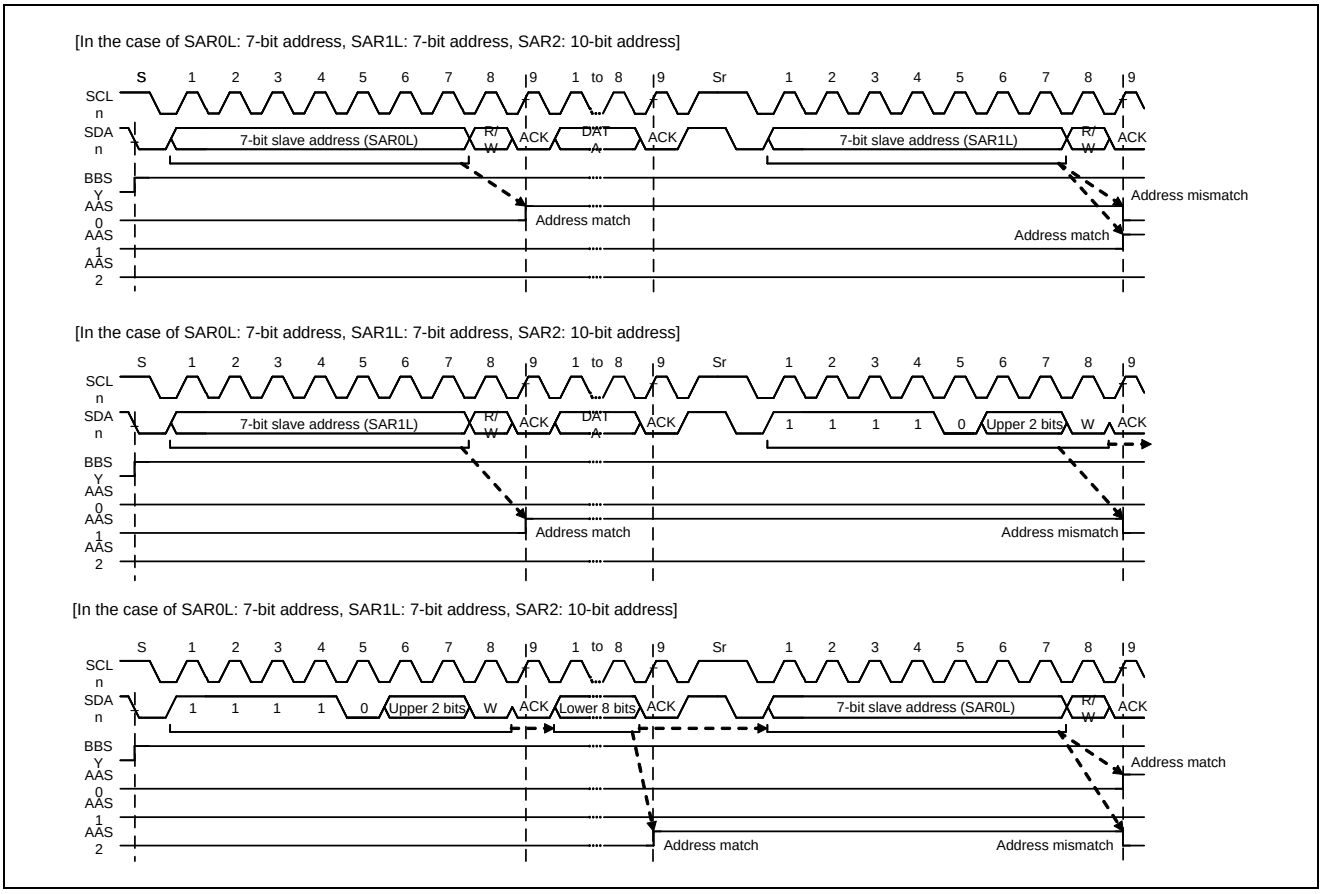



**Figure 22.25 Timing of AASy Flag Setting to 1/0 with 7-Bit/10-Bit Address Formats Mixed**



Downloaded from Arrow.com.

### 22.7.2 Detection of the General Call Address

The RIIC has a facility for detecting the general call address (0000 000b + 0 [W]). This is enabled by setting the GCAE bit in ICSER to 1.

If the address received after a start or restart condition is issued is 0000 000b + 1[R] (start byte), the RIIC recognizes this as the address of a slave device with an "all-zero" address but not as the general call address.

When the RIIC detects the general call address, both the GCA flag in ICSR1 and the RDRF flag in ICSR2 are set to 1 on the falling edge of the ninth cycle of SCL clock. This leads to the generation of a receive data full interrupt (ICRXI). The value of the GCA flag can be confirmed to recognize that the general call address has been transmitted.

Operation after detection of the general call address is the same as normal slave receive operation.

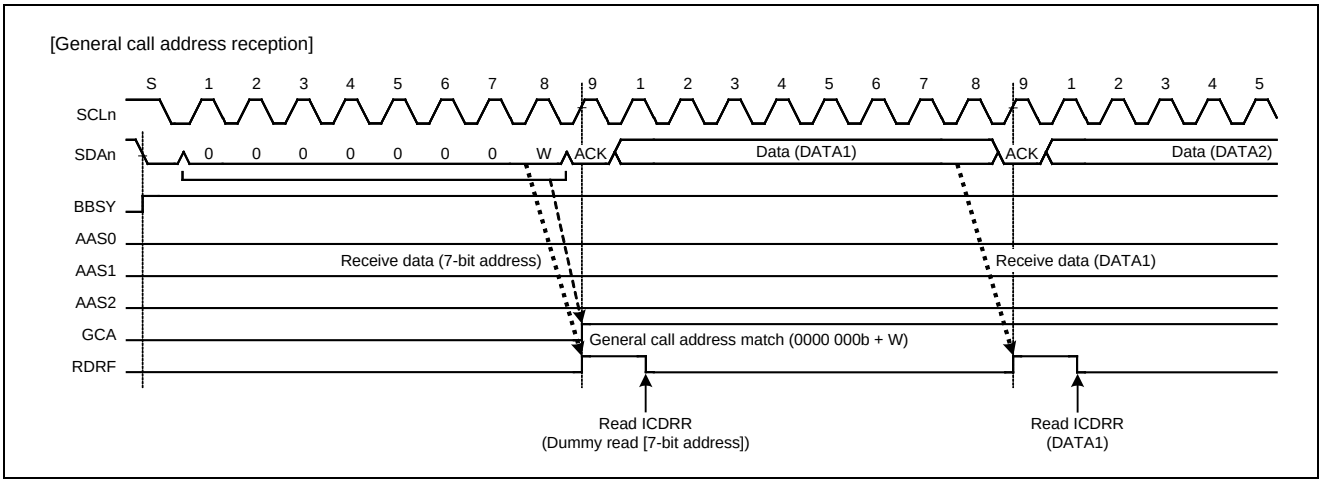



**Figure 22.26 Timing of GCA Flag Setting to 1 during Reception of General Call Address**



Downloaded from Arrow.com.

### 22.7.3 Device-ID Address Detection

The RIIC module has a facility for detecting device-ID addresses conformant with the I²C bus specification (Rev. 03). When the RIIC receives 1111 100b as the first byte after a start condition or restart condition was issued with the DIDE bit in ICSER set to 1, the RIIC recognizes the address as a device ID, sets the DID flag in ICSR1 to 1 on the rising edge of the ninth SCL clock cycle when the following R/W# bit is 0, and then compares the second and subsequent bytes with its own slave address. If the address matches the value in the slave address register, the RIIC sets the corresponding AASy flag (y = 0 to 2) in ICSR1 to 1.

After that, when the first byte received after a restart condition is issued matches the device ID address (1111 100b) again and the following R/W# bit is 1, the RIIC does not compare the second and subsequent bytes and sets the ICSR2.TDRE flag to 1.

In the device-ID address detection function, the RIIC clears the DID flag to 0 if a match with the RIIC's own slave address is not obtained or a match with the device ID address is not obtained after a match with the RIIC's own slave address and the detection of a restart condition. If the first byte after detection of a start or restart condition matches the device ID address (1111 100b) and the R/W# bit is 0, the RIIC sets the DID flag to 1 and compares the second and subsequent bytes with the RIIC's slave address. If the R/W# bit is 1, the DID flag holds the previous value and the RIIC does not compare the second and subsequent bytes. Therefore, the reception of a device-ID address can be checked by reading the DID flag after confirming that TDRE = 1.

Furthermore, prepare the device-ID fields (three bytes: 12 bits indicating the manufacturer + 9 bits identifying the part + 3 bits indicating the revision) that must be sent to the host after reception of a continuous device-ID field as normal data for transmission. For details of the information that must be included in device-ID fields, contact NXP Semiconductors.



Downloaded from Arrow.com.

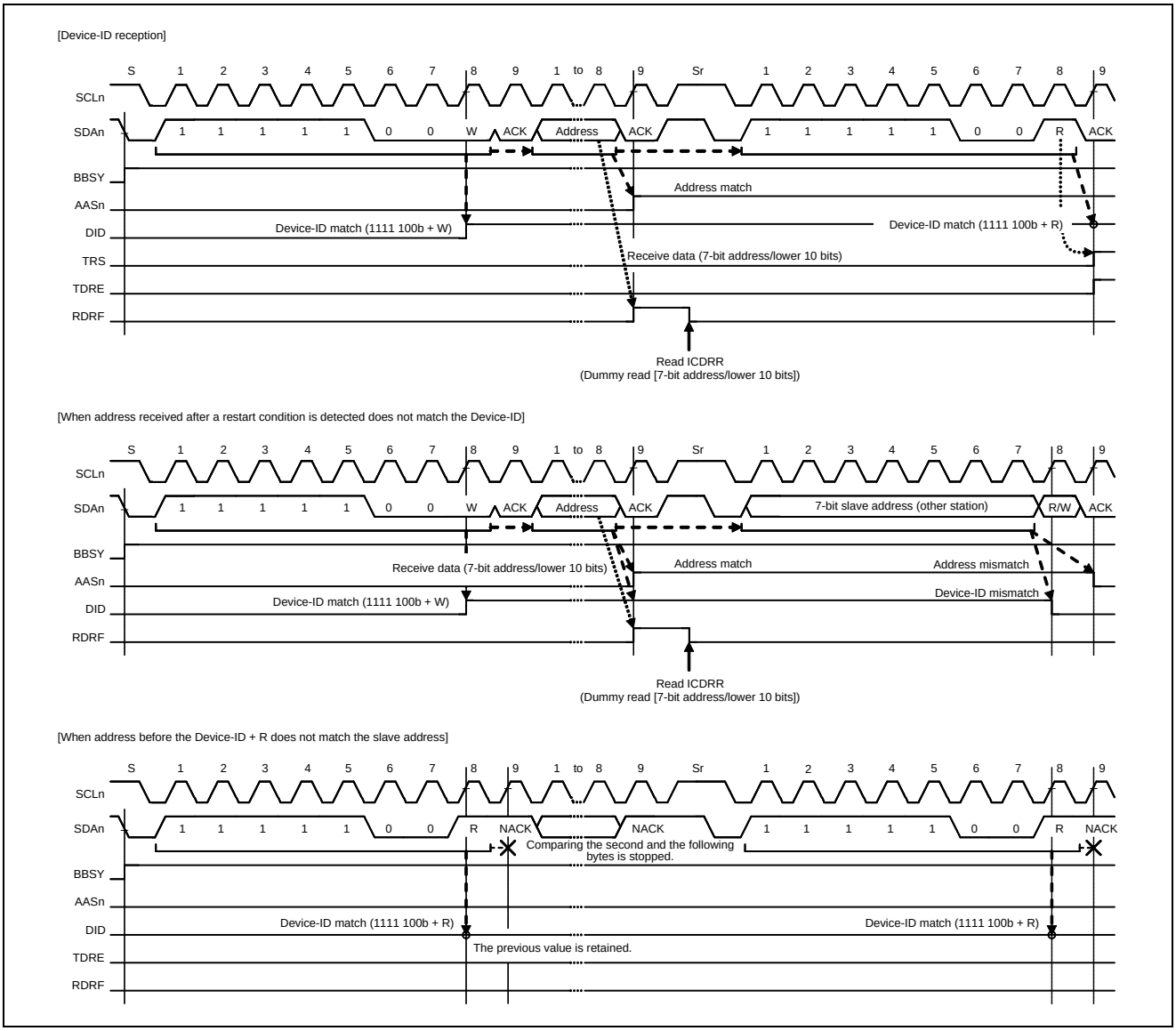



**Figure 22.27 AASy/DID Flag Set/Clear Timing during Reception of Device-ID**



Downloaded from Arrow.com.

### 22.7.4 Host Address Detection

The RIIC has a function to detect the host address while the SMBus is operating. When the HOAE bit in ICSER is set to 1 while the SMBS bit in ICMR3 is 1, the RIIC can detect the host address (0001 000b) in slave receive mode (MST and TRS bits = 00b in ICCR2).

When the RIIC detects the host address, the HOA flag in ICSR1 is set to 1 at the falling edge of the ninth SCL clock cycle, and at the same time, the TDRE flag in ICSR2 is set to 1 when the R/W# bit is 0 (Wr bit). This causes a transmit data empty interrupt (ICTXI) to be generated. The HOA flag is used to recognize that the host address was sent from the smart battery or other devices.

If the bit following the host address (0001 000b) is an Rd bit (R/W# bit = 1), the RIIC can also detect the host address. After the host address is detected, the RIIC operates in the same manner as normal slave operation.

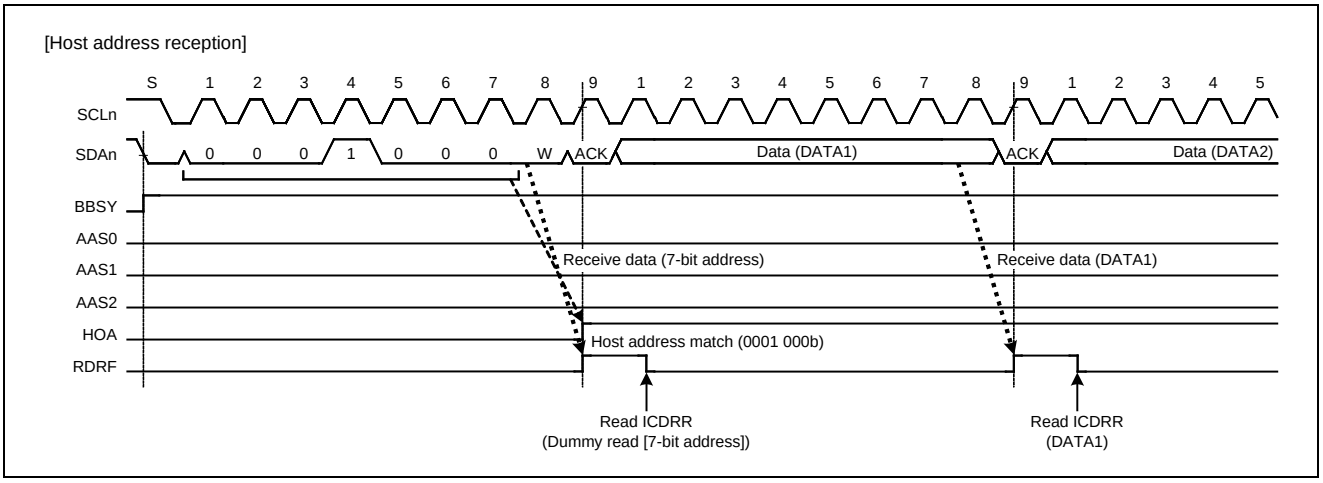



**Figure 22.28 HOA Flag Set Timing during Reception of Host Address**



Downloaded from Arrow.com.

## 22.8 Function to Automatically Hold SCLn Clock Low

### 22.8.1 Function to Prevent Wrong Transmission of Transmit Data

If the shift register (ICDRS) is empty when data have not been written to the transmit data register (ICDRT) with the RIIC in transmission mode (TRS bit = 1 in ICCR2), the SCLn signal is automatically held at the low level over the intervals shown below. This low-hold period is extended until data for transmission have been written, which prevents the unintended transmission of erroneous data.

<Master transmitter mode>

- Low-level interval after a start condition or restart condition is issued
- Low-level interval of one clock cycle between the ninth clock cycle of one transfer and the first clock cycle of the next

<Slave transmitter mode>

- Low-level interval between the ninth clock cycle of one transfer and the first clock cycle of the next

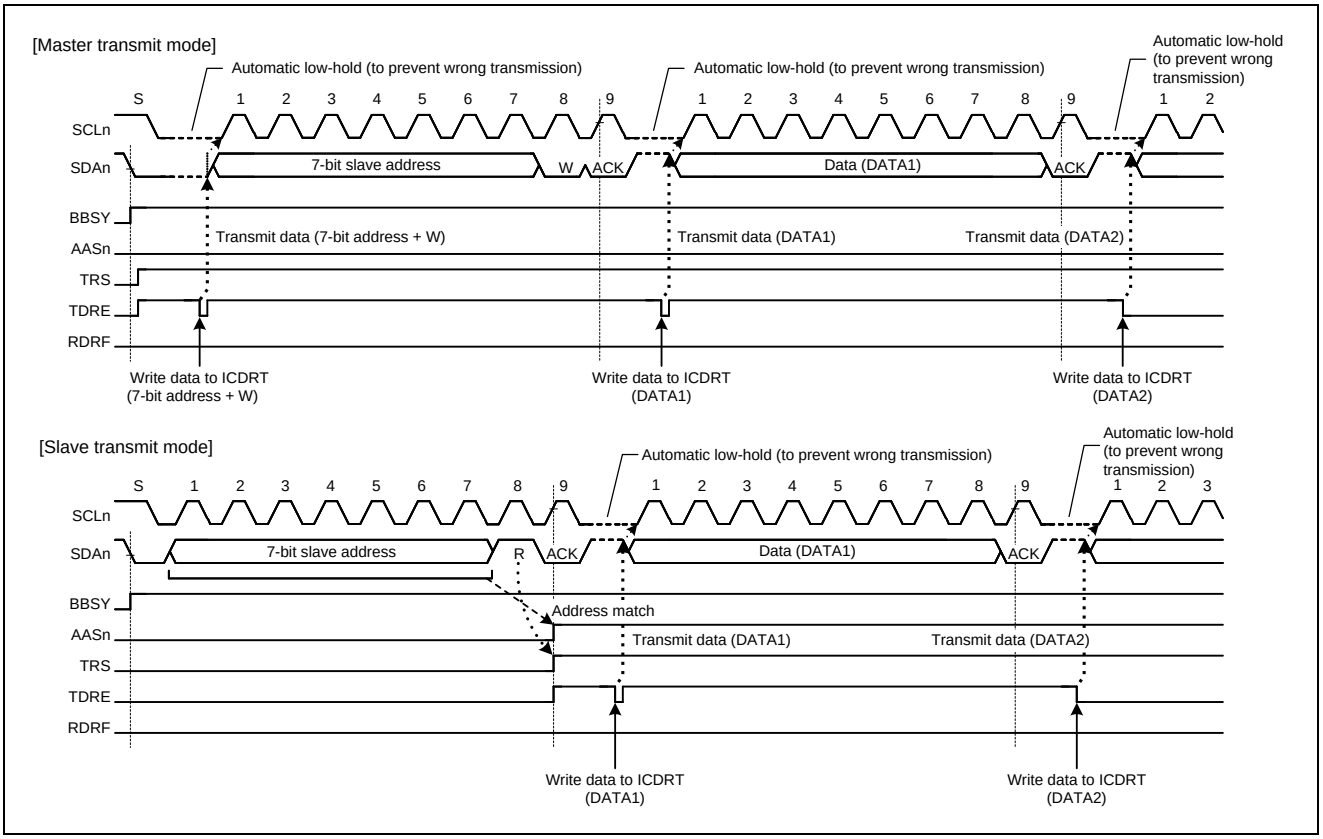



**Figure 22.29 Automatic Low-Hold Operation in Transmit Mode**



Downloaded from Arrow.com.

## 22.8.2 NACK Reception Transfer Suspension Function

The RIIC has a function to suspend transfer operation when NACK is received in transmit mode (TRS bit = 1 in ICCR2). This function is enabled when the NACKE bit in ICFER is set to 1 (transfer suspension enabled). If the next transmit data has already been written (TDRE flag = 0 in ICSR2) when NACK is received, next data transmission at the falling edge of the ninth SCL clock cycle is automatically suspended. This prevents the SDAn line output level from being held low when the MSB of the next transmit data is 0.

If the transfer operation is suspended by this function (NACKF flag = 1 in ICSR2), transmit operation and receive operation are discontinued. To restore transmit/receive operation, be sure to clear the NACKF flag to 0. In master transmit mode, clear the NACKF flag to 0, issue a restart or stop condition, and then issue a start condition again.

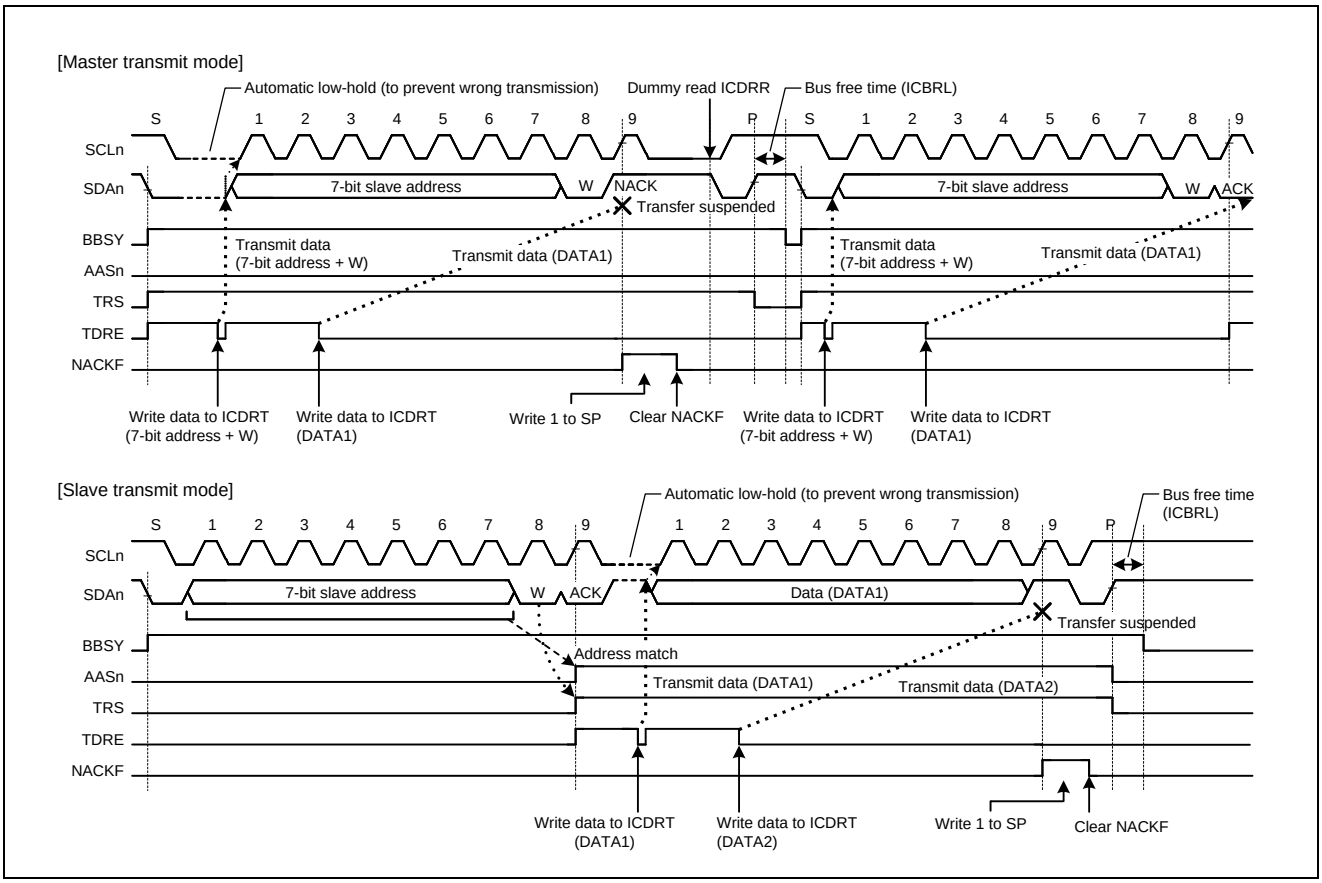



**Figure 22.30 Suspension of Data Transfer when NACK is Received (NACKE = 1)**



Downloaded from Arrow.com.

### 22.8.3 Function to Prevent Failure to Receive Data

If response processing is delayed when receive data (ICDRR) read is delayed for a period of one transfer frame or more with receive data full (RDRF flag = 1 in ICSR2) in receive mode (TRS = 0 in ICCR2), the RIIC holds the SCLn line low automatically immediately before the next data is received to prevent failure to receive data.

This function to prevent failure to receive data using the automatic low-hold function is also enabled even if the read processing of the final receive data is delayed and, in the meantime, the RIIC's own slave address is designated after a stop condition is issued. This function does not disturb other communication because the RIIC does not hold the SCLn line low when a mismatch with its own slave address occurs after a stop condition is issued.

Sections in which the SCLn line is held low can be selected with a combination of the WAIT and RDRFS bits in ICMR3.

#### (1) One-Byte Receive Operation and Automatic Low-Hold Function Using the WAIT Bit

When the WAIT bit in ICMR3 is set to 1, the RIIC performs one-byte receive operation using the WAIT bit function. Furthermore, when the ICMR3.RDRFS bit is 0, the RIIC automatically sends the ACKBT bit value in ICMR3 for the acknowledge bit in the period from the falling edge of the eighth SCL clock cycle to the falling edge of the ninth SCL clock cycle, and automatically holds the SCLn line low at the falling edge of the ninth SCL clock cycle using the WAIT bit function. This low-hold is released by reading data from ICDRR, which enables bytewise receive operation.

The WAIT bit function is enabled for receive frames after a match with the RIIC's own slave address (including the general call address and host address) is obtained in master receive mode or slave receive mode.

#### (2) One-Byte Receive Operation (ACK/NACK Transmission Control) and Automatic Low-Hold Function Using the RDRFS Bit

When the RDRFS bit in ICMR3 is set to 1, the RIIC performs one-byte receive operation using the RDRFS bit function. When the RDRFS bit is set to 1, the RDRF flag (receive data full) in ICSR2 is set to 1 at the rising edge of the eighth SCL clock cycle, and the SCLn line is automatically held low at the falling edge of the eighth SCL clock cycle. This low-hold is released by writing a value to the ACKBT bit in ICMR3, but cannot be released by reading data from ICDRR, which enables receive operation by the ACK/NACK transmission control according to the data received in byte units.

The RDRFS bit function is enabled for receive frames after a match with the RIIC’s own slave address (including the general call address and host address) is obtained in master receive mode or slave receive mode.



Downloaded from Arrow.com.

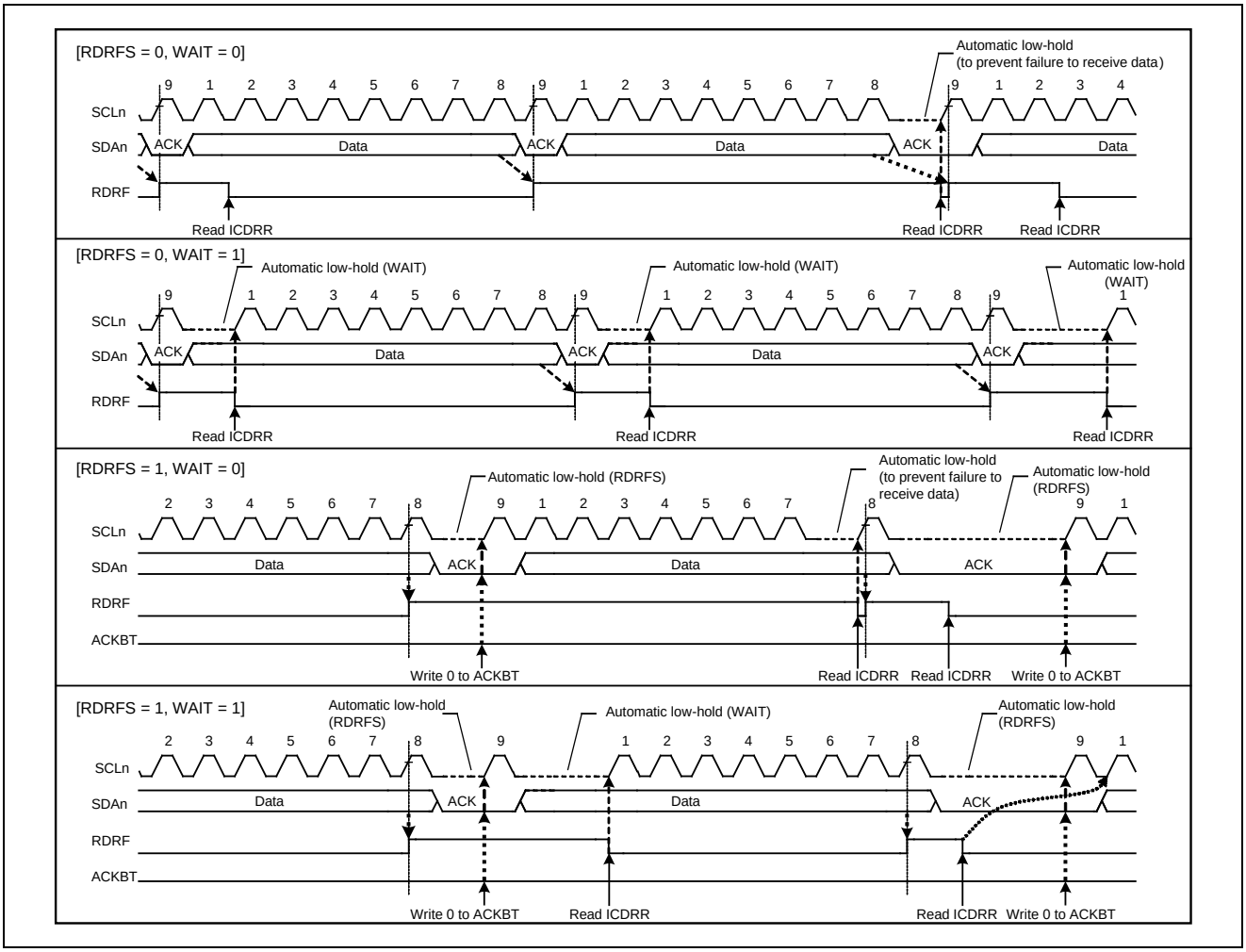



**Figure 22.31 Automatic Low-Hold Operation in Receive Mode (Using RDRFS and WAIT Bits)**



Downloaded from Arrow.com.

## 22.9 Arbitration-Lost Detection Functions

In addition to the normal arbitration-lost detection function defined by the I²C bus standard, the RIIC has functions to prevent double-issue of a start condition, to detect arbitration during transmission of NACK, and to detect arbitration in slave transmit mode.

### 22.9.1 Master Arbitration Lost Detection (MALE Bit)

The RIIC drives the SDAn line low to issue a start condition. However, if the SDAn line has already been driven low by another master device issuing a start condition, the RIIC regards its own issuing of a start condition as an error and considers this a loss in arbitration, so priority is given to transfer by the other master device. Similarly, if a request to issue a start condition is made by setting the ST bit in ICCR2 to 1 while the bus is busy (BBSY flag = 1 in ICCR2), the RIIC regards this as a double-issuing-of-start-condition error and considers itself to have lost in arbitration, thus preventing a failure of transfer due to issuing of a start condition while transfer is in progress.

When a start condition is issued successfully, if the data for transmission including the address bits (i.e. the internal SDA output level) and the level on the SDAn line do not match (the high output as the internal SDA output; i.e. the SDAn pin is in the high-impedance state) and the low level is detected on the SDAn line, the RIIC loses in arbitration.

After a loss in arbitration of mastership, the RIIC immediately enters slave receiver mode. If a slave address (including the general call address) matches its own address at this time, the RIIC continues in slave operation.

A loss in arbitration of mastership is detected when the following conditions are met while the MALE bit in ICFER is 1 (master arbitration lost detection enabled).

[Master arbitration lost conditions]

- Non-matching of the internal level for output on SDA and the level on the SDAn line after a start condition was issued by setting the ST bit in ICCR2 to 1 while the BBSY flag in ICCR2 was cleared to 0 (erroneous issuing of a start condition)
- Setting of the ST bit in ICCR2 to 1 (start condition double-issue error) while the BBSY flag is set to 1
- When the transmit data excluding acknowledge (internal SDA output level) does not match the level on the SDAn line in master transmit mode (MST and TRS bits = 11b in ICCR2)



Downloaded from Arrow.com.

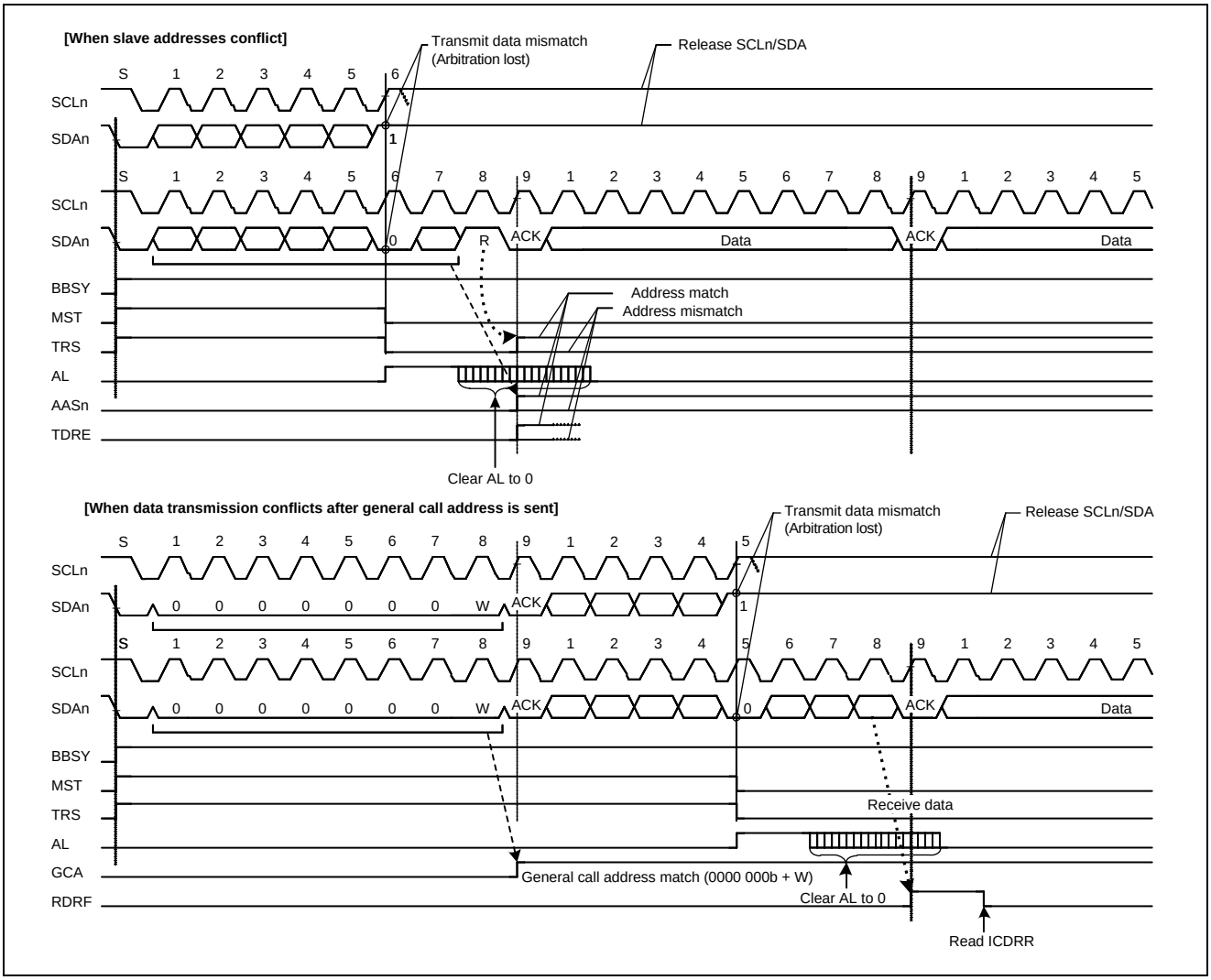



**Figure 22.32 Examples of Master Arbitration Lost Detection (MALE = 1)**

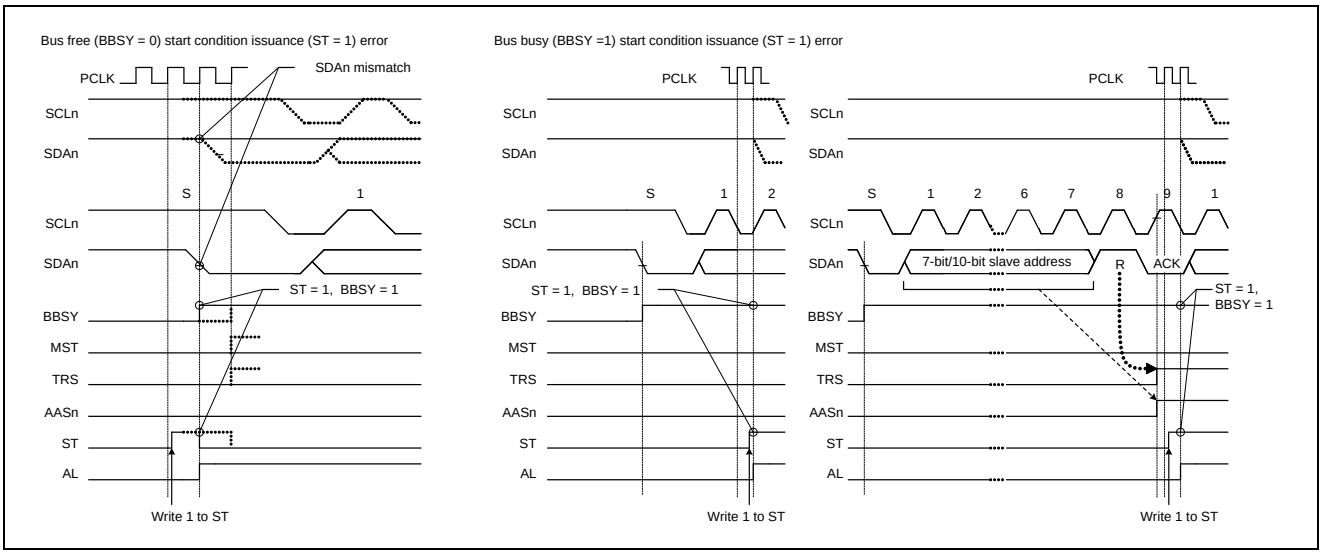



**Figure 22.33 Arbitration Lost when a Start Condition is Issued (MALE = 1)**



Downloaded from Arrow.com.

### 22.9.2 Function to Detect Loss of Arbitration during NACK Transmission (NALE Bit)

The RIIC has a function to cause arbitration to be lost if the internal SDA output level does not match the level on the SDAn line (the high output as the internal SDA output; i.e. the SDAn pin is in the high-impedance state) and the low level is detected on the SDA line during transmission of NACK in receive mode. Arbitration is lost due to a conflict of NACK transmission and ACK transmission when two or more master devices receive data from the same slave device simultaneously in a multi-master system. Such conflict occurs when multiple master devices send/receive the same information through a single slave device. Figure 22.34 shows an example of arbitration lost detection during transmission of NACK.

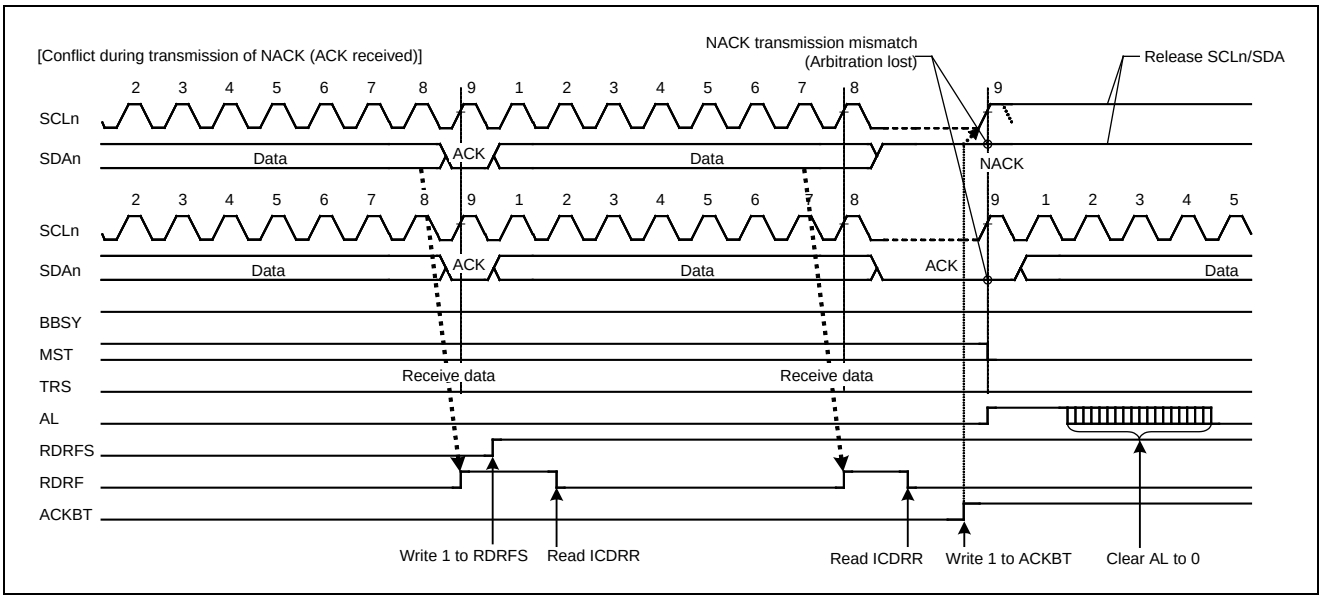



**Figure 22.34 Example of Arbitration Lost Detection during Transmission of NACK (NALE = 1)**

The following explains arbitration lost detection using an example where two master devices (master A and master B) and a single slave device are connected through the bus. In this example, master A receives two bytes of data from the slave device, and master B receives four bytes of data from the slave device.

If master A and master B access the slave device simultaneously, because the slave address is identical, arbitration is not lost in both master A and master B during access to the slave device. Therefore, both master A and master B recognize that they have obtained the bus mastership and operate as such. Here, master A sends NACK when it has received two final bytes of data from the slave device. Meanwhile, master B sends ACK because it has not received necessary four bytes of data. At this time, the NACK transmission from master A and the ACK transmission from master B conflict. In general, if a conflict like this occurs, master A cannot detect ACK transmitted by master B and issues a stop condition. Therefore, the issuance of the stop condition conflicts with the SCL clock output of master B, which disturbs communication.

When the RIIC receives ACK during transmission of NACK, it detects a defeat in conflict with other master devices and causes arbitration to be lost.

If arbitration is lost during transmission of NACK, the RIIC immediately cancels the slave match condition and enters slave receive mode. This prevents a stop condition from being issued, preventing a communication failure on the bus.

Similarly, in the ARP command processing of SMBus, the function to detect loss of arbitration during transmission of NACK is also available for eliminating the extra clock cycle processing (such as FFh transmission processing) necessary if no response is received in the Get UDID (general) processing after the Assign address command.



Downloaded from Arrow.com.

The RIIC detects arbitration lost during transmission of NACK when the following condition is met with the NALE bit in ICFER set to 1 (arbitration lost detection during NACK transmission enabled).

[Condition for arbitration lost during NACK transmission]

- When the internal SDA output level does not match the SDAn line (ACK is received) during transmission of NACK (ACKBT bit = 1 in ICMR3)

### 22.9.3 Slave Arbitration Lost Detection (SALE Bit)

The RIIC has a function to cause arbitration to be lost if the data for transmission (i.e. the internal SDA output level) and the level on the SDAn line do not match (the high output as the internal SDA output; i.e. the SDAn pin is in the high-impedance state) and the low level is detected on the SDA line in slave transmitter mode. This arbitration lost detection function is mainly used when transmitting a UDID (Unique Device Identifier) over an SMBus.

When it loses slave arbitration, the RIIC is immediately released from the slave-matched state and enters slave receiver mode. This function can detect conflicts of data during transmission of UDIDs over an SMBus and eliminates subsequent redundant processing (processing for the transmission of FFh).

The RIIC detects slave arbitration lost when the following condition is met with the SALE bit in ICFER set to 1 (slave arbitration lost detection enabled).

[Condition for slave arbitration lost]

- When transmit data excluding acknowledge (internal SDA output level) does not match the SDAn line in slave transmit mode (MST and TRS bits = 01b in ICCR2)

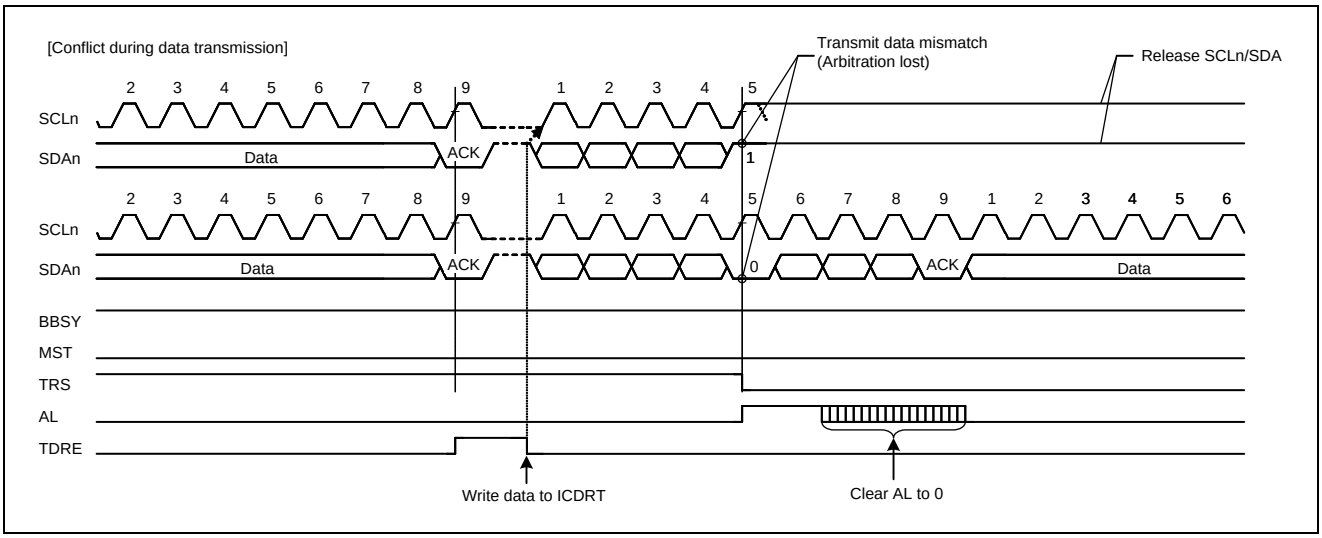



**Figure 22.35 Example of Slave Arbitration Lost Detection (SALE = 1)**



Downloaded from Arrow.com.

## 22.10 Start Condition/Restart Condition/Stop Condition Issuing Function

### 22.10.1 Issuing a Start Condition

The RIIC issues a start condition when the ST bit in ICCR2 is set to 1.

When the ST bit is set to 1, a start condition issuance request is made and the RIIC issues a start condition when the BBSY flag in ICCR2 is 0 (bus free). When a start condition is issued normally, the RIIC automatically shifts to the master transmit mode.

A start condition is issued in the following sequence.

[Start condition issuance]

- Drive the SDAn line low (high level to low level).
- Ensure the time set in ICBRH and the start condition hold time.
- Drive the SCLn line low (high level to low level).
- Detect low level of the SCLn line and ensure the low-level period of SCLn set in ICBRL.

### 22.10.2 Issuing a Restart Condition

The RIIC issues a restart condition when the RS bit in ICCR2 is set to 1.

When the RS bit is set to 1, a restart condition issuance request is made and the RIIC issues a restart condition when the BBSY flag in ICCR2 is 1 (bus busy) and the MST bit in ICCR2 is 1 (master mode).

A restart condition is issued in the following sequence.

[Restart condition issuance]

- Release the SDAn line.
- Ensure the low-level period of SCLn line set in ICBRL.
- Release the SCLn line (low level to high level).
- Detect a high level of the SCLn line and ensure the time set in ICBRL and the restart condition setup time.
- Drive the SDAn line low (high level to low level).
- Ensure the time set in ICBRH and the restart condition hold time.
- Drive the SCLn line low (high level to low level).
- Detect a low level of the SCLn line and ensure the low-level period of SCLn line set in ICBRL.



Downloaded from Arrow.com.

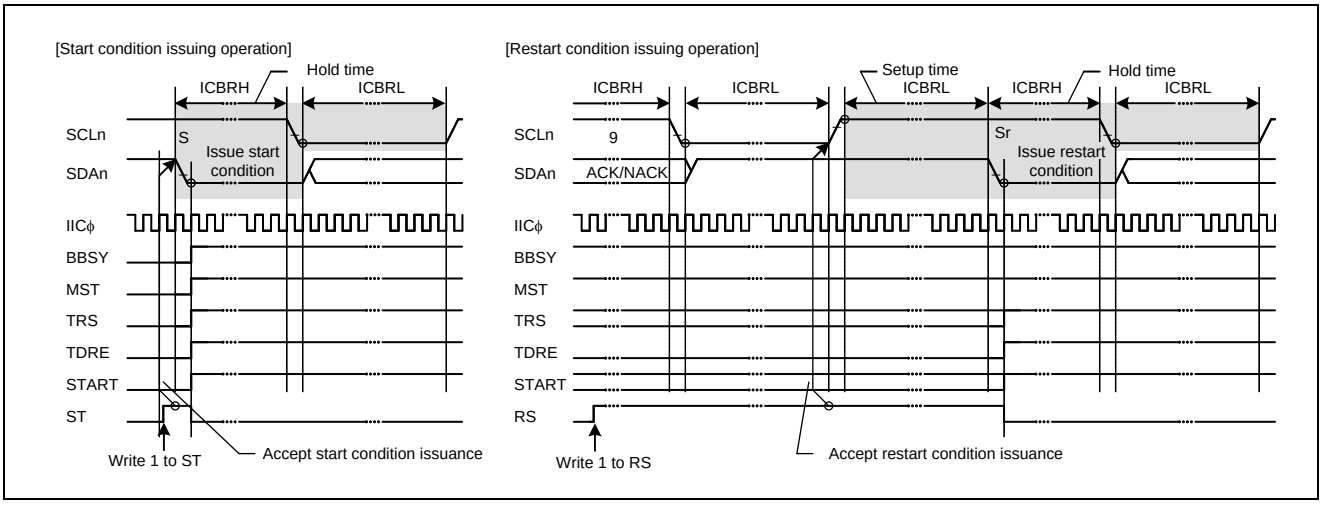



**Figure 22.36 Start Condition/Restart Condition Issue Timing (ST and RS Bits)**

## 22.10.3 Issuing a Stop Condition

The RIIC issues a stop condition when the SP bit in ICCR2 is set to 1.

When the SP bit is set to 1, a stop condition issuance request is made and the RIIC issues a stop condition when the BBSY flag in ICCR2 is 1 (bus busy) and the MST bit in ICCR2 is 1 (master mode).

A stop condition is issued in the following sequence.

[Stop condition issuance]

- Drive the SDAn line low (high level to low level).
- Ensure the low-level period of SCLn line set in ICBRL.
- Release the SCLn line (low level to high level).
- Detect a high level of the SCLn line and ensure the time set in ICBRH and the stop condition setup time.
- Release the SDAn line (low level to high level).
- Ensure the time set in ICBRL and the bus free time.
- Clear the BBSY flag to 0 (to release the bus mastership).

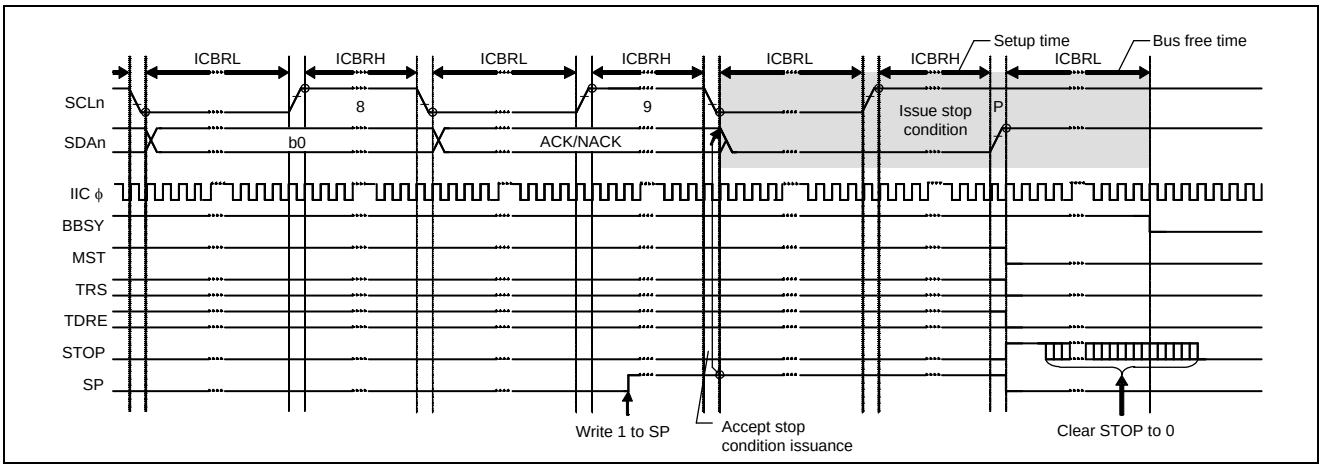



**Figure 22.37 Stop Condition Issue Timing (SP Bit)**



Downloaded from Arrow.com.

## 22.11 Bus Hanging

If the clock signals from the master and slave devices go out of synchronization due to noise or other factors, the I²C bus might hang with a fixed level on the SCLn line and/or SDAn line.

As measures against the bus hanging, the RIIC has a timeout detection function to detect hanging by monitoring the SCLn line, a function for the output of an extra SCL clock cycle to release the bus from a hung state due to clock signals being out of synchronization, and the RIIC/internal reset function.

By checking the SCLO, SDAO, SCLI, and SDAI bits in ICCR1, it is possible to see whether the RIIC or its partner in communications is placing the low level on the SCLn or SDAn lines.

### 22.11.1 Timeout Detection Function

The RIIC has the timeout detection function to detect an abnormality that the SCLn line is held for a certain period of time. In the bus busy state, the RIIC can detect an abnormal bus state by monitoring that the SCLn line is held low or high for a predetermined time.

The timeout detection function monitors the SCLn line state and counts the low-level period or high-level period using the internal counter. The timeout detection function resets the internal counter each time the SCLn line changes (rising or falling), but continues to count unless the SCLn line changes. If the internal counter overflows due to no SCLn line change, the RIIC can detect the timeout and report the bus abnormality.

This timeout detection function is enabled when the TMOE bit in ICFER is 1. It detects an abnormal bus state that the SCLn line is held low or high when the bus is busy (BBSY flag = 1 in ICCR2) in master mode or when the BBSY flag is 1 and the RIIC's own slave address matches (ICSR1 is not 00h) in slave mode.

The internal counter of the timeout detection function works using the internal reference clock (IICϕ) set by the CKS[2:0] bits in ICMR1 as a count source. It functions as a 16-bit counter when long mode is selected (TMOS bit = 0 in ICMR2) or a 14-bit counter when short mode is selected (TMOS bit = 1).

The SCLn line level (low/high or both levels) during which this counter is activated can be selected by the setting of the TMOH and TMOL bits in ICMR2. If both TMOL and TMOH bits are cleared to 0, the internal counter does not work.



Downloaded from Arrow.com.

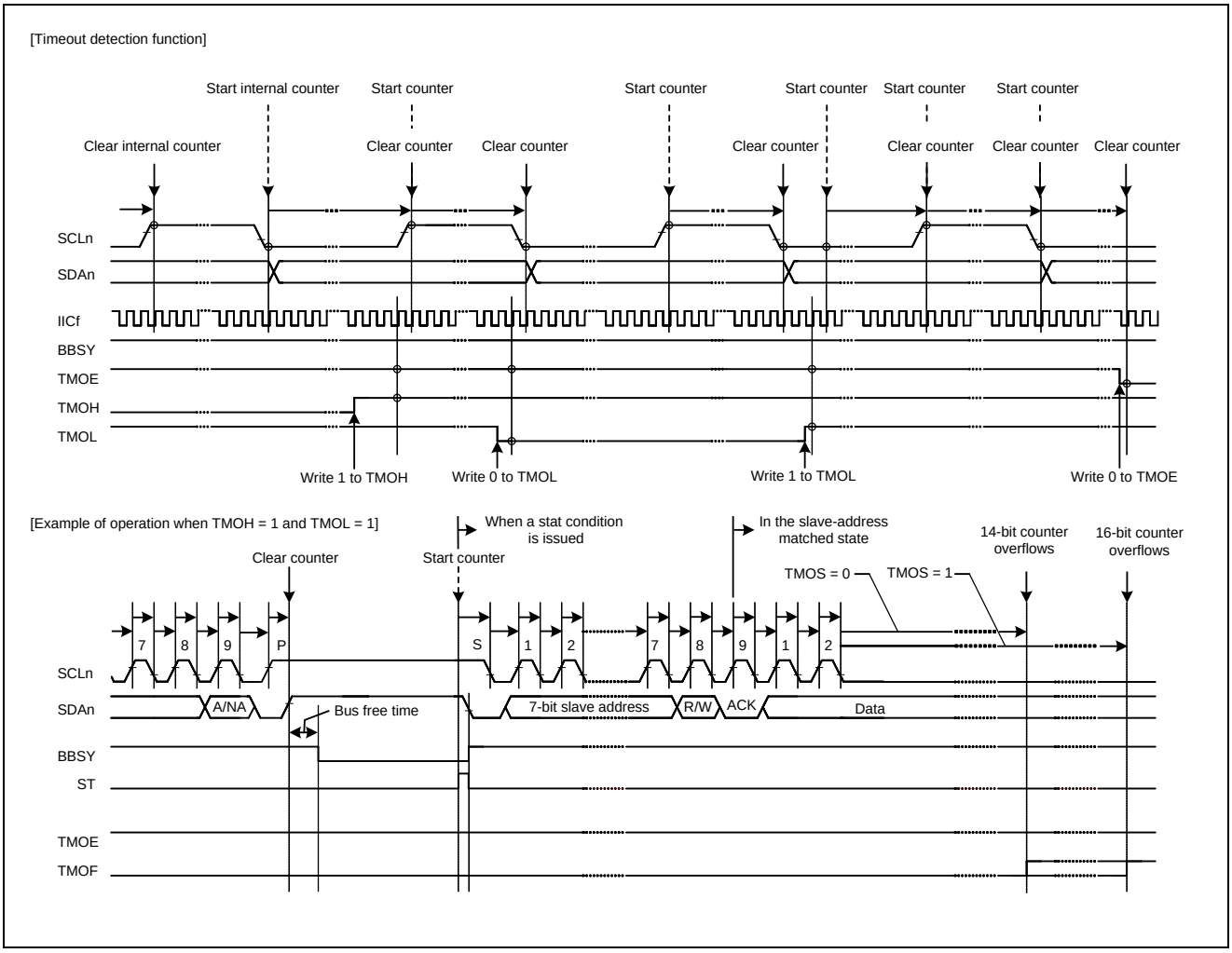



**Figure 22.38 Timeout Detection Function (TMOE, TMOS, TMOH, and TMOL Bits)**



Downloaded from Arrow.com.

### 22.11.2 Extra SCL Clock Cycle Output Function

In master mode, the RIIC module has a facility for the output of extra SCL (clock) cycles to release the SDAn line of the slave device from being held at the low level due to the master being out of synchronization with the slave device.

This function is mainly used in master mode to release the SDAn line of the slave device from the state of being fixed to the low level by including extra cycles of SCLn output from the RIIC with single cycles of the SCL (clock) signal as the unit in the case of a bus error where the RIIC cannot issue a stop condition because the slave device is holding the SDAn line at the low level. Do not use this facility in normal situations. Using it when communications are proceeding correctly will lead to malfunctions.

When the CLO bit in ICCR1 is set to 1 in master mode, a single cycle of the SCL clock at the frequency corresponding to the transfer rate settings (settings of the CKS[2:0] bits in ICMR1, and of the ICBRH and ICBRL registers) is output as an extra clock cycle. After output of this single cycle of the SCL clock, the CLO bit is automatically cleared to 0. Therefore, further extra clock cycles can be output consecutively by the software program writing 1 to the CLO bit after having read CLO = 0.

When the RIIC module is in master mode and the slave device is holding the SDAn line at the low level because synchronization with the slave device has been lost due to the effects of noise etc., the output of a stop condition is not possible. The facility for output of an extra cycle of the SCL (clock) signal can be used to output extra cycles of SCL one by one to make the slave device release the SDAn line from being held at the low level, thus recovering the bus from an unusable state. Release of the SDAn line by the slave device can be monitored by reading the SDAI bit in ICCR1. After confirming release of the SDAn line by the slave device, complete communications by reissuing the stop condition.

Use this facility with the MALE bit (master arbitration lost detection disabled) in ICFER cleared to 0. If the MALE bit is set to 1 (master arbitration lost detection enabled), arbitration is lost when the value of the SDAO bit in ICCR1does not match the state of the SDAn line, so take care on this point.

[Conditions for using the CLO bit in ICCR1]

- When the bus is free (BBSY flag in ICCR2 = 0) or in master mode (MST bit = 1 and BBSY flag = 1 in ICCR2)
- When the communication device does not hold the SCL line low

Figure 22.39 shows the operation timing of the extra SCL clock cycle output function.

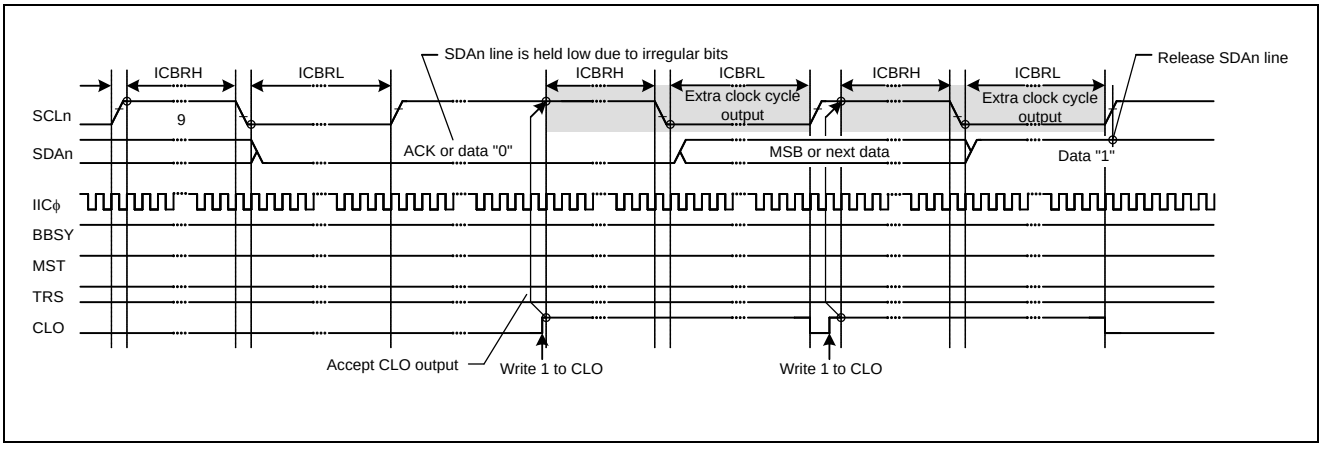



**Figure 22.39 Extra SCL Clock Cycle Output Function (CLO Bit)**



Downloaded from Arrow.com.

### 22.11.3 RIIC Reset and Internal Reset

The RIIC module incorporates a function for resetting itself. There are two types of reset. One is referred to as an RIIC reset; this initializes all registers including the BBSY flag in ICCR2. The other is referred to as an internal reset; this releases the RIIC from the slave-address matched state and initializes the internal counter while retaining other settings. After issuing a reset, be sure to clear the IICRST bit in ICCR1 to 0.

Both types of reset are effective for release from bus-hung states since both restore the output state of the SCLn and SDAn pins to the high impedance state.

Issuing a reset during slave operation may lead to a loss of synchronization between the master device clock and the slave device clock, so avoided this where possible. Note that monitoring of the bus state, such as for the presence of a start condition, is not possible during an RIIC reset (ICE and IICRST bits = 01b in ICCR1).

For a detailed description of the RIIC and internal resets, see section 22.14, Reset States.



Downloaded from Arrow.com.

## 22.12 SMBus Operation

The RIIC is available for data communication conforming to the SMBus (Version 2.0). To perform SMBus communication, set the SMBS bit in ICMR3 to 1 to select input level conforming to the SMBus for the SCLn pin/SDAn pin function. To use the transfer rate within a range of 10 kbps to 100 kbps of the SMBus standard, set the CKS[2:0] bits in ICMR1, ICBRH, and ICBRL. In addition, determine the values of the DLCS bit in ICMR2 and the SDDL[2:0] bits in ICMR2 to meet the data hold time specification of 300 ns or more. When the RIIC is used only as a slave device, the transfer rate setting is not necessary, whereas the ICBRL needs to be set to a value longer than the data setup time (250 ns).

For the SMBus device default address (1100 001b), use one of the slave address registers L0 to L2 (SARL0, SARL1, and SARL2), and set the corresponding FS bit (7-bit/10-bit address format select) in SARUy (y = 0 to 2) to 0 (7-bit address format).

When transmitting the UDID (Unique Device Identifier), set the SALE bit in ICFER to 1 to enable the slave arbitration lost detection function.

### 22.12.1 SMBus Timeout Measurement

#### (1) Measuring timeout of slave device

The following period (timeout interval: $T_{LOW:SEXT}$) must be measured for slave devices in SMBus communication.

- From start condition to stop condition

To measure timeout for slave devices, measure the period from start condition detection to stop condition detection with the TPU or TMR timer using a start condition detection interrupt (STI) and stop condition detection interrupt (SPI) of the RIIC. The measured timeout period must be within the total clock low-level period [slave device] $T_{LOW:SEXT}$: 25 ms (max.) of the SMBus standard.

If the time measured with the TPU or TMR exceeds the clock low-level detection timeout $T_{TIMEOUT}$: 25 ms (min.) of the SMBus standard, the slave device must release the bus by writing 1 to the IICRST bit in ICCR1 to issue an internal reset of the RIIC. When an internal reset is issued, the RIIC stops driving the bus for the SCLn pin and SDAn pin and make the SCLn/SDAn pin outputs high impedance, which releases the bus.

#### (2) Measuring timeout of master device

The following periods (timeout interval: $T_{LOW:MEXT}$) must be measured for master devices in SMBus communication.

- From start condition to acknowledge bit
- Between acknowledge bits
- From acknowledge bit to stop condition

To measure timeout for master devices, measure these periods with the TPU or TMR timer using a start condition detection interrupt (STI), stop condition detection interrupt (SPI), and transmit end interrupt (ICTEI) or receive data full interrupt (ICRXI) of the RIIC. The measured timeout period must be within the total clock low-level extended period [master device] $T_{LOW:MEXT}$: 10 ms (max.) of the SMBus standard, and the total of all $T_{LOW:MEXT}$ from start condition to stop condition must be within $T_{LOW:SEXT}$: 25 ms (max.).

For the ACK receive timing (rising edge of the ninth SMBCLK clock cycle), monitor the TEND flag in ICSR2 in master transmit mode (master transmitter) and the RDRF flag in ICSR2 in master receive mode (master receiver). For this reason, perform bytewise transmit operation in master transmit mode, and hold the RDRFS bit in ICMR3 0 until the byte just before reception of the final byte in master receive mode. While the RDRFS bit is 0, the RDRF flag is set to 1 at the rising edge of the ninth SMBCLK clock cycle.



Downloaded from Arrow.com.

If the period measured with the TPU or TMR exceeds the total clock low-level extended period [master device] $T_{LOW:MEXT}$: 10 ms (max.) of the SMBus standard or the total of measured periods exceeds the clock low-level detection timeout $T_{TIMEOUT}$: 25 ms (min.) of the SMBus standard, the master device must stop the transaction by issuing a stop condition. In master transmit mode, immediately stop the transmit operation (writing data to ICDRT).

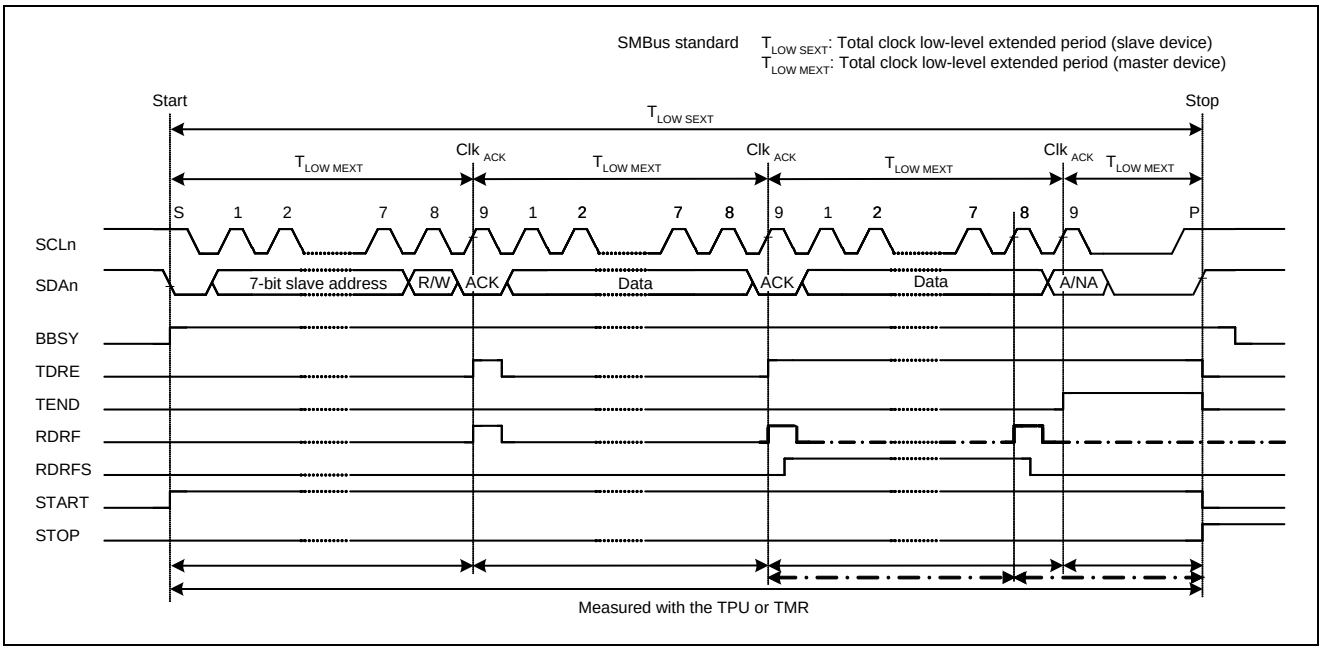



**Figure 22.40 SMBus Timeout Measurement**

## 22.12.2 Packet Error Code (PEC)

The RX610 Group incorporates a CRC operation circuit. The CRC operation circuit enables transmission of a packet error code (PEC) or checking the received data of the SMBus in data communication of the RIIC. For the CRC generating polynomials of the CRC operation circuit, see section 21, CRC Operation Circuit (CRC).

The PEC data in master transmit mode (master transmitter) can be generated by writing all transmit data to the CRC data input register (CRCDIR) in the CRC operation circuit.

The REC data in master receive mode (master receiver) can be checked by writing all receive data to CRCDIR in the CRC operation circuit and comparing the obtained value in the CRC data output register (CRCDOR) with the received PEC data.

To send ACK or NACK according to the match or mismatch result when the final byte is received as a result of the PEC code check, set the RDRFS bit in ICMR3 to 1 before the rising edge of the eighth SMBCLK clock cycle during reception of the final byte, and hold the SCLn line low at the falling edge of the eighth clock cycle.



Downloaded from Arrow.com.

### 22.12.3 SMBus Host Notification Protocol/Notify ARP Master

In communications over an SMBus, a slave device can temporarily act as a master device to notify the SMBus host (or ARP master) of (or request the SMBus host for) its own slave address or to request its own slave address from the SMBus host.

For a product of the RX610 Group to operate as an SMBus host (or ARP master), the host address (0001 000b) sent from the slave device must be detected as a slave address, so the RIIC has a function for detecting the host address. To detect the host address as a slave address, set the SMBS bit in ICMR3 and the HOAE bit in ICSER to 1. Operation after the host address has been detected is the same as normal slave operation.



Downloaded from Arrow.com.

## 22.13 Interrupt Request

The RIIC issues four types of interrupt request: transfer error or event generation (arbitration lost, NACK detection, timeout detection, start condition detection, and stop condition detection), receive data full, transmit data empty, and transmit end.

Table 22.7 shows details of the several interrupt requests. The receive data full and transmit data empty are both capable of launching data transfer by the DTC or DMAC.

**Table 22.7 Interrupt Sources**

| Abbreviation | Interrupt Request | Interrupt Flag | DTC Launching | DMAC Launching | Priority | Interrupt Condition |
|---|---|---|---|---|---|---|
| ICEEI | Transfer Error/ Event Generation | AL, NACKF, TMOF, START, STOP | Not poss ble | Not possible | High | AL=1 and AL E=1<br>NACKF=1 and NAKIE=1<br>TMOF=1 and TMOIE=1<br>START=1 and STIE=1<br>STOP=1 and SPIE=1 |
| ICRXI | Receive Data Full | — | Possible | Possible | | RDRF=1 and RIE=1 |
| ICTXI | Transmit Data Empty | — | Possible | Possible | | TDRE=1 and TIE=1 |
| ICTEI | Transmit End | TEND | Not poss ble | Not possible | Low | TEND=1 and TEIE=1 |

Clear or mask the various interrupt sources during interrupt handling.

Notes on interrupt processing:

1. There is a latency (delay) between the execution of a write instruction for a peripheral module by the CPU and actual writing to the module. Thus, when an interrupt flag has been cleared or masked, read the relevant flag again to check whether clearing or masking has been completed, and then return from interrupt processing. Returning from interrupt processing without checking that writing to the module has been completed creates a possibility of repeated processing of the same interrupt.
2. Since ICTXI is an edge-detected interrupted, it does not require clearing. Furthermore, the TDRE flag in ICSR2 (a condition for ICTXI) is automatically cleared to 0 when data for transmission are written to ICDRT or a stop condition is detected (STOP flag = 1 in ICSR2).
3. Since ICRXI is an edge-detected interrupted, it does not require clearing. Furthermore, the RDRF flag in ICSR2 (a condition for ICRXI) is automatically cleared to 0 when data are read out from ICDRR.
4. When using the ICTEI interrupt, clear the TEND flag in ICSR2 in the ICTEI interrupt processing, Note that the TEND flag in ICSR2 is automatically cleared to 0 when data for transmission are written to ICDRT or a stop condition is detected (STOP flag = 1 in ICSR2).



Downloaded from Arrow.com.

## 22.14 Reset States

The RIIC has chip reset, RIIC reset, and internal reset functions. Table 22.8 shows the scope of each reset and reset conditions.

**Table 22.8 Reset Conditions**

| | | Chip Reset | RIIC Reset (ICE = 0, IICRST = 1) | Internal Reset (ICE = 1, IICRST = 1) | Start Condition/ Restart Condition Detection | Stop Condition Detection |
|---|---|---|---|---|---|---|
| ICCR1 | ICE, IICRST | At a reset | Retained | Retained | Operation (retained) | Operation (retained) |
| | SCLO, SDAO | | At a reset | At a reset | | |
| | Others | | | Retained | | |
| ICCR2 | BBSY | At a reset | At a reset | Operation | Operation | Operation |
| | ST | | | At a reset | At a reset | Operation (retained) |
| | Others | | | | | At a reset |
| ICMR1 | BC[2:0] | At a reset | At a reset | At a reset | At a reset | Operation (retained) |
| | Others | | | Retained | Operation (retained) | |
| ICMR2 | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICMR3 | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICFER | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICSER | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICIER | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICSR1 | | At a reset | At a reset | At a reset | Operation (retained) | At a reset |
| ICSR2 | TDRE, TEND | At a reset | At a reset | At a reset | Operation (retained) | At a reset |
| | START | | | | Operation | |
| | STOP | | | | Operation (retained) | Operation |
| | Others | | | | | Operation (retained) |
| SARL0 to SARL2 SARU0 to SARU2 | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICBRH, ICBRL | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICDRT | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICDRR | | At a reset | At a reset | Retained | Operation (retained) | Operation (retained) |
| ICDRS | | At a reset | At a reset | At a reset | Operation (retained) | Operation (retained) |
| Timeout detection function | | At a reset | At a reset | Operation | Operation | Operation |
| Bus free time measurement | | At a reset | At a reset | Operation | Operation | Operation |



Downloaded from Arrow.com.

## 22.15 Usage Notes

### 22.15.1 Setting Module Stop Function

Module stop state can be entered or canceled using module stop control register B (MSTPCRB). The initial setting is for operation of the RIIC to be halted. RIIC register access is enabled by clearing module stop state.
For details of module stop control register B, see section 8, Low Power Consumption.

### 22.15.2 Setting Input Buffer Control Register

Input to peripheral modules can be enabled or disabled using the input buffer control register (Pn.ICR). The initial setting is for input to the RIIC to be disabled.
As the SCL and SDA lines on the I²C bus are bidirectional, the SCLn and SDAn pins of the RIIC are input/output pins. Make appropriate settings in the input buffer control bits in the P1.ICR corresponding to the SCLn and SDAn pins of the RIIC to enable input to the RIIC. If the required input is disabled, the RIIC cannot detect start conditions (including restart conditions) or stop conditions, or count the SCL clock cycles.
For details of the input buffer control register, see section 14, I/O Ports.

### 22.15.3 Timings for Writing and Outputting of Transmit Acknowledge Bit

The transmit acknowledge (ICMR3.ACKBT) bit value at the falling edge of the eighth SCL clock cycle is always output regardless of the setting in the RDRF flag set timing selection (ICMR3.RDRFS) bit. The ACKBT value written after the falling edge of the eighth SCL clock cycle is output at the falling edge of the eighth clock cycle of the next frame.

### 22.15.4 Restrictions on Timings for Stop Condition Issuance Request and Transmit Data Writing in Master Transmitter Mode

In master transmitter mode, when the low-level output in the ninth clock cycle ends at the same time as the stop condition issuance request (writing 1 to the ICCR2.SP bit) while the transmit end (ICSR2.TEND) flag is 1, if a sharp rise occurs on the SCL line due to the external pull-up resistance, the generated waveforms may look the same as the start condition waveforms depending on the timing relationship between the SCL sharp rise and the fall on the SDA line by the stop condition issuance. In this case, reexamine the external pull-up resistance or send a stop condition issuance request after waiting for the end of the low-level output in the ninth clock cycle.
Likewise, in maser transmitter mode, when the low-level output in the ninth clock cycle ends at the same time as writing of transmit data (ICDRT) while the transmit end (ICSR2.TEND) flag is 1, a momentary low signal may be output on the SDA line and the generated waveforms may look the same as the stop condition waveforms. In master transmitter mode, be sure to write transmit data while the transmit end flag is 0; if the transmit end flag is 1, write transmit data after waiting for the end of the low-level output in the ninth clock cycle.



Downloaded from Arrow.com.

### 22.15.5 Notes when Communication is Restarted with the NACK Reception in Master Mode

When NACK is received from the slave device in master transmission mode (ICMR3.ACKBR = 1), be sure to end the communication once by issuing the stop condition, and then restart the communication by issuing the start condition.

As the RIIC transmission is a buffer operation, if the communication continues without issuing the stop condition to end the communication but with issuing the restart condition, the transmission data in ICDRT that has not been transmitted might be output. To restart the communication after the NACK is received, make sure to end the communication once by issuing the stop condition, and then restart the communication by issuing the start condition.

### 22.15.6 Notes on the RDRF Flag Set Timing Selection Bit (RDRFS)

In slave mode, slave addresses are matched while the RDRF flag set timing selection bit (RDRFS bit) is 1 (the receive data full flag (RDRF flag) is set to 1 at the eighth rising cycle of the SCL clock). In such a case, if the communication is not ended by the stop condition and the restart condition specifies the slave addresses (SAR0 to SAR2) of the RIIC again, the RDRF flag is set to 1 at the ninth rising cycle of the SCL soon after the flag is set to 1 at the eighth rising cycle of the SCL.



Downloaded from Arrow.com.

# 23. A/D Converter

## 23.1 Overview

The RX610 Group includes four successive approximation type 10-bit A/D converters (units 0 to 3). Each unit allows up to four analog input channels to be selected.

The A/D converter has two kinds of operating modes, that are single mode which converts the analog input of the specified single channel for only once and scan mode which continuously converts the analog inputs of the specified channels up to four.

Table 23.1 lists the specifications of the A/D converter and the table 23.2 indicates the comparison of functions by each unit. Figures 23.1 to 23.4 show block diagrams of the A/D converter units 0 to 4, respectively.

**Table 23.1 Specifications of A/D Converter**

| Item | Specifications |
|---|---|
| Number of units | Four units |
| Input channels | Each unit: 4 channels (total 16 channels) |
| A/D conversion method | Successive approximation method |
| Resolution | 10 bits |
| Conversion time | 1.0 μs per 1 channel (when operating peripheral module clock PCLK = 50 MHz) |
| A/D conversion clock | 4 types: PCLK, PCLK/2, PCLK/4, PCLK/8 |
| Operating modes | • Single mode: A/D conversion is to be performed for only once on the analog input of the specified single channel.<br>• Scan mode<br>Continuous scan mode: A/D conversion is to be performed sequentially on the analog inputs of the specified channels up to four.<br>Single scan mode: A/D conversion is to be performed for one cycle on the analog inputs of the specified channels up to four. |
| Conditions of A/D conversion start | • Software trigger<br>• Conversion start trigger by the 16-bit timer pulse unit (TPU) or 8-bit timer (TMR).<br>• External trigger<br>A/D conversion for each unit can be externally triggered from the ADTRGn# pin.<br>For units 0 and 1, an external trigger can be simultaneously started from the ADTRG0# pin.<br>For units 2 and 3, an external trigger can be simultaneously started from the ADTRG2# pin. |
| Function | • Sample-and-hold function<br>• Number of sampling state is adjustable. |
| Interrupt source | • A/D conversion end interrupt (ADI) request can be generated by each unit.<br>• An ADI interrupt can activate data transfer controller (DTC) or DMA controller (DMAC). |
| Power-down function | Module stop state can be set for each unit. |



Downloaded from Arrow.com.

**Table 23.2 Comparison of Functions by Each Unit**

| Item | | | Unit 0 (AD0) | Unit 1 (AD1) | Unit 2 (AD2) | Unit 3 (AD3) |
|---|---|---|---|---|---|---|
| Analog input channel | | | AN0<br>AN1<br>AN2<br>AN3 | AN4<br>AN5<br>AN6<br>AN7 | AN8<br>AN9<br>AN10<br>AN11 | AN12<br>AN13<br>AN14<br>AN15 |
| A/D conversion start conditions*1 | Software | Software trigger | √ | √ | √ | √ |
| | External trigger | ADTRG0# | √ | √ | — | — |
| | | ADTRG1# | — | √ | — | — |
| | | ADTRG2# | — | — | √ | √ |
| | | ADTRG3# | — | — | — | √ |
| | Timer trigger | Compare match/ input capture from TPU0 | TGRA | TGRB | TGRC | TGRD |
| | | Compare match/ input capture from TPU0 to TPU5 | TGRA | TGRA | TGRA | TGRA |
| | | Compare match/ input capture from TPU6 to TPU11 | TGRA | TGRA | TGRA | TGRA |
| | | Compare match from TMR0 | Compare match A | Compare match A | — | — |
| | | Compare match from TMR2 | — | — | Compare match A | Compare match A |
| Interrupt | | | ADI0 | ADI1 | ADI2 | ADI3 |
| Module stop function setting*2 | | | MSTPCRA. MSTPA23 bit | MSTPCRA. MSTPA22 bit | MSTPCRA. MSTPA21 bit | MSTPCRA. MSTPA20 bit |

[Legend]

√: Enabled

—: Disabled

Notes: 1. A/D conversion start conditions can be selected by each unit.

2. For details, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

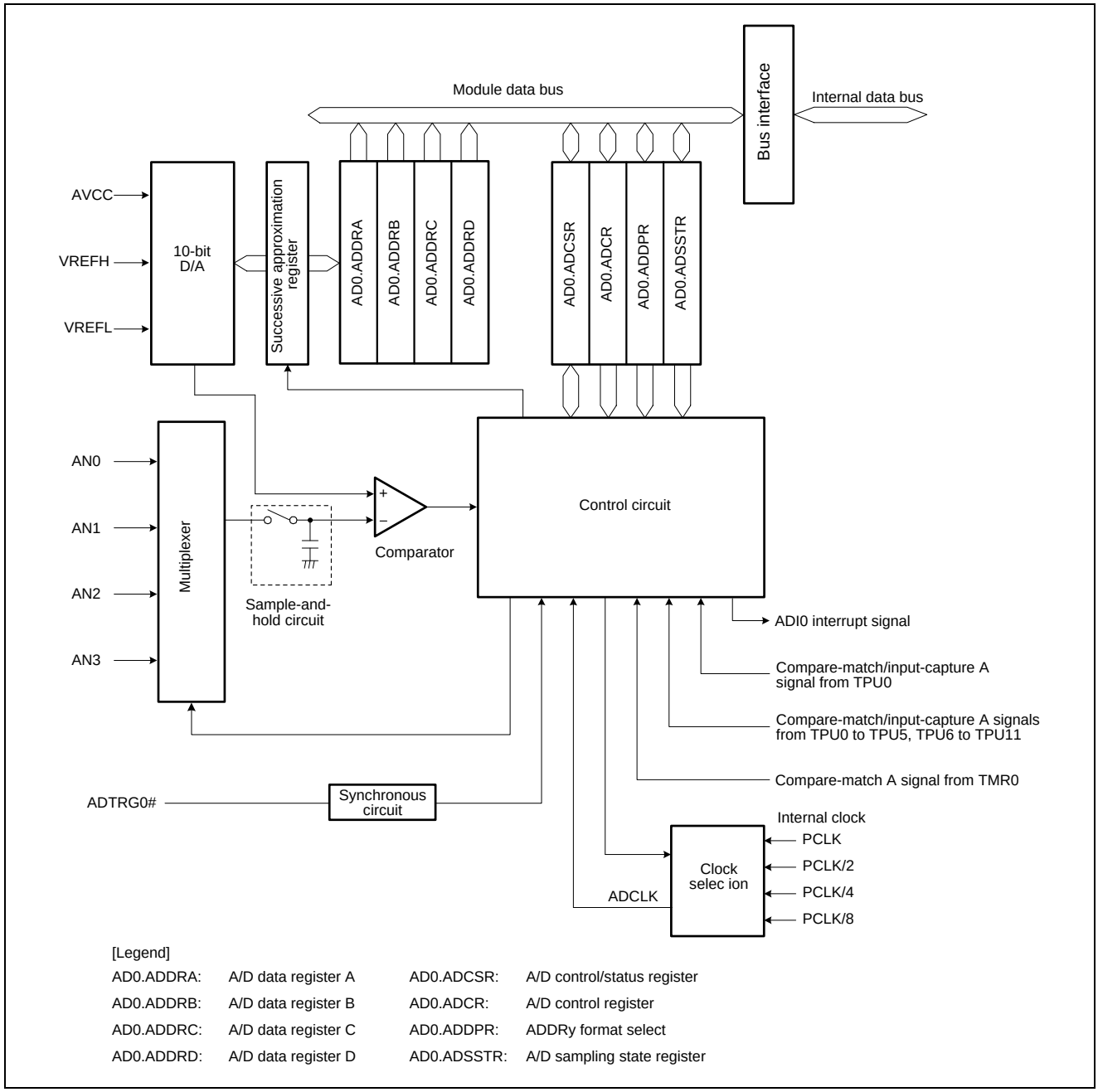



**Figure 23.1 Block Diagram of A/D Converter Unit 0 (AD0)**



Downloaded from Arrow.com.

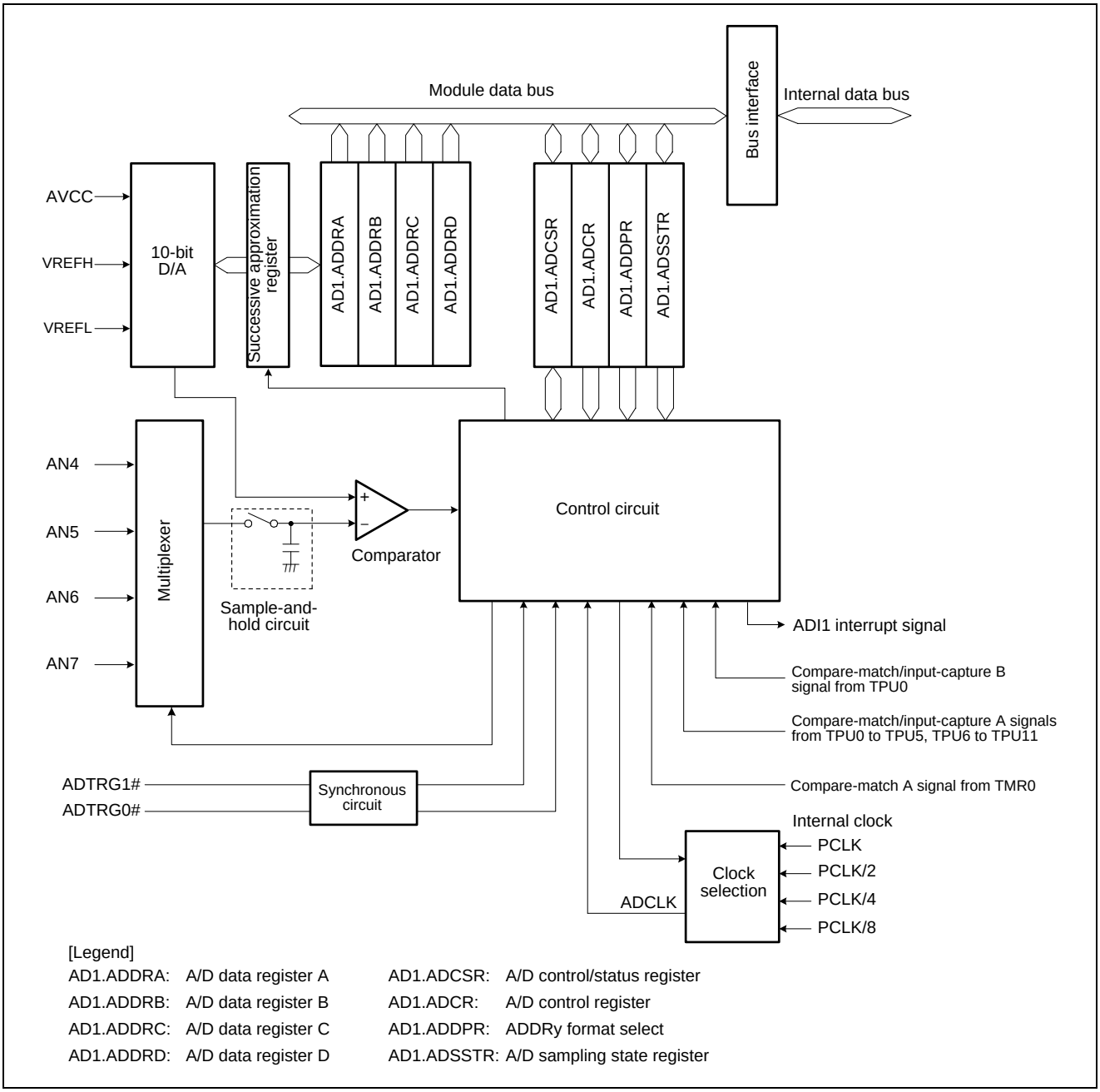



**Figure 23.2 Block Diagram of A/D Converter Unit 1 (AD1)**



Downloaded from Arrow.com.

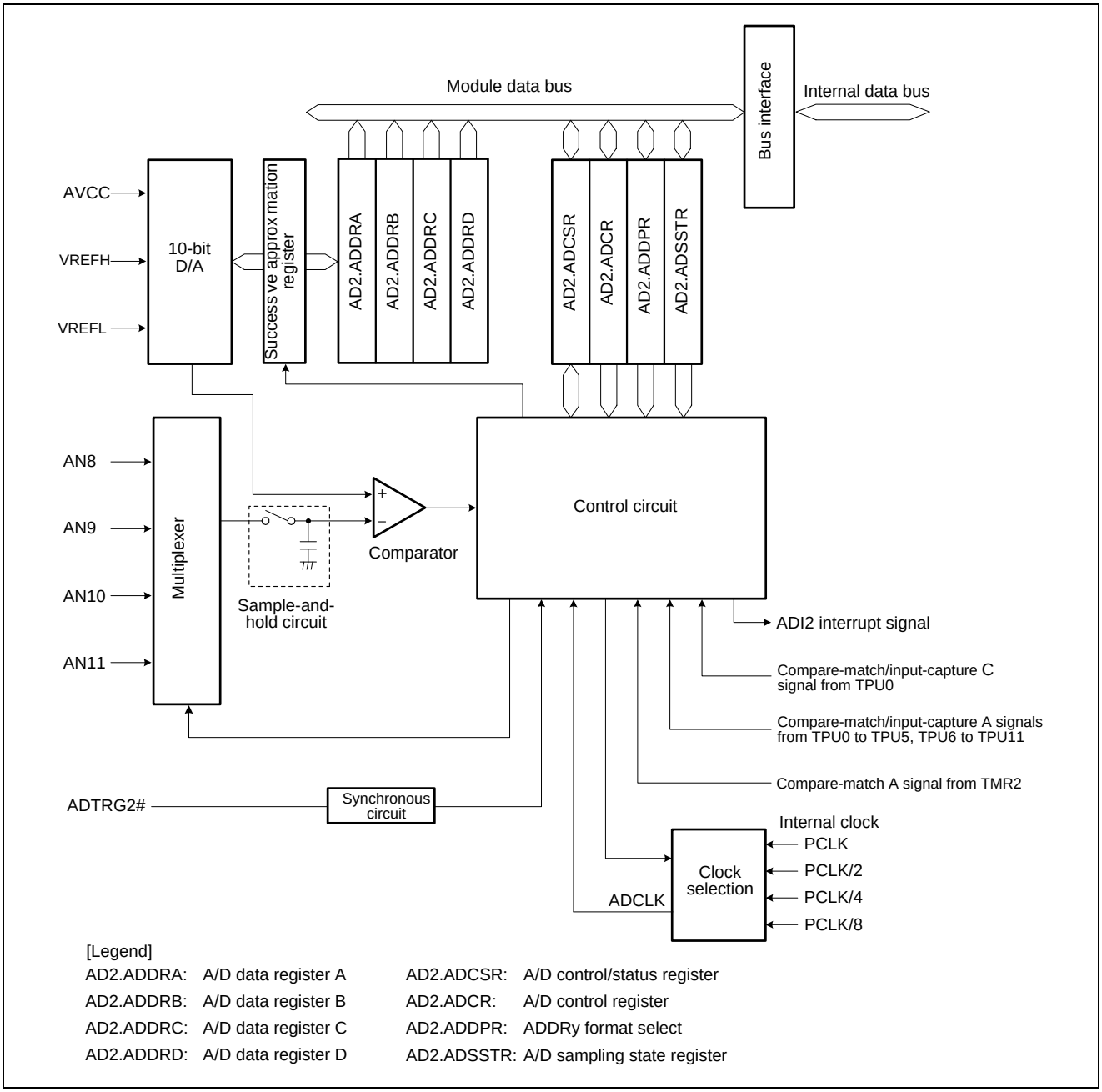



**Figure 23.3 Block Diagram of A/D Converter Unit 2 (AD2)**



Downloaded from Arrow.com.

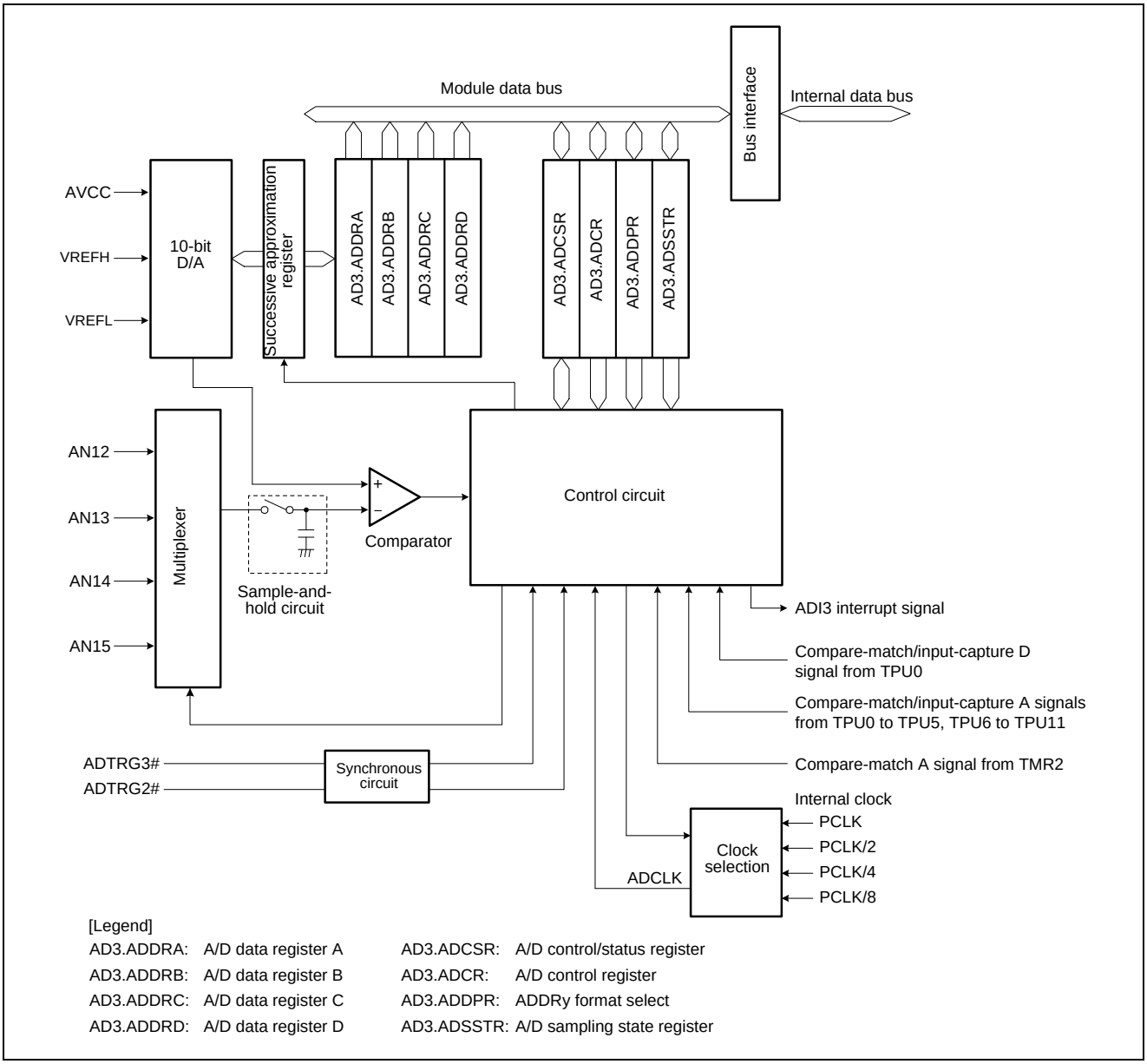



**Figure 23.4 Block Diagram of A/D Converter Unit 3 (AD3)**



Downloaded from Arrow.com.

Table 23.3 indicates the input pins of the A/D converter.

**Table 23.3 Input Pins of A/D Converter**

| Unit | Module Symbol | Pin Name | Input | Function |
|---|---|---|---|---|
| 0 | AD0 | AN0 to AN3 | Input | Analog input pins |
| | | ADTRG0# | Input | External trigger input pin for starting A/D conversion |
| 1 | AD1 | AN4 to AN7 | Input | Analog input pins |
| | | ADTRG0# | Input | External trigger input pin for starting A/D conversion |
| | | ADTRG1# | Input | External trigger input pin for starting A/D conversion |
| 2 | AD2 | AN8 to AN11 | Input | Analog input pins |
| | | ADTRG2# | Input | External trigger input pin for starting A/D conversion |
| 3 | AD3 | AN12 to AN15 | Input | Analog input pins |
| | | ADTRG2# | Input | External trigger input pin for starting A/D conversion |
| | | ADTRG3# | Input | External trigger input pin for starting A/D conversion |
| Common | | $AV_{CC}$ | Input | Analog circuit power supply pin |
| | | $AV_{SS}$ | Input | Analog circuit ground pin |
| | | $V_{REFH}$ | Input | A/D converter reference power supply pin |
| | | $V_{REFL}$ | Input | A/D converter reference ground pin<br>Connect this pin to the analog reference power supply (0 V). |



Downloaded from Arrow.com.

## 23.2 Register Descriptions

Table 23.4 lists the registers of the A/D converter.

**Table 23.4 Registers of A/D Converter**

| Unit | Module Symbol | Register Name | Register Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| 0 | AD0 | A/D data register A | ADDRA | 0000h | 0008 8040h | 16 |
| | | A/D data register B | ADDRB | 0000h | 0008 8042h | 16 |
| | | A/D data register C | ADDRC | 0000h | 0008 8044h | 16 |
| | | A/D data register D | ADDRD | 0000h | 0008 8046h | 16 |
| | | A/D control/status register | ADCSR | x0h | 0008 8050h | 8 |
| | | A/D control register | ADCR | 00h | 0008 8051h | 8 |
| | | ADDRy format select register | ADDPR | 00h | 0008 8052h | 8 |
| | | A/D sampling state register | ADSSTR | 19h | 0008 8053h | 8 |
| 1 | AD1 | A/D data register A | ADDRA | 0000h | 0008 8060h | 16 |
| | | A/D data register B | ADDRB | 0000h | 0008 8062h | 16 |
| | | A/D data register C | ADDRC | 0000h | 0008 8064h | 16 |
| | | A/D data register D | ADDRD | 0000h | 0008 8066h | 16 |
| | | A/D control/status register | ADCSR | x0h | 0008 8070h | 8 |
| | | A/D control register | ADCR | 00h | 0008 8071h | 8 |
| | | ADDRy format select register | ADDPR | 00h | 0008 8072h | 8 |
| | | A/D sampling state register | ADSSTR | 19h | 0008 8073h | 8 |
| 2 | AD2 | A/D data register A | ADDRA | 0000h | 0008 8080h | 16 |
| | | A/D data register B | ADDRB | 0000h | 0008 8082h | 16 |
| | | A/D data register C | ADDRC | 0000h | 0008 8084h | 16 |
| | | A/D data register D | ADDRD | 0000h | 0008 8086h | 16 |
| | | A/D control/status register | ADCSR | x0h | 0008 8090h | 8 |
| | | A/D control register | ADCR | 00h | 0008 8091h | 8 |
| | | ADDRy format select register | ADDPR | 00h | 0008 8092h | 8 |
| | | A/D sampling state register | ADSSTR | 19h | 0008 8093h | 8 |
| 3 | AD3 | A/D data register A | ADDRA | 0000h | 0008 80A0h | 16 |
| | | A/D data register B | ADDRB | 0000h | 0008 80A2h | 16 |
| | | A/D data register C | ADDRC | 0000h | 0008 80A4h | 16 |
| | | A/D data register D | ADDRD | 0000h | 0008 80A6h | 16 |
| | | A/D control/status register | ADCSR | x0h | 0008 80B0h | 8 |
| | | A/D control register | ADCR | 00h | 0008 80B1h | 8 |
| | | ADDRy format select register | ADDPR | 00h | 0008 80B2h | 8 |
| | | A/D sampling state register | ADSSTR | 19h | 0008 80B3h | 8 |



Downloaded from Arrow.com.

### 23.2.1 A/D Data Register y (ADDRy) (y = A to D)

Addresses: AD0.ADDRA 0008 8040h, AD0.ADDRB 0008 8042h, AD0.ADDRC 0008 8044h, AD0.ADDRD 0008 8046h
AD1.ADDRA 0008 8060h, AD1.ADDRB 0008 8062h, AD1.ADDRC 0008 8064h, AD1.ADDRD 0008 8066h
AD2.ADDRA 0008 8080h, AD2.ADDRB 0008 8082h, AD2.ADDRC 0008 8084h, AD2.ADDRD 0008 8086h
AD3.ADDRA 0008 80A0h, AD3.ADDRB 0008 80A2h, AD3.ADDRC 0008 80A4h, AD3.ADDRD 0008 80A6h

ADDPR.DPSEL bit = 0 (Data padded at the LSB end)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | | | | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADDPR.DPSEL bit = 1 (Data padded at the MSB end)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADDRy registers are 16-bit read-only registers, which store an A/D conversion result for each channel.

Table 23.5 lists the analog input channels and corresponding ADDRy registers.

10-bit data can be relocated by setting the DPSEL bit in ADDPR.

Bits "—" are always read as 0. The write value should always be 0.

**Table 23.5 Analog Input Channels and Corresponding ADDRy Registers**

| Analog Input Channel | ADDRy Register |
|---|---|
| AN0 | AD0.ADDRA |
| AN1 | AD0.ADDRB |
| AN2 | AD0.ADDRC |
| AN3 | AD0.ADDRD |
| AN4 | AD1.ADDRA |
| AN5 | AD1.ADDRB |
| AN6 | AD1.ADDRC |
| AN7 | AD1.ADDRD |
| AN8 | AD2.ADDRA |
| AN9 | AD2.ADDRB |
| AN10 | AD2.ADDRC |
| AN11 | AD2.ADDRD |
| AN12 | AD3.ADDRA |
| AN13 | AD3.ADDRB |
| AN14 | AD3.ADDRC |
| AN15 | AD3.ADDRD |



Downloaded from Arrow.com.

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

Addresses: AD0.ADCSR 0008 8050h, AD1.ADCSR 0008 8070h
AD2.ADCSR 0008 8090h, AD3.ADCSR 0008 80B0h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | — | ADIE | ADST | — | CH[3:0] | | | |
| Value after reset: | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[Legend] x: Undefined

| Bit | Symbol | Bit Name | Description | | | R/W |
|---|---|---|---|---|---|---|
| b3 to b0 | CH[3:0] | Channel Select* | Unit | Single mode (ADCR.MODE[1:0] = 00b) | Scan mode (ADCR.MODE[1:0] = 10b or 11b) | R/W |
| | | | Unit 0 | b3 b0<br>0 0 0 0: AN0<br>0 0 0 1: AN1<br>0 0 1 0: AN2<br>0 0 1 1: AN3<br>Settings other than above are prohibited. | b3 b0<br>0 0 0 0: AN0<br>0 0 0 1: AN0 and AN1<br>0 0 1 0: AN0 to AN2<br>0 0 1 1: AN0 to AN3<br>Settings other than above are prohibited. | |
| | | | Unit 1 | b3 b0<br>0 0 0 0: AN4<br>0 0 0 1: AN5<br>0 0 1 0: AN6<br>0 0 1 1: AN7<br>Settings other than above are prohibited. | b3 b0<br>0 0 0 0: AN4<br>0 0 0 1: AN4 and AN5<br>0 0 1 0: AN4 to AN6<br>0 0 1 1: AN4 to AN7<br>Settings other than above are prohibited. | |
| | | | Unit 2 | b3 b0<br>0 0 0 0: AN8<br>0 0 0 1: AN9<br>0 0 1 0: AN10<br>0 0 1 1: AN11<br>Settings other than above are prohibited. | b3 b0<br>0 0 0 0: AN8<br>0 0 0 1: AN8 and AN9<br>0 0 1 0: AN8 to AN10<br>0 0 1 1: AN8 to AN11<br>Settings other than above are prohibited. | |
| | | | Unit 3 | b3 b0<br>0 0 0 0: AN12<br>0 0 0 1: AN13<br>0 0 1 0: AN14<br>0 0 1 1: AN15<br>Settings other than above are prohibited. | b3 b0<br>0 0 0 0: AN12<br>0 0 0 1: AN12 and AN13<br>0 0 1 0: AN12 to AN14<br>0 0 1 1: AN12 to AN15<br>Settings other than above are prohibited. | |
| b4 | — | Reserved | This bit is always read as 0. The write value should always be 0. | | | R/W |
| b5 | ADST | A/D Start | 0: Stops A/D conversion<br>1: Starts A/D conversion | | | R/W |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b6 | ADIE | A/D Interrupt Enable | 0: ADI interrupt is disabled by completing A/D conversion<br>1: ADI interrupt is enabled by completing A/D conversion | R/W |
| b7 | — | Reserved | This bit is always read as an undefined value. The write value should always be 1. | R/W |

Note: * The Pm.DDR.Bj bit for the analog input should be set to 0 (input port) and the Pm.ICR.Bj (m = 4 or 9, j = 0 to 7) bit should be set to 0 (disabling the input buffer and fixing the input signal to the high level). For details, see section 14, I/O Ports.

ADCSR controls the A/D conversion operations.

### CH[3:0] Bits (Channel Select)

These bits select analog input channels that allow A/D conversion.

- Single mode (ADCR.MODE[1:0] bits = 00b)
  Select the single analog input channel that allows A/D.
- Scan mode (ADCR.MODE[1:0] bits = 10b or 11b)
  Select analog input channels up to 4 that allow A/D conversion.

### ADST Bit (A/D Start)

The ADST bit starts/stops A/D conversion.

Before setting the ADST bit to 1, complete the setting for A/D conversion clock and the operation mode.

[Setting conditions]

- When 1 is written by software
- Detection of the trigger selected by the ADCR.TRGS [2:0] bits

[Clearing conditions]

- When 0 is written by software
- The A/D conversion is completed in single mode
- The A/D conversion is completed on every selected channel in single scan mode.

### ADIE Bit (A/D Interrupt Enable)

The ADIE bit enables/disables the A/D conversion end interrupt (ADI).



Downloaded from Arrow.com.

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

Addresses: AD0.ADCR 0008 8051h, AD1.ADCR 0008 8071h
AD2.ADCR 0008 8091h, AD3.ADCR 0008 80B1h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | TRGS[2:0] | | | – | CKS[1:0] | | MODE[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | | R/W |
|---|---|---|---|---|---|
| b1, b0 | MODE[1:0] | Operation Mode Select | b1 b0<br>0 0: Single mode<br>0 1: Setting prohibited<br>1 0: Continuous scan mode<br>1 1: Single scan mode | | R/W |
| b3, b2 | CKS[1:0] | Clock Select | b3 b2<br>0 0: PCLK/8<br>0 1: PCLK/4<br>1 0: PCLK/2<br>1 1: PCLK | | R/W |
| b4 | — | Reserved | This bit is always read as 0. The write value should always be 0. | | R/W |
| b7 to b5 | TRGS[2:0] | Trigger Select | Unit | Trigger signal | R/W |
| | | | Unit 0 | b7 b6 b5<br>0 0 0: Software trigger<br>0 0 1: Compare-match/input-capture A signals from TPU0 to TPU5<br>0 1 0: Compare-match A signal from TMR0<br>0 1 1: Trigger from ADTRG0#*<br>1 0 0: Compare-match/input-capture A signal from TPU0<br>1 0 1: Compare-match/input-capture A signals from TPU6 to TPU11<br>1 1 0: Setting prohibited<br>1 1 1: Setting prohibited | |
| | | | Unit 1 | b7 b6 b5<br>0 0 0: Software trigger<br>0 0 1: Compare-match/input-capture A signals from TPU0 to TPU5<br>0 1 0: Compare-match A signal from TMR0<br>0 1 1: Trigger from ADTRG1#*<br>1 0 0: Compare-match/input-capture B signal from TPU0<br>1 0 1: Compare-match/input-capture A signals from TPU6 to TPU11<br>1 1 0: Setting prohibited<br>1 1 1: Trigger from ADTRG0#* | |



Downloaded from Arrow.com.

| Bit | Symbol | Bit Name | Description | | R/W |
|---|---|---|---|---|---|
| b7 to b5 | TRGS[2:0] | Trigger Select | Unit | Trigger signal | R/W |
| | | | Unit 2 | b7 b6 b5 | |
| | | | | 0 0 0: Software trigger | |
| | | | | 0 0 1: Compare-match/input-capture A signals from TPU0 to TPU5 | |
| | | | | 0 1 0: Compare-match A signal from TMR2 | |
| | | | | 0 1 1: Trigger from ADTRG2#* | |
| | | | | 1 0 0: Compare-match/input-capture C signal from TPU0 | |
| | | | | 1 0 1: Compare-match/input-capture A signals from TPU6 to TPU11 | |
| | | | | 1 1 0: Setting prohibited | |
| | | | | 1 1 1: Setting prohibited | |
| | | | Unit 3 | b7 b6 b5 | |
| | | | | 0 0 0: Software trigger | |
| | | | | 0 0 1: Compare-match/input-capture A signals from TPU0 to TPU5 | |
| | | | | 0 1 0: Compare-match A signal from TMR2 | |
| | | | | 0 1 1: Trigger from ADTRG3#* | |
| | | | | 1 0 0: Compare-match/input-capture D signal from TPU0 | |
| | | | | 1 0 1: Compare-match/input-capture A signals from TPU6 to TPU11 | |
| | | | | 1 1 0: Setting prohibited | |
| | | | | 1 1 1: Trigger from ADTRG2#* | |

Note: * To start the A/D conversion by the ADTRGn# (n = 0 to 3) pin, the Pm.DDR.Bj bit should be set to 0 (input port) and the Pm.ICR.Bj (m = 1 or 7, j = 0, 3, 4, or 7) bit should be set to 1 (input buffer for the corresponding pin is valid). For details, see section 14, I/O Ports.

ADCR enables setting for an A/D conversion start trigger, an operating mode, and A/D conversion clock mode.
Set ADCR while the ADST bit in ADCSR is 0.

### MODE[1:0] Bits (Operating Mode Select)

These bits select the A/D conversion operation mode.

### CKS[1:0] Bits (Clock Select)

These bits set the frequency of the A/D conversion clock (ADCLK) and thus select the A/D conversion time.

Set the frequency of ADCLK to 4 MHz or higher.

For details, see section 23.3.3, Input Sampling and A/D Conversion Time.

### TRGS[2:0] Bits (Trigger Select)

These bits select the A/D conversion start trigger.



Downloaded from Arrow.com.

### 23.2.4 ADDRy Format Select Register (ADDPR)

Addresses: AD0.ADDPR 0008 8052h, AD1.ADDPR 0008 8072h
AD2.ADDRR 0008 8092h, AD3.ADDPR 0008 80B2h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DPSEL | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b6 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DPSEL | ADDRy Format Select | 0: A/D data is padded at the LSB end.<br>1: A/D data is padded at the MSB end. | R/W |

ADDPR selects the placement of data in the A/D data registers.

#### DPSEL Bit (ADDRy Format Select)

The DPSEL bit selects whether data in the A/D data registers is padded at the LSB or MSB end.

### 23.2.5 A/D Sampling State Register (ADSSTR)

Addresses: AD0.ADSSTR 0008 8053h, AD1.ADSSTR 0008 8073h
AD2.ADSSTR 0008 8093h, AD3.ADSSTR 0008 80B3h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

ADSSTR is an 8-bit readable/writable register that is used to set the sampling time for analog inputs.

Sampling time is adjustable when the signal source impedance of analog input is high and the sampling time is insufficient or the speed of peripheral module clock (PCLK) is low.

Set the value of 02h or larger.

Ensure to rewrite this register while the A/D conversion is stopped (the ADST bit in ADCSR = 0) in order to prevent incorrect operation.

For details, see section 23.3.3, Input Sampling and A/D Conversion Time.



Downloaded from Arrow.com.

## 23.3 Operation

The RX610 Group includes four units of A/D converter and the each unit has a same feature.

Definitions of a single unit are given below.

The A/D converter has two operating modes: single mode and scan mode.

In single mode, A/D conversion is to be performed for only once on the analog input of the specified single channel.

In scan mode, A/D conversion is to be performed sequentially on the analog inputs of the specified channels up to four. Two types of scan mode are provided, that is, continuous san mode where A/D conversion is repeatedly performed and single scan mode where A/D conversion is performed for the specified channels for one cycle.

### 23.3.1 Single Mode

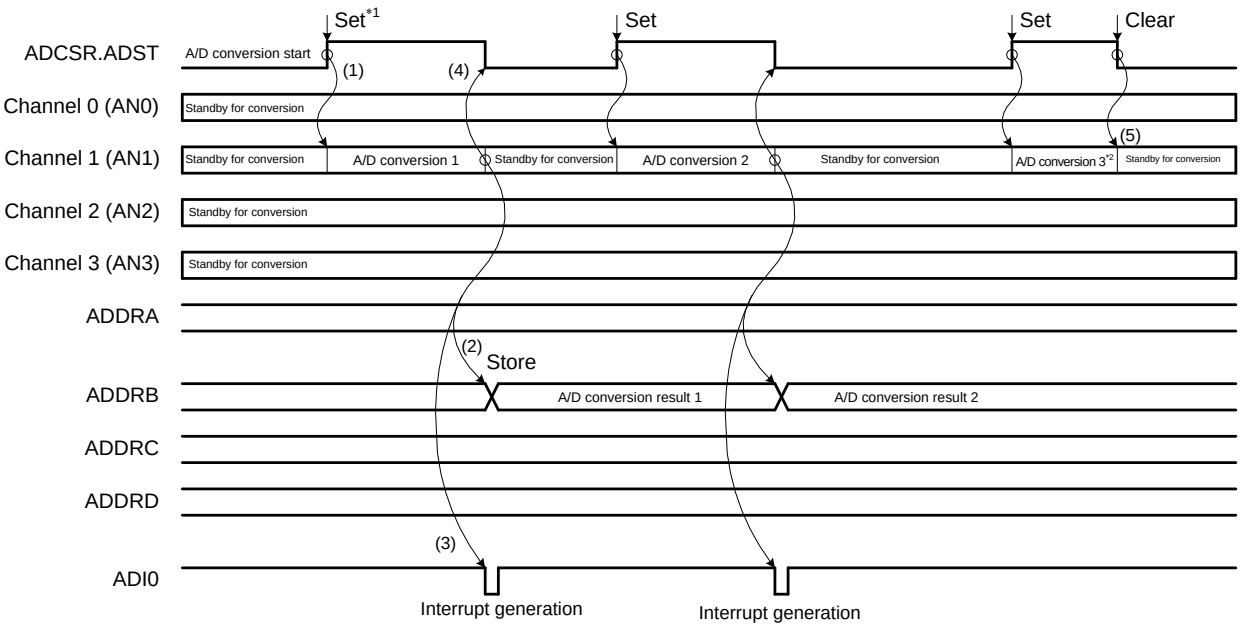
In single mode, A/D conversion is to be performed for only once on the analog input of the specified single channel as below.

1. A/D conversion for the selected channel is started when the ADST bit in ADCSR is set to 1 (A/D conversion start) by software, TPU, TMR, or an external trigger input.
2. When A/D conversion is completed, the A/D conversion result is stored into the corresponding A/D data register (ADDRn) of the channel.
3. When A/D conversion is completed, if the ADIE bit in ADCSR is set to 1 (A/D interrupt enable by completing A/D conversion), an ADI interrupt request is generated.
4. The ADST bit remains at 1 during A/D conversion, and is automatically cleared to 0 when A/D conversion ends. Then the A/D converter enters a wait state.
5. If the ADST bit is cleared to 0 during A/D conversion (A/D conversion stop), A/D conversion stops and the A/D converter enters a wait state.

Figure 23.5 shows an example of operation when AN1 is selected as an analog input.



Downloaded from Arrow.com.




Notes:
1. ↓ indicates an instruction execution by software.
2. Data during conversion is ignored.

**Figure 23.5 Example of A/D Converter Operation (Single Mode)**



Downloaded from Arrow.com.

### 23.3.2 Scan Mode

In scan mode, A/D conversion is to be performed sequentially on the analog inputs of the specified channels up to four. Two types of scan mode are provided, that is, continuous scan mode where A/D conversion is repeatedly performed and single scan mode where A/D conversion is performed for the specified channels for one cycle.

#### 23.3.2.1 Continuous Scan Mode

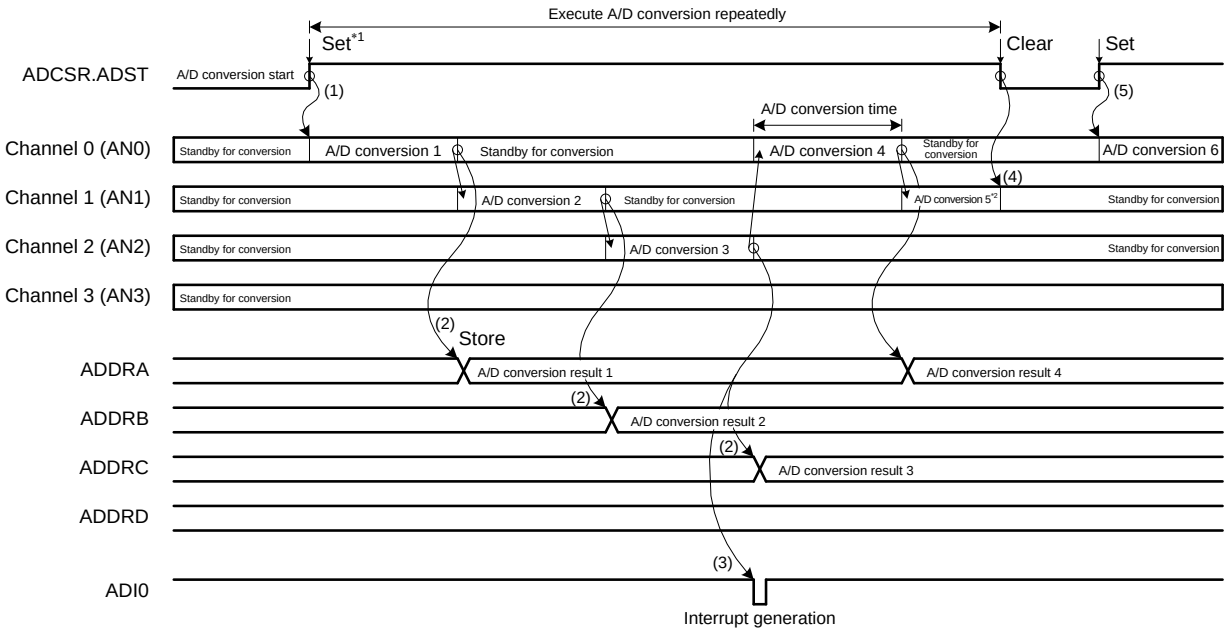
In continuous scan mode, A/D conversion is to be performed sequentially on the analog inputs of the specified channels as below.

1. When the ADST bit in ADCSR is set to 1 (A/D conversion start) by software, TPU, TMR, or an external trigger input, A/D conversion starts on the first channel in the specified channel group.
2. When A/D conversion for each channel is completed, the A/D conversion result is stored into the corresponding A/D data registers (ADDRy).
3. When A/D conversion of all selected channels is completed, if the ADIE bit in ADCSR is set to 1 (A/D interrupt enable by completing A/D conversion), an ADI interrupt request is generated. A/D converter starts A/D conversion from the first channel.
4. The ADST bit is not cleared automatically, and steps 2 to 3 are repeated as long as the ADST bit remains set to 1 (A/D conversion start). When the ADST bit is cleared to 0 (A/D conversion stop), A/D conversion stops and the A/D converter enters a wait state.
5. If the ADST bit is later set to 1 (A/D conversion start), A/D conversion starts again from the first channel in the group.

Figure 23.6 shows an example of A/D conversion when three channels (AN0 to AN2) are selected for analog input.



Downloaded from Arrow.com.




Notes:
1. ↓ indicates an instruction execution by software.
2. Data during conversion is ignored.

**Figure 23.6 Example of A/D Converter Operation (Continuous Scan Mode)**



Downloaded from Arrow.com.

### 23.3.2.2 Single Scan Mode

In single scan mode, A/D conversion is to be performed for one cycle on the analog inputs of the specified channels as below.

1. When the ADST bit in ADCSR is set to 1 (A/D conversion start) by software, TPU, TMR, or an external trigger input, A/D conversion starts on the first channel in the specified channel group.
2. When A/D conversion for each channel is completed, the A/D conversion result is stored into the corresponding A/D data registers (ADDRy).
3. When A/D conversion of all selected channels is completed, if the ADIE bit in ADCSR is set to 1 (A/D interrupt enable by completing A/D conversion), an ADI interrupt request is generated.
4. The ADST bit remains at 1 (A/D conversion start) during A/D conversion, and is automatically cleared to 0 when A/D conversion of all selected channels end. The A/D converter enters a wait state.

Figure 23.7 shows an example of A/D conversion when three channels (AN4 to AN6) are selected for analog input.

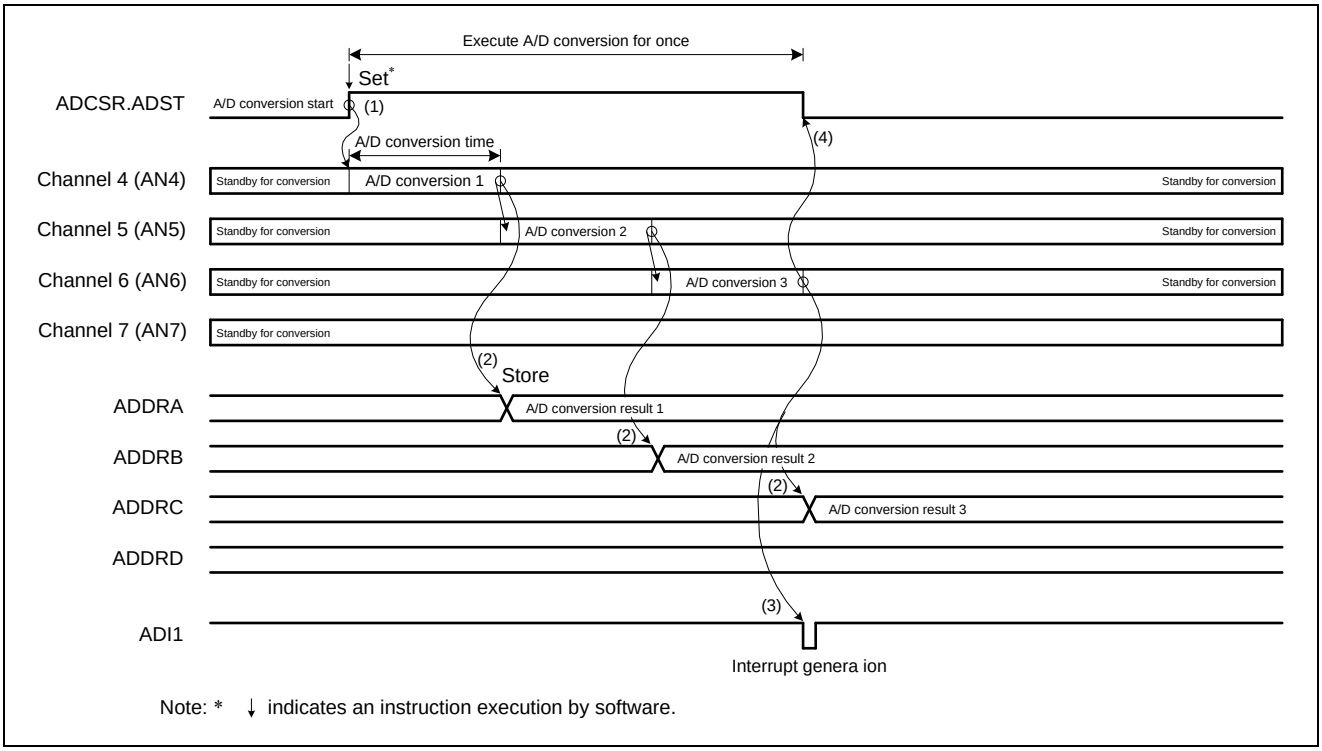



**Figure 23.7 Example of A/D Converter Operation (Single Scan Mode)**



Downloaded from Arrow.com.

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

The A/D converter samples the analog input when the A/D conversion start delay time (tD) passes after the conditions of A/D conversion start are generated by software, TPU, TMR, or an external trigger, then starts A/D conversion.

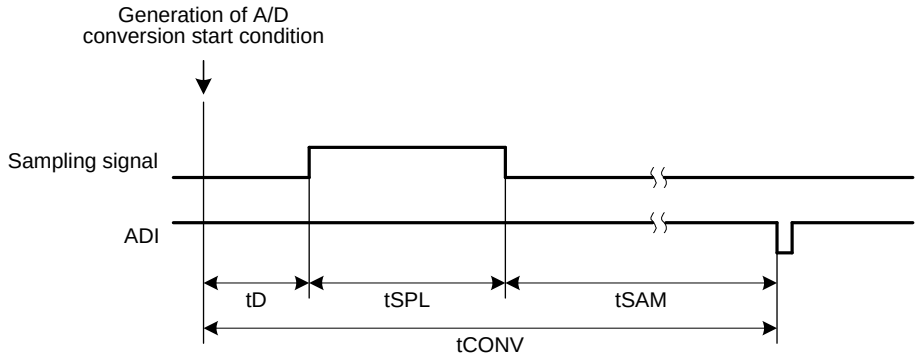
Figure 23.8 shows the A/D conversion timing.

The A/D conversion time (tCONV) directly after the generation of the A/D conversion start condition includes the A/D conversion start delay time (tD), the input sampling time (tSPL), and the successive conversion time (tSAM). The subsequent A/D conversion time (tCONV) includes tSPL and tSAM.

The input sampling time (tSPL) is the time charging of the input capacitance of the A/D converter's sample-and-hold circuit takes. If the impedance of the signal source is high and the sampling time is insufficient or the peripheral module clock (PCLK) is running at low speed, the sampling time can be adjusted by using the ADSSTR.

The successive conversion time (tSAM) is fixed at 25 states of ADCLK.

Table 23.6 lists the sample of ADSSTR settings and table 23.7 lists the A/D conversion time.




[Legend]
tD: A/D conversion start delay time
tSPL: Sampling time
tSAM: Successive conversion time
tCONV: A/D conversion time

**Figure 23.8 A/D Conversion Timing**

**Table 23.6 Example of ADSSTR Settings**

| Example of Setting | Setting Range | Sampling Time* |
|---|---|---|
| Standard (initial value) | 19h | 0.5 μs (When PCLK = ADCLK = 50 MHz) |
| Analog input signal impedance is high and the sampling time may be insufficient | 1Ah to FFh | Example: FFh<br>5.1 μs (When PCLK = ADCLK = 50 MHz) |
| Input sampling time is less than the initial value when ADCLK is below 50 MHz | 02h to 18h | Example: 14h<br>0.5 μs (When PCLK = ADCLK = 40 MHz) |

Note: * Set the sampling time ≥ 0.5 μs. Sampling time is shown as the formula below.

$$\text{Sampling time (μs)} = \frac{\text{Setting value of ADSSTR}}{\text{ADCLK (MHz)}}$$



Downloaded from Arrow.com.

**Table 23.7 A/D Conversion Time**

| Item | Symbol | Formula | |
|---|---|---|---|
| | | min | max |
| A/D conversion start delay time (1) | tD | $\frac{3}{\text{PCLK (MHz)}}$ | $\frac{1}{\text{ADCLK (MHz)}} + \frac{4}{\text{PCLK (MHz)}}$ |
| Input sampling time (2) | tSPL | $\frac{\text{Setting value of ADSSTR}}{\text{ADCLK (MHz)}}$ | |
| Successive conversion time (3) | tSAM | $\frac{25}{\text{ADCLK (MHz)}}$ | |
| A/D conversion time*1 | tCONV | (1) + (2) + (3) | |
| A/D conversion time*2 | tCONV | (2) + (3) | |

Notes: 1. A/D conversion time in single mode and scan mode (first round)
2. A/D conversion time in scan mode (after the second round)

The examples of the calculation of A/D conversion times are listed below.

When PCLK = ADCLK = 50 MHz, ADSSTR = 19h, and the conversion is the second round in scan mode,
A/D conversion time (tCONV) = ADSSTR/ADCLK + 25/ADCLK
= 25/50 MHz + 25/50 MHz
= 0.5 µs + 0.5 µs
= 1.0 µs

When PCLK = ADCLK = 40 MHz, ADSSTR = 14h, and conversion is the first (min.) round in scan mode,
A/D conversion time (tCONV) = 3/PCLK + ADSSTR/ADCLK + 25/ADCLK
= 3/40 MHz + 20/40 MHz + 25/40 MHz
= 0.075 µs + 0.5 µs + 0.625 µs
= 1.2 µs



Downloaded from Arrow.com.

### 23.3.4 Activation by External Triggers

External trigger signals (ADTRG0# to ADTRG3#) are capable of starting A/D conversion by each of the units.

For unit 0, when the setting of the AD0.ADCR.TRGS[2:0] bits is 011b (specifying ADTRG0# as a trigger), a falling edge on the ADTRG0# pin leads to setting of the ADST (A/D conversion start) bit in AD0.ADCR to 1 and thus starts A/D conversion. Figure 23.9 shows the timing.

An external trigger input is capable of simultaneously starting A/D conversion by two units (synchronized unit activation). Simultaneous activation of units 0 and 1 by a falling edge of the ADTRG0# signal is specified by setting the AD0.ADCR.TRGS[2:0] bits to 011b (specifying ADTRG0# as the trigger for unit 0) and the AD1.ADCR.TRGS[2:0] bits to 111b (specifying ADTRG0# as the trigger for unit 1). Simultaneous activation of units 2 and 3 by a falling edge of the ADTRG2# signal can be set up in the same way.

Take note that if the external trigger input is already at the low-level, selecting the external trigger may generate a falling edge in the internal signal and thus start A/D conversion.

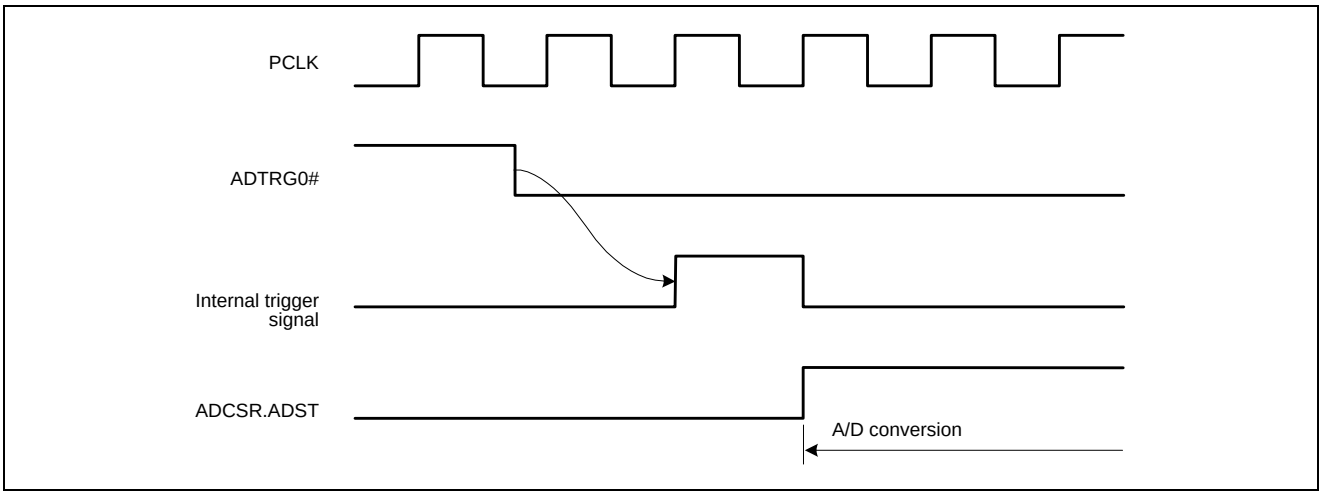



**Figure 23.9 Timing of Activation by an External Trigger**



Downloaded from Arrow.com.

### 23.3.5 Activation by the Compare-Match/Input-Capture A to D Signals from TPU0

The compare-match/input-capture A to D signals from TPU0 are capable of starting A/D conversion by units 0 to 3.

The connections between the units of the converter and the compare-match/input-capture A to D signals from TPU0 are shown in figure 23.10.

The compare-match/input-capture A to D signals from TPU0 are capable of starting A/D conversion by all four converter units. Setting the AD0.ADCR.TRGS[2:0] bits to 100b (specifying the compare-match/input-capture A signal from TPU0), setting the AD1.ADCR.TRGS[2:0] bits to 100b (specifying the compare-match/input-capture B signal from TPU0), setting the AD2.ADCR.TRGS[2:0] bits to 100b (specifying the compare-match/input-capture C signal from TPU0), and setting the AD3.ADCR.TRGS[2:0] bits to 100b (specifying the compare-match/input-capture D signal from TPU0) set up the compare-match/input-capture A to D signals from TPU0 as the triggers that start conversion by units 0 to 3.

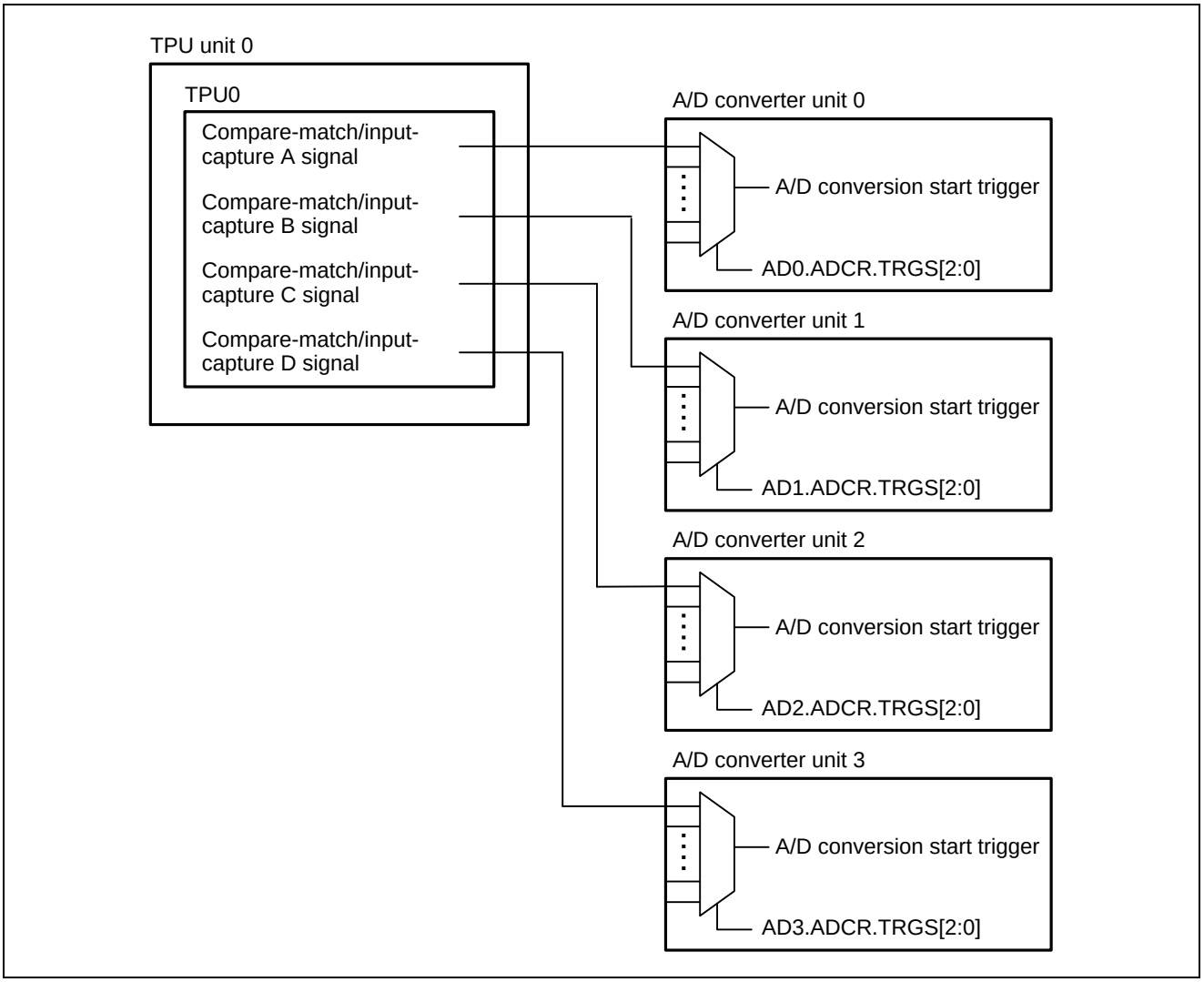



**Figure 23.10 Connections between Compare-Match/Input-Capture A to D Signals from TPU0 and the Respective Converter Units**



Downloaded from Arrow.com.

### 23.3.6 Activation by the Compare-Match/Input-Capture A Signals from TPU0 to TPU5

The compare-match/input-capture A signals from TPU0 to TPU5 are capable of starting A/D conversion by units 0 to 3. The compare-match/input-capture A signals from TPU6 to TPU11 are also capable of starting A/D conversion by units 0 to 3.

The connections between the units of the converter and the compare-match/input-capture A signals are shown in figure 23.11.

The compare-match/input-capture A signals from TPU0 to TPU5 are capable of simultaneously starting A/D conversion by a maximum of all four converter units. Setting the ADn.ADCR.TRGS[2:0] bits (n = 0 to 3) to 001b (specifying the compare-match/input-capture A signals from TPU0 to TPU5) and setting the TTGE bits in the TIERs of TPU0 and TPU2 to 1 set up the compare-match/input-capture A signals from TPU0 and TPU2 as the triggers that start simultaneous conversion by units 0 to 3.

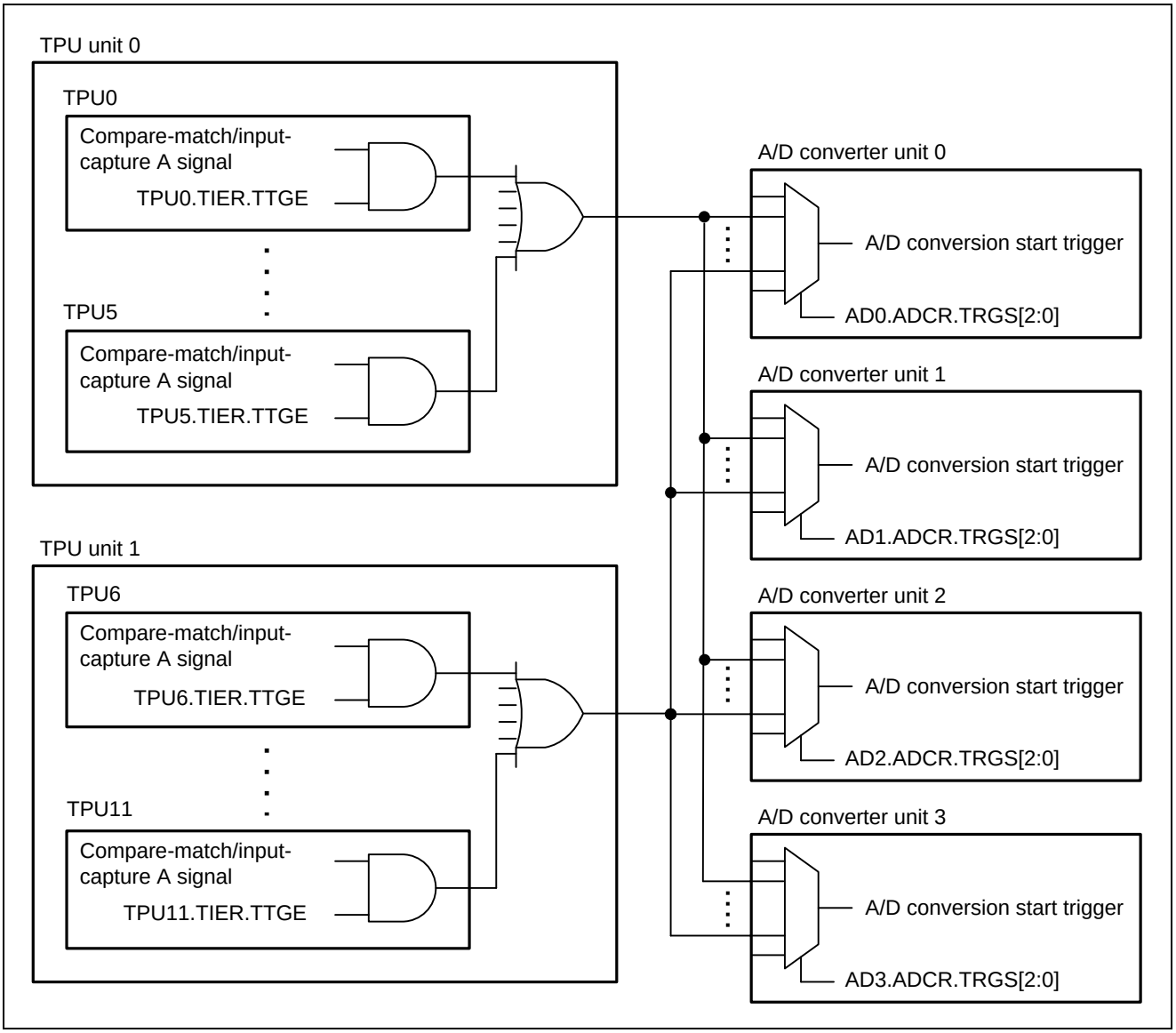



**Figure 23.11 Connections between Compare-Match/Input-Capture A Signals and the Respective Converter Units**



Downloaded from Arrow.com.

### 23.3.7 Activation on Compare-Match of TMR Units

The compare-match A signal from TMR0 is capable of starting A/D conversion by units 0 and 1. In the same way, the compare-match A signal from TMR2 is capable of starting A/D conversion by units 2 and 3.

The connections between the units of the converter and the compare-match A signals from TMR0 and TMR2 are shown in figure 23.12.

The compare-match A signal from TMR0 is capable of simultaneously starting A/D conversion by no more than two converter units. Setting the ADn.ADCR.TRGS[2:0] bits (n = 0, 1) to 010b (specifying the compare-match A signal from TMR0) and setting the ADTE bit in TCSR of TMR0 to 1 set up the compare-match A signal from TMR0 as the trigger that starts simultaneous conversion by units 0 and 1.

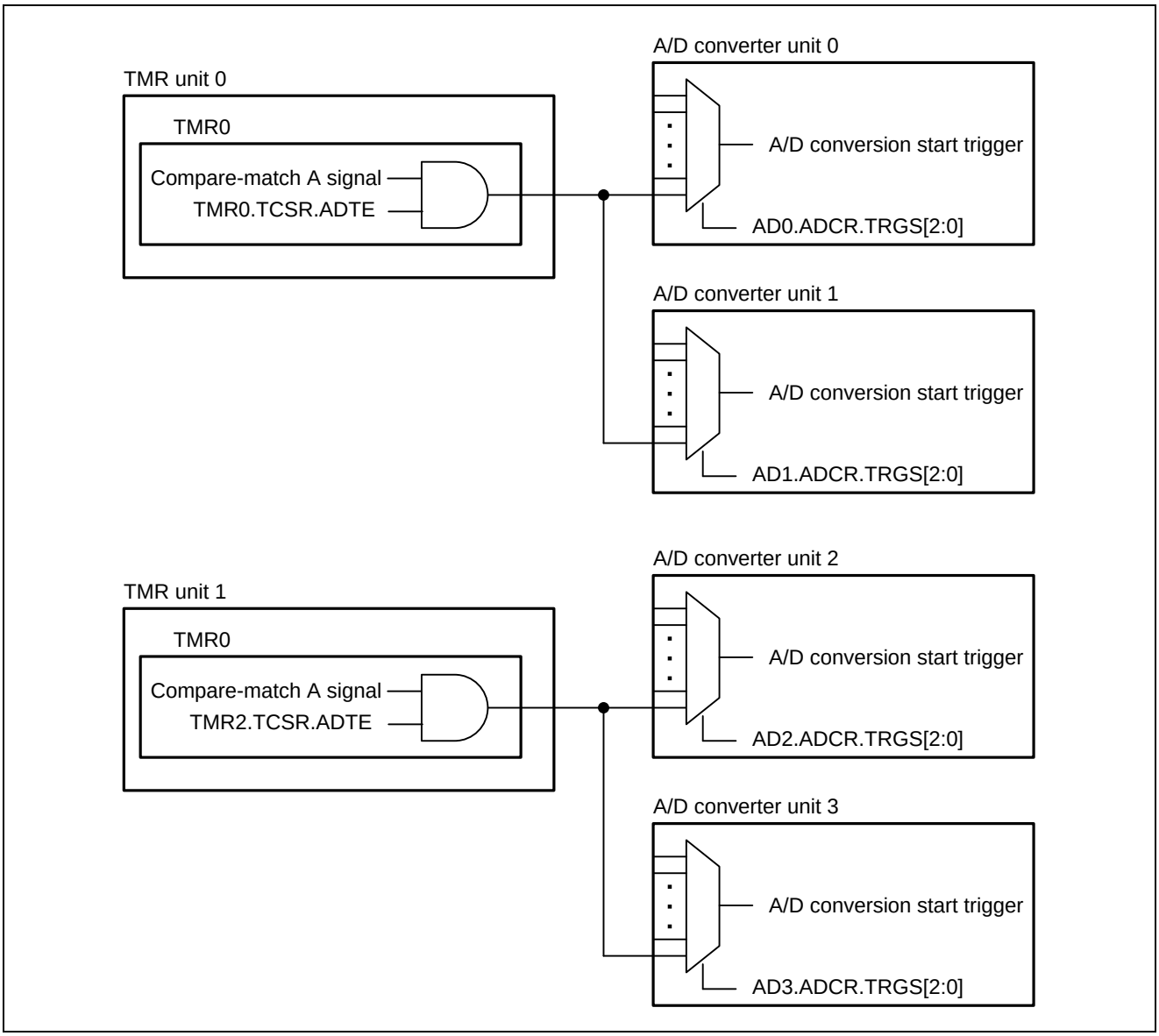



**Figure 23.12 Connections between Compare-Match A Signals from TMR0 and TMR2 and the Respective Converter Units**



Downloaded from Arrow.com.

## 23.4 Interrupt Source

The A/D converter generates an A/D conversion end interrupt (ADI) at the end of A/D conversion while the ADIE bit in ADCSR is set to 1 (after ADI interrupt is enabled by completing A/D conversion).

The data transfer controller (DTC) and DMA controller (DMAC) can be activated by an ADI interrupt. Having the converted data read by the DTC or DMAC in response to an ADI interrupt enables continuous conversion to be achieved without imposing a load on software.

**Table 23.8 A/D Converter Interrupt Source**

| Name | Interrupt Source | Interrupt Status Flag | DTC Activation | DMAC Activation |
|---|---|---|---|---|
| ADI0 | A/D conversion end | ICU.IR98.IR | Possible | Possible |
| ADI1 | A/D conversion end | ICU.IR99.IR | Possible | Possible |
| ADI2 | A/D conversion end | ICU.IR100.IR | Possible | Possible |
| ADI3 | A/D conversion end | ICU.IR101.IR | Possible | Possible |

## 23.5 A/D Conversion Accuracy Definitions

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

- Resolution

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

  The deviation inherent in the A/D converter, given by 1/2 LSB (see figure 23.13)
- Offset error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from the minimum voltage value 0000000000b (000h) to 0000000001b (001h) (see figure 23.14)
- Full-scale error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from 1111111110b (3FEh) to 1111111111b (3FFh) (see figure 23.14)
- Nonlinearity error

  The error with respect to the ideal A/D conversion characteristic between the zero voltage and the full-scale voltage. Does not include the offset error, full-scale error, or quantization error (see figure 23.14).
- Absolute accuracy

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



Downloaded from Arrow.com.

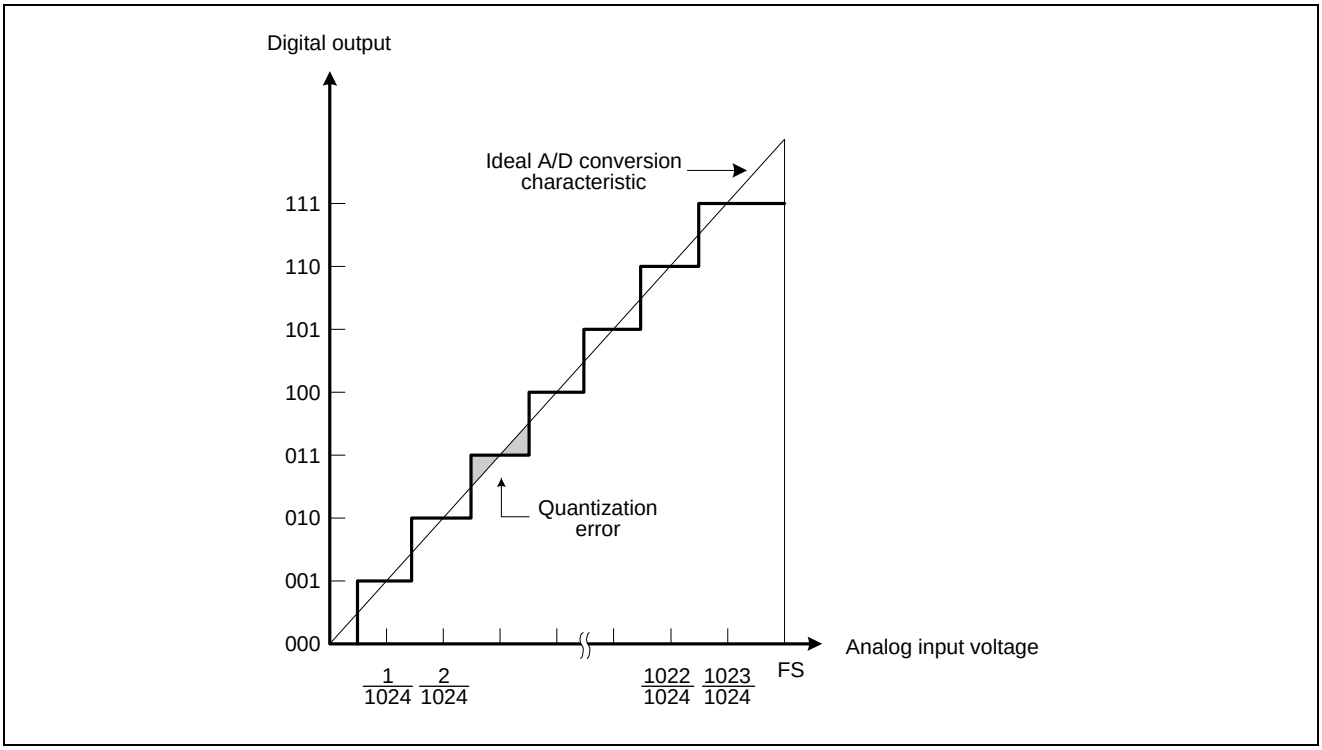



**Figure 23.13 A/D Conversion Accuracy Definitions (1)**

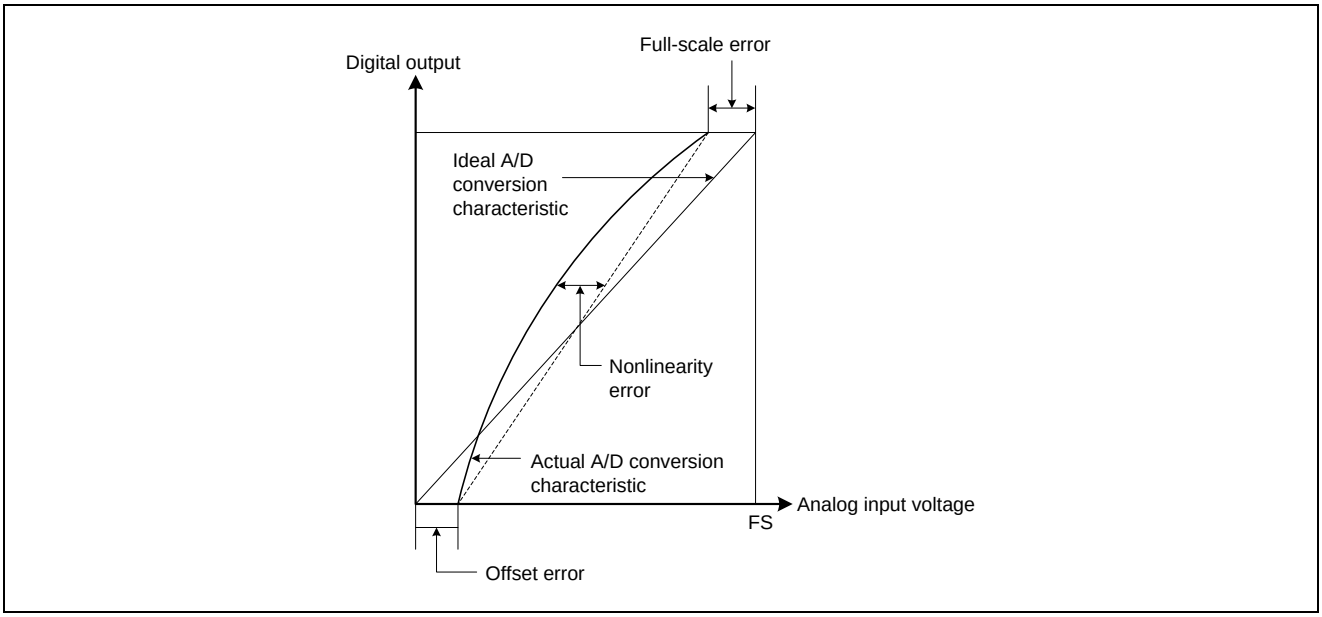



**Figure 23.14 A/D Conversion Accuracy Definitions (2)**



Downloaded from Arrow.com.

## 23.6 Usage Notes

### 23.6.1 Module Stop Function Setting

Operation of the A/D converter can be disabled or enabled for each unit using the module stop control register. The initial setting is for operation of the A/D converter to be halted. Register access is enabled by clearing the module stop state. For details, see section 8, Low Power Consumption.

### 23.6.2 Notes on Disabling A/D Conversion

To disable A/D conversion when an external trigger or timer has been selected as the condition for starting A/D conversion, set the ADCR.TRGS[2:0] bits to 000b to select the software trigger as the trigger to start A/D conversion, and then set the ADST bit in ADCSR to 0 (to stop A/D conversion).

### 23.6.3 Notes on Restarting A/D Conversion

Stopping analog input to the A/D converter by clearing the ADST bit in ADCSR to 0, requires one cycle of the ADCLK.

If A/D conversion is to be restarted right after the ADST bit was set to 0, set the ADST bit to 1 allow A/D conversion to restart after one clock cycle has elapsed.

### 23.6.4 Notes on Entering Power-Down States

When this LSI enters the module stop state or software standby mode with A/D conversion enabled, the analog power supply current is the same as it is during A/D conversion. If the analog power supply current needs to be reduced in the module stop state or software standby mode, disable A/D conversion. To do so, set the ADST bit in ADCSR to 0, and allow time for disabling of the analog input to the A/D converter.

Follow the procedure given below to ensure that this time is secured.

(1) Set the ADCR.TRGS[2:0] bits to 000b (software trigger).

(2) Clear the ADCSR.ADST bit to 0.

(3) Set the ADCR.CKS[1:0] bits to 11b (PCLK).

(4) After confirming that the A/D converter has been disabled, place the LSI in the module stop state or software standby mode.



Downloaded from Arrow.com.

### 23.6.5 Permissible Impedance of Signal Sources

To realize high-speed conversion 1.0 µs, the A/D conversion accuracy is guaranteed only when the impedance of the signal sources for analog input signals of this LSI circuit is less than or equal to 1.0 kΩ. If a large external capacitance is provided in the case of conversion in single mode, the input load becomes only the actual internal input resistance (6.5 kΩ), so the impedance of the signal source becomes insufficient. However, since the circuit has become a low-pass filter, the rapid and large fluctuations in the analog signal produced by differentiation (for example, greater than 5 mV/µs) become impossible to track (figure 23.15). Include a low-impedance buffer if high-rate analog signals are to be converted or conversion is to be in scan mode.

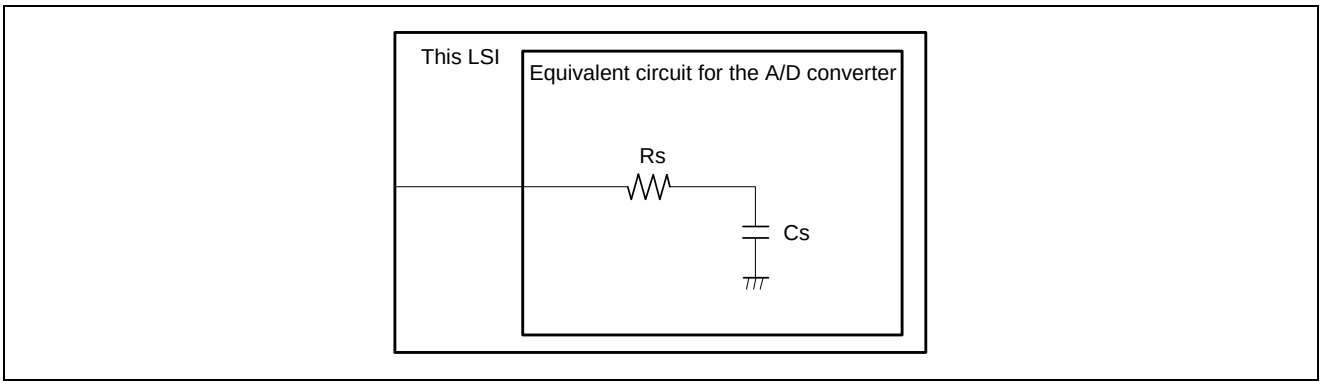



**Figure 23.15 Equivalent Circuit for the Internal Circuit of Analog Input Pins**

**Table 23.9 Specifications of Analog Pins**

| Item | | Min. | Max. | Unit |
|---|---|---|---|---|
| Permissible signal-source impedance | | — | 1.0 | kΩ |
| Values in the equivalent circuits for the internal circuits of pins | Rs | — | 6.5 | kΩ |
| | Cs | — | 6.0 | pF |

### 23.6.6 Factors Affecting Absolute Accuracy

Including a capacitor introduces ground coupling. A noisy ground can have a bad effect on absolute accuracy, so be sure to connect the $V_{REFL}$ pins and so on to an electrically stable ground.

Furthermore, a mounted filter circuit can interfere with digital-signal lines on the board, so take care to ensure that the design does not set up an antenna.



Downloaded from Arrow.com.

### 23.6.7 Ranges of Settings for Analog Power Supply and Other Pins

Using this LSI circuit with voltages beyond the ranges given below can have a bad effect on LSI reliability.

- Range for the setting of analog input voltages
  Keep voltages applied to analog input pins (ANn pins) within the range defined by $V_{REFL} \leq V_{AN} \leq V_{REFH}$.
- Relations between AVcc and AVss, $V_{REFH}$ and $V_{REFL}$, and Vcc and Vss
  Ensure that the relations between AVcc and AVss and between Vcc and Vss are AVcc = Vcc and AVss = Vss.
  To realize AVcc = Vcc and AVss = Vss at the points where the power-supply lines originate, set up closed loops with wiring runs between power-supply pins that are as short as possible and connect 0.1-µF capacitors as shown in figure 23.16. Even if the A/D converters are not in use, ensure that $V_{REFH}$ = AVcc = Vcc and $V_{REFL}$ = AVss = Vss by making the same connections.
- Range for the setting of $V_{REFH}$
  Keep the reference voltage on the $V_{REFH}$ pin within the range defined by $V_{REFH} \leq$ AVcc.

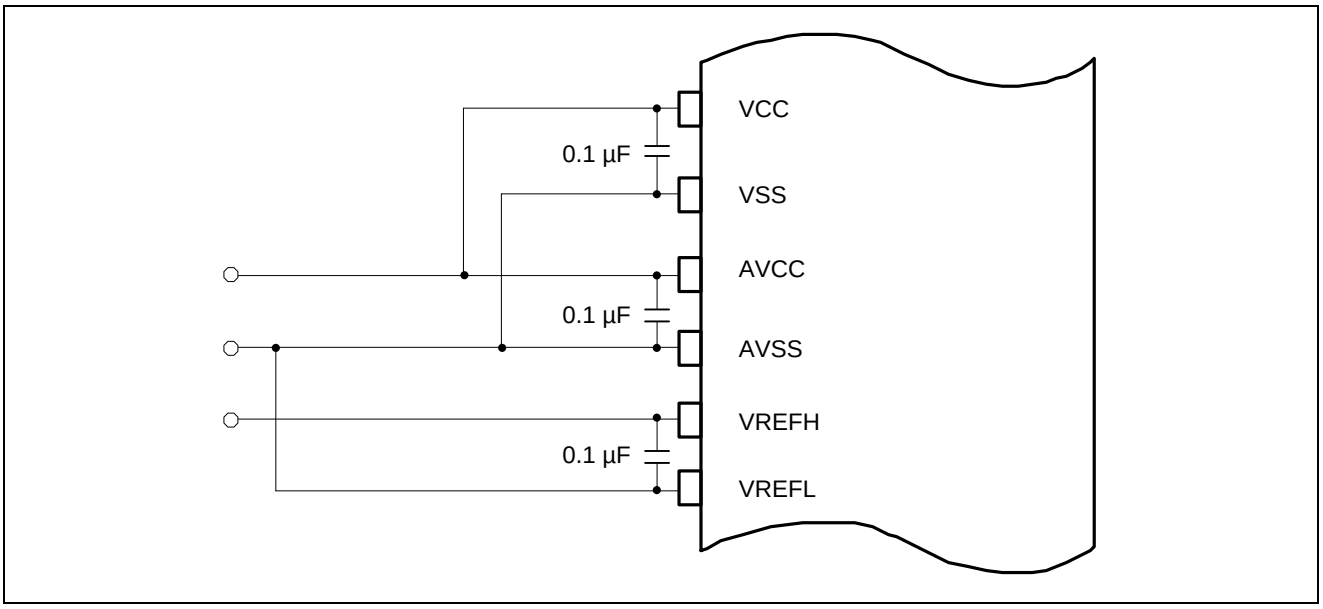



**Figure 23.16 Example of Connection for Power-Supply Pins**

### 23.6.8 Point for Caution Regarding Board Design

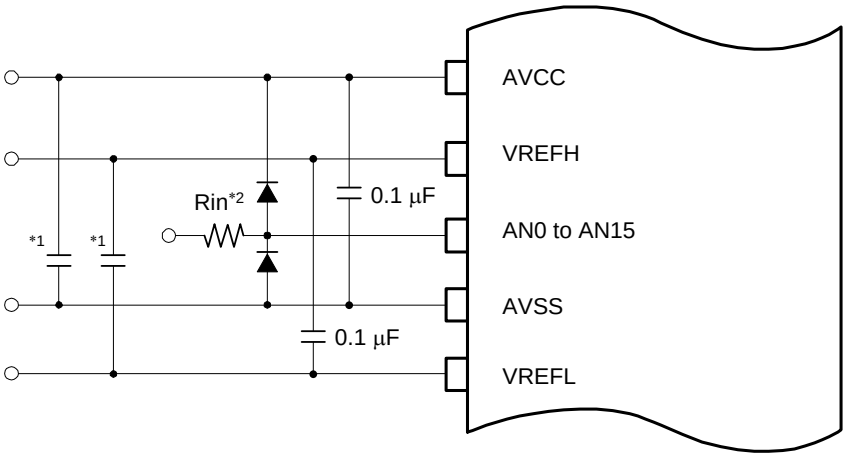
As far as possible, separate the analog circuits from the digital circuits in board design. Furthermore, do not allow the wiring runs for a signal of a digital circuit and a signal of an analog circuit to cross or be in each other's vicinity. Inductive coupling leads to analog circuits operating incorrectly and has a bad effect on the results of analog conversion. Keep the signal lines for analog input pins (AN0 to AN15), the analog reference power supply pin ($V_{REFH}$), the analog power-supply voltage (AVcc), and analog ground (AVSS) away from digital circuitry. Furthermore, connect the analog reference ground ($V_{REFL}$) to the stabilized ground (Vss) on the board at a single point.



Downloaded from Arrow.com.

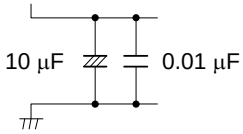
### 23.6.9 Point for Caution Regarding Countermeasures for Noise

To prevent destruction of the circuits for the analog input pins by abnormal voltages such as excessively large surges, connect capacitors as shown in figure 23.17 between AVcc and AVss, and between $V_{REFH}$ and $V_{REFL}$; also connect suitable protective circuits to the analog input pins (AN0 to AN15).




Notes: The value in parentheses indicates the pin number.

1. The value is for reference.




2. Rin: Input impedance

**Figure 23.17 Example of a Protective Circuit for Analog Inputs**

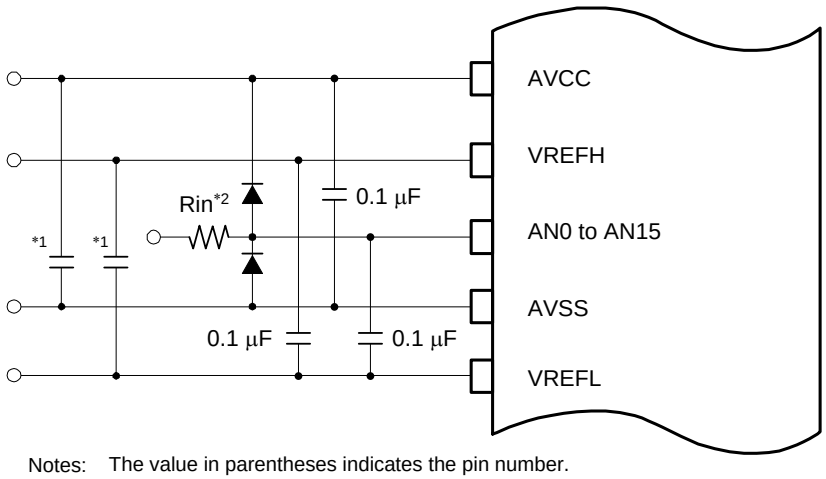


Downloaded from Arrow.com.

## 23.6.10 Realizing High-Speed Conversion

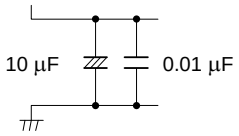
To realize high-speed conversion, connect external 0.1-µF capacitors between the analog input pins (AN0 to AN15) and $V_{REFL}$. This is shown in figure 23.18. However, to prevent the impedance of the signal source due to the input capacitance of the sample-and-hold circuit of the A/D converter from affecting the conversion time, the externally connected capacitors must be fully charged before the start of conversion.

Furthermore, when the voltages on the analog input pins fluctuate due to scanning and so on, so describing renewal of the charge of the externally connected capacitors, high-speed conversion is not realized.




Notes: The value in parentheses indicates the pin number.

1. The value is for reference.




2. Rin: Input impedance

**Figure 23.18 Example of an Externally Connected Capacitor for High-Speed Conversion**



Downloaded from Arrow.com.

### 23.6.11 Notes when Using Multiple Units of A/D Converter

Since the same power supply is used between the units of the A/D converter contained in the RX610 Group, when multiple units are used and conversion start timing between each unit is different, conversion accuracy may get affected. When the conversion accuracy is affected, adopt the following methods and perform an adequate evaluation.

#### (1) Example of recommended operation of simultaneous conversion of each unit at the time of multiple unit operation

When multiple units of A/D converter are used, select the activation method of the trigger select bit as either activation by compare match from TPU or activation by compare match from TMR, and match the conversion start timing and end timing of each unit.

Figure 23.19 shows timing example (1) when the conversion timings of four units are simultaneous.

Figure 23.20 shows timing example (2) when the conversion timings of four units are simultaneous.

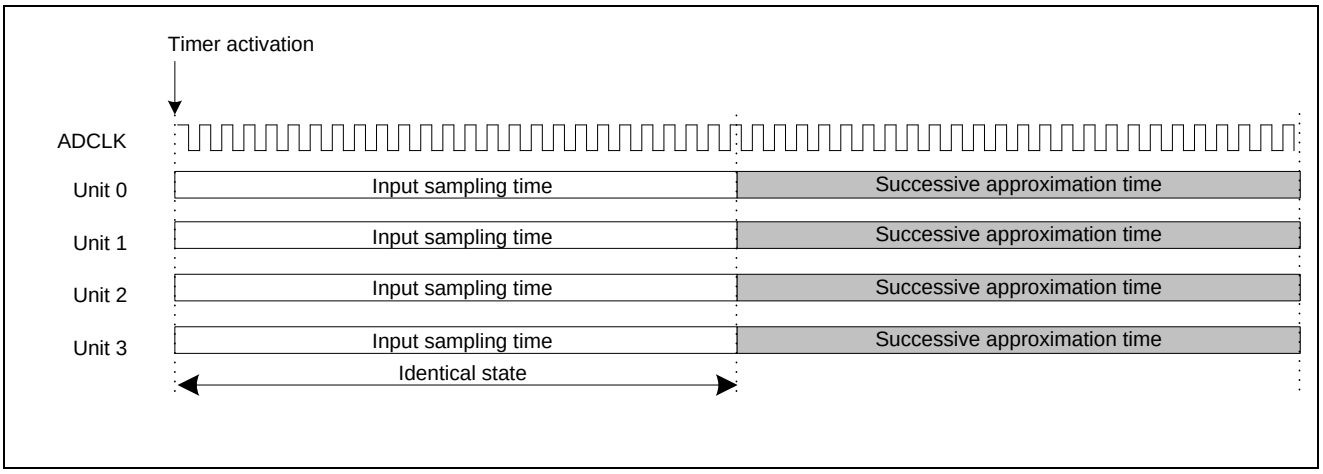



**Figure 23.19 Example (1) of Simultaneous Conversion Timings of Four Units**

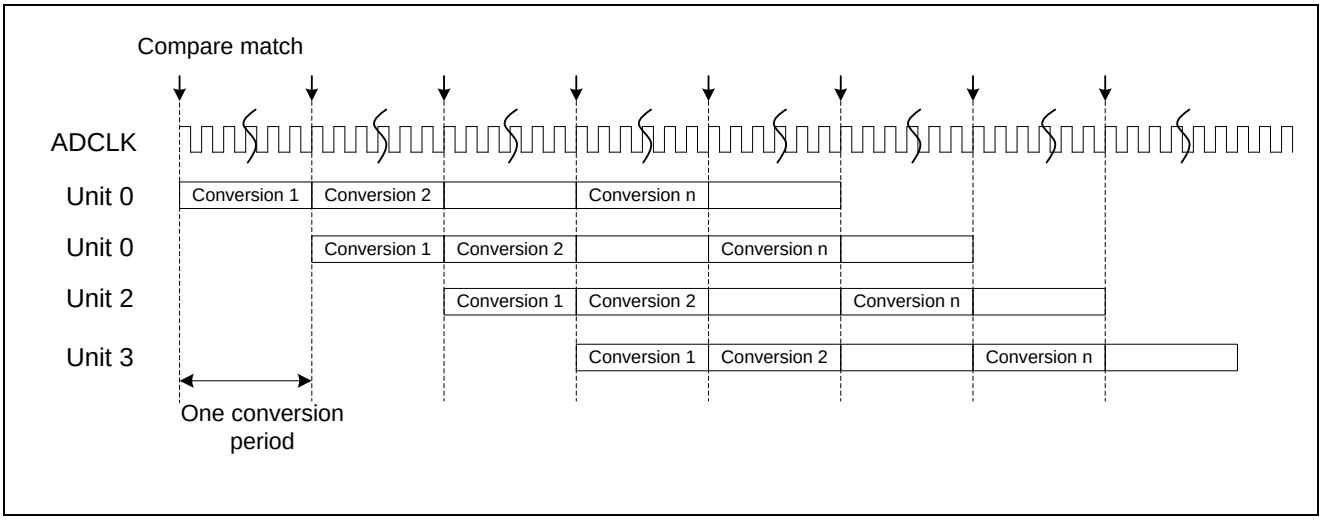



**Figure 23.20 Example (2) of Simultaneous Conversion Timings of Four Units**



Downloaded from Arrow.com.

### (2) Register settings of recommended operation

The following registers and bits should be set to the same value between all of the units.

- A/D sampling state register (ADSSTR)
- Clock select bits (ADCR.CKS[1:0])
- Trigger select bits (ADCR.TRGS[2:0])

### (3) Trigger selection corresponding to number of units

The number of units that can be activated depends on the selected trigger. See the following table.

**Table 23.10 Trigger Activation and Number of Corresponding Units**

| No. | Trigger Selection | ADCR.TRGS[2:0] | Number of Multiple Units | | |
|---|---|---|---|---|---|
| | | | 4 | 3 | 2 |
| 1 | Activation by compare match A to D[*1] of TPU0 | 100b | Possible | Possible | Possible |
| 2 | Activation by compare match A of TPU0 to TPU5[*2] | 001b | Possible | Possible | Possible |
| 3 | Activation by compare match A of TPU6 to TPU11[*2] | 101b | Possible | Possible | Possible |
| 4 | Activation by compare match A of TMR0[*3] | 010b | Impossible | Impossible | Possible |
| 5 | Activation by compare match A of TMR2[*3] | 010b | Impossible | Impossible | Possible |

Notes: 1. Set the same value to TGRA to TGRD.

2. The compare match A that is to be input to each unit of A/D converter should be set for the same TPU channel.

3. Units 0 and 1 of A/D converter become compare match A of TMR0, and units 2 and 3 of that become compare match A of TMR2.

For details on activation method of each unit, see section 23.3.5, Activation by the Compare-Match/Input-Capture A to D Signals from TPU0, section 23.3.6, Activation by the Compare-Match/Input-Capture A Signals from TPU0 to TPU5, and section 23.3.7, Activation on Compare-Match of TMR Units.

### (4) Other procedures

- Perform an averaging procedure using a program.

  Averaging procedure example: Perform A/D conversion of the analog input to identical pins successively for four times. Calculate an average of two values of the A/D conversion result excluding the maximum value and the minimum value.

- When the A/D converter is operated asynchronously, ensure that the conversion of one unit is complete and then operation of another single unit is started.



Downloaded from Arrow.com.

# 24. D/A Converter

## 24.1 Overview

The RX610 Group includes a two-channel of 10-bit D/A converter.

Table 24.1 lists the specifications of the D/A converter and figure 24.1 shows a block diagram of the D/A converter.

**Table 24.1 Specifications of D/A Converter**

| Item | Specifications |
|---|---|
| Resolution | 10 bits |
| Output channels | Two channels |
| Power-down function | Module stop state can be set for each unit. |

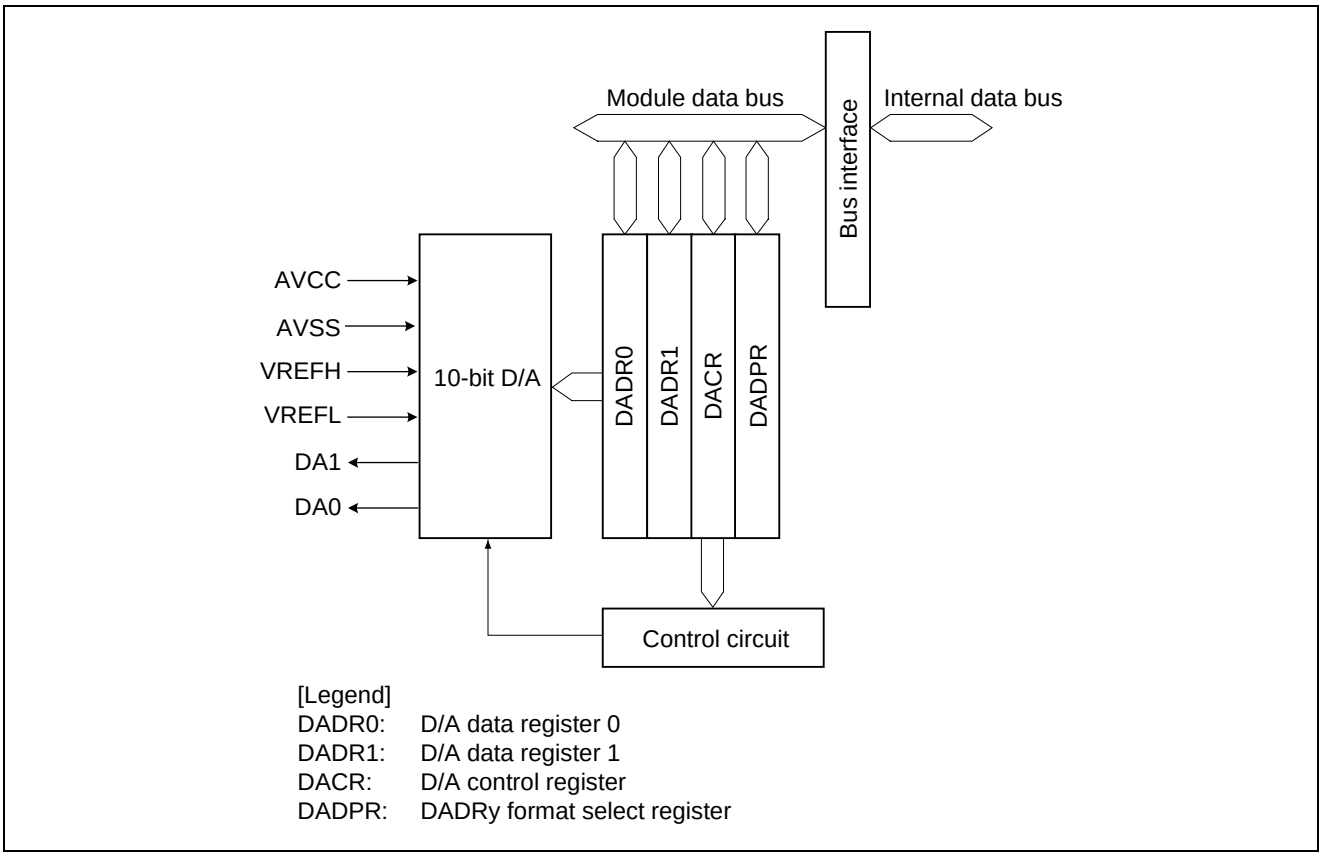



**Figure 24.1 Block Diagram of D/A Converter**



Downloaded from Arrow.com.

Table 24.2 lists the pin configuration of the D/A converter.

**Table 24.2 Pin Configuration of D/A Converter**

| Pin Name | I/O | Function |
|---|---|---|
| $AV_{CC}$ | Input | Analog circuit power supply pin |
| $AV_{SS}$ | Input | Analog circuit ground pin |
| VREFH | Input | D/A converter reference power supply pin |
| VREFL | Input | D/A converter reference ground pin<br>Connect this pin to the analog reference power supply (0 V). |
| DA0 | Output | Channel 0 analog output pin |
| DA1 | Output | Channel 1 analog output pin |



Downloaded from Arrow.com.

## 24.2 Register Descriptions

Table 24.3 lists the registers of the D/A converter.

**Table 24.3 Registers of D/A Converter**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| D/A data register 0 | DADR0 | 0000h | 0008 80C0h | 16 |
| D/A data register 1 | DADR1 | 0000h | 0008 80C2h | 16 |
| D/A control register | DACR | 1Fh | 0008 80C4h | 8 |
| DADRy format select register | DADPR | 00h | 0008 80C5h | 8 |

### 24.2.1 D/A Data Register y (DADRy) (y = 0, 1)

Addresses: 0008 80C0h, DADR1 0008 80C2h

DADPR.DPSEL bit = 0 (Data padded at the LSB end)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | | | | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DADPR.DPSEL bit = 1 (Data padded at the MSB end)

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DADRy registers are 16-bit readable/writable registers, which store data to which D/A conversion is to be performed. Whenever an analog output is enabled, the values in DADRy are converted and output to the analog output pins.

10-bit data can be relocated by setting the DPSEL bit in DADPR.

Bits "—" are always read as 0. The write value should always be 0.



Downloaded from Arrow.com.

### 24.2.2 D/A Control Register (DACR)

Address: 0008 80C4h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DAOE1 | DAOE0 | DAE | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b4 to b0 | — | Reserved | These bits are read as 1. The write value should always be 1. | R/W |
| b5 | DAE*1 | D/A Enable | 0: D/A conversion is independently controlled on channels 0 and 1.<br>1: D/A conversion on channels 0 and 1 is controlled as a single whole. | R/W |
| b6 | DAOE0 | D/A Output Enable 0 | 0: Analog output of channel 0 (DA0) is disabled.<br>1: D/A conversion of channel 0 is enabled.<br>Analog output of channel 0 (DA0) is enabled.*2 | R/W |
| b7 | DAOE1 | D/A Output Enable 1 | 0: Analog output of channel 1 (DA1) is disabled.<br>1: D/A conversion of channel 1 is enabled.<br>Analog output of channel 1 (DA1) is enabled.*2 | R/W |

Notes: 1. This bit controls D/A conversion in combination with the DAOEj bit (j = 0, 1). The DAOEj bit controls output of the results of conversion. For details, see table 24.4.

2. Set the P6.DDR.Bj bits (j = 7, 6) for pins used as analog outputs and the corresponding P6.ICR.Bj bits (j = 7, 6) to 0. For details, see section 14, I/O ports.

**Table 24.4 Controls of D/A Conversion**

| b5 | b7 | b6 | |
|---|---|---|---|
| DAE | DAOE1 | DAOE0 | Description |
| 0 | 0 | 0 | D/A conversion is disabled. |
| | | 1 | D/A conversion of channel 0 is enabled. D/A conversion of channel 1 is disabled.<br>Analog output of channel 0 (DA0) is enabled. Analog output of channel 1 (DA1) is disabled. |
| | 1 | 0 | D/A conversion of channel 0 is disabled. D/A conversion of channel 1 is enabled.<br>Analog output of channel 0 (DA0) is disabled. Analog output of channel 1 (DA1) is enabled. |
| | | 1 | D/A conversion of channels 0 and 1 is enabled.<br>Analog output of channels 0 and 1 (DA0 and DA1) is enabled. |
| 1 | 0 | 0 | D/A conversion of channels 0 and 1 is enabled.<br>Analog output of channels 0 and 1 (DA0 and DA1) is disabled. |
| | | 1 | D/A conversion of channels 0 and 1 is enabled.<br>Analog output of channel 0 (DA0) is enabled. Analog output of channel 1 (DA1) is disabled. |
| | 1 | 0 | D/A conversion of channels 0 and 1 is enabled.<br>Analog output of channel 0 (DA0) is disabled. Analog output of channel 1 (DA1) is enabled. |
| | | 1 | D/A conversion of channels 0 and 1 is enabled.<br>Analog output of channels 0 and 1 (DA0 and DA1) is enabled. |

DACR controls the operation of the D/A converter.



Downloaded from Arrow.com.

### DAE Bit (D/A Enable)

The DAE bit controls D/A conversion in combination with the DAOE0 and DAOE1 bits.

When the DAE bit is 0, D/A conversion is independently controlled on channels 0 and 1. When the DAE bit is 1, D/A conversion on channels 0 and 1 is controlled as a single whole. The DAOE0 and DAOE1 bits control output of the results of conversion.

### DAOE0 Bit (D/A Output Enable 0)

The DAOE0 bit controls the D/A conversion and analog output.

### DAOE1 Bit (D/A Output Enable 1)

The DAOE1 bit controls the D/A conversion and analog output.



Downloaded from Arrow.com.

### 24.2.3 DADRy Format Select Register (DADPR)

Address: 0008 80C5h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | DPSEL | — | — | — | — | — | — | — |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b6 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | DPSEL | DADRy Format Select | 0: D/A data register is padded at the LSB end.<br>1: D/A data register is padded at the MSB end. | R/W |

DADPR selects the placement of data in the D/A data registers.

#### DPSEL Bit (DADRy Format Select)

The DPSEL bit selects whether data in the D/A data registers is padded at the LSB or MSB end.



Downloaded from Arrow.com.

## 24.3 Operation

The D/A converter includes D/A conversion circuits for two channels, each of which can operate independently. When the DAOEn bit (n = 0, 1) in DACR is set to 1, D/A converter is enabled and the conversion result is output.

An operation example of D/A conversion on channel 0 is shown below. Figure 24.2 shows the timing of this operation.

1. Write the data for conversion to DADR0.
2. Set the DAOE0 bit in DACR to 1 to start D/A conversion. The conversion result is output from the analog output pin DA0 after the conversion time tDCCONV has elapsed. The conversion result continues to be output until DADR0 is written to again or the DAOE0 bit is cleared to 0. The output value is expressed by the following formula:

$$\frac{\text{Setting value of DADR0}}{1024} \times \text{VREFH}$$

3. If DADR0 is written to again, the conversion is immediately started. The conversion result is output after the conversion time tDCONV has elapsed.
4. If the DAOE0 bit is cleared to 0, analog output is disabled.

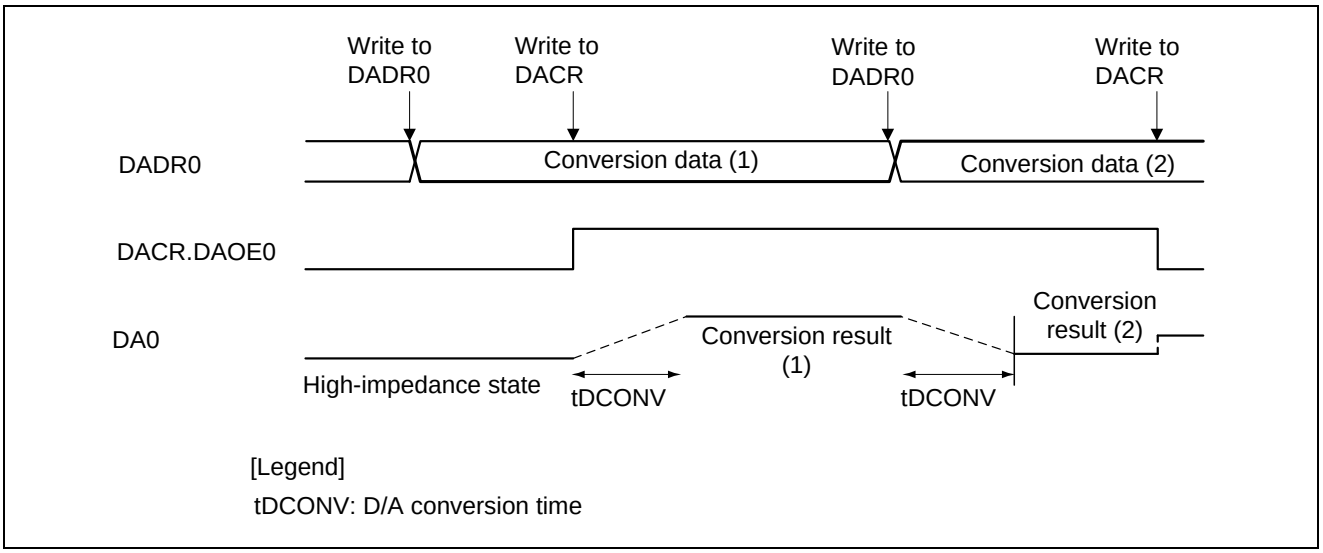



**Figure 24.2 Example of D/A Converter Operation**



Downloaded from Arrow.com.

## 24.4 Usage Notes

### 24.4.1 Module Stop Function Setting

Operation of the D/A converter can be disabled or enabled by using the module stop control register. The initial setting is for operation of the D/A converter to be halted. Register access is enabled by clearing the module stop state. For details, refer to section 8, Low Power Consumption.

### 24.4.2 Operation of the D/A Converter in Module Stop State

When this LSI enters the module stop state with D/A conversion enabled, the D/A outputs are retained, and the analog power supply current is the same as during D/A conversion. If the analog power supply current has to be reduced in the module stop state, disable D/A conversion by clearing the DAOE1, DAOE0, and DAE bits to 0.

### 24.4.3 Operation of the D/A Converter in Software Standby Mode

When this LSI enters software standby mode with D/A conversion enabled, the D/A outputs are retained, and the analog power supply current is the same as during D/A conversion. If the analog power supply current has to be reduced in software standby mode, disable D/A conversion by clearing the DAOE1, DAOE0, and DAE bits to 0.

### 24.4.4 Note on Entering Deep Software Standby Mode

When this LSI enters deep software standby mode with D/A conversion enabled, the outputs of the D/A converter are placed in a high impedance state.

### 24.4.5 Note when Using the A/D Converter and D/A Converter Simultaneously

Because the A/D converter and the D/A converter of the RX610 Group use the same power supply, conversion accuracy of A/D conversion result may get affected depending on the usage. The conversion accuracy is likely to be affected in the following cases.

- If the D/A data register (DADR) of D/A converter is rewritten while the A/D converter is being operated
- If the D/A control register (DACR) is rewritten by setting a value other than 000h to the DADR

When conversion accuracy is affected, implement the following measures.

#### (1) Method of rewriting DADR with AD converter operated

When rewriting the DADR while the AD converter is being operated, implement any one of the following methods.

1 When the DADR register is rewritten, discard the result of the A/D converter that is undergoing conversion.

2 When the DADR register is written, perform an averaging procedure for the A/D conversion result using a program.
Averaging procedure example: Perform A/D conversion of the analog input to identical pins successively for four times.

3 Calculate an average of two values of the A/D conversion result excluding the maximum and minimum values.

4 Set the rewriting procedure for DADR register as follows.

5 When the DADR register is written, keep the difference before changing and after changing at less than 100h, and when the next data is written, keep an interval of more than one conversion period of A/D converter. However, when rewriting is performed by maintaining the difference before and after changing at less than 080h, it is not necessary to maintain an interval of more than one conversion period of A/D converter.



Downloaded from Arrow.com.

Further, when multiple units of A/D converter are operated, implement the method described in section 23.6.11, Notes when Using Multiple Units of A/D Converter, and match one conversion period between each of the units.

Figure 24.3 shows an example of procedure of rewriting the DADR register from 000h to 3FFh.

- Before changing 000h (00 0000 0000b)
- First period of rewriting 100h (01 0000 0000b) => Difference before and after rewriting 100h
- Second period of rewriting 200h (10 0000 0000b) => Difference before and after rewriting 100h
- Third period of rewriting 300h (11 0000 0000b) => Difference before and after rewriting 100h
- Fourth period of rewriting 3FFh (11 1111 1111b) => Difference before and after rewriting 0FFh
- After changing 3FFh (11 1111 1111b)

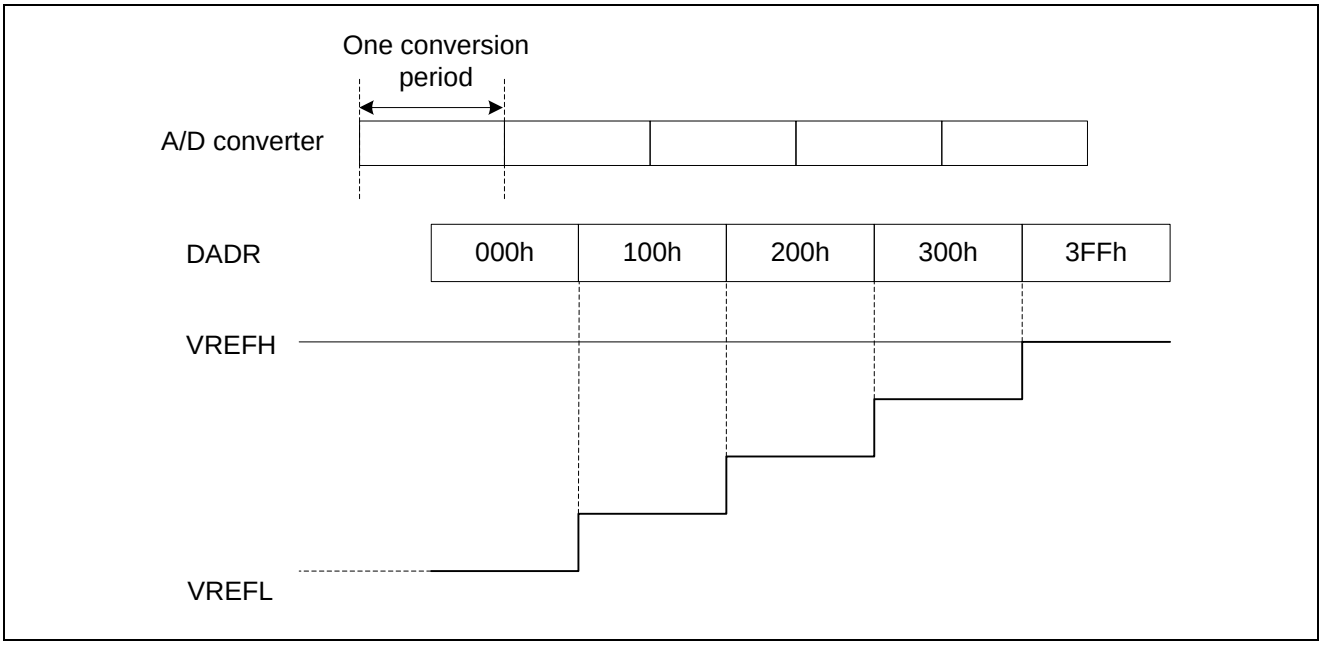



**Figure 24.3 Example of Procedure of Rewriting the DADR Register from 000h to 3FFh**

### (2) Rewriting DACR with AD converter operated

When the value of DACR is changed, rewrite the DACR with the DADR register value set to 000h. If the DACR register is rewritten under other conditions, the accuracy of the AD converter may not be guaranteed.



Downloaded from Arrow.com.

# 25. RAM

The RX610 Group has a high-speed static RAM.

## 25.1 Overview

Table 25.1 lists the specifications of the RAM.

**Table 25.1 Specifications of the RAM**

| Item | Description |
|---|---|
| RAM capacity | 128 Kbytes (RAM0: 64 Kbytes, RAM1: 64 Kbytes) |
| RAM address | RAM0: 0000 0000h to 0000 FFFFh<br>RAM1: 0001 0000h to 0001 FFFFh |
| Access | • Single-cycle access is possible for both reading and writing.<br>• Enabling or disabling of on-chip RAM is selectable.* |
| Data retention function | Data in RAM0 can be retained during periods in deep standby mode. |
| Power-down function | The module stop state is independently selectable for RAM0 and RAM1. |

Note: * Selectable by the RAME bit in SYSCR1. For details on SYSCR1, see section 3.2.4, System Control Register 1 (SYSCR1).

## 25.2 Operation

### 25.2.1 Data Retention

The address space for on-chip RAM is divided into the RAM0 and RAM1 areas. The difference between the two is whether internal power can be supplied in deep software standby mode.

Whether or not the supply of internal power to RAM0 continues in deep software standby mode is selectable by the RAMCUTn bit (n = 2 to 0) in DPSBYCR.

If continuation of the supply of internal power is selected, data in RAM0 are retained during periods in deep software standby mode. The supply of internal power supply to RAM1 is halted at this time, so data are not retained in RAM 1.

See section 8, Low Power Consumption, for details on the DPSBYCR.RAMCUTn (n = 0 to 2) bits.

### 25.2.2 Power-Down Function

Power consumption can be reduced by setting the module stop control register C (MSTPCRC) to stop supply of the clock signal to the on-chip RAM.

If the MSTPC0 bit in MSTPCRC is set to 1, supply of the clock signal to RAM0 is stopped. If the MSTPC1 bit in MSTPCRC is set to 1, supply of the clock signal to RAM1 is stopped.

The respective modules (RAM0 and RAM1) are thus placed in the module stop state by stopping supply of the clock signals. The initial value after a reset is for the RAM to be operational.

RAM is not accessible if it is in the module stop state. A transition to the module stop state should not be made while access to RAM is in progress.

For details on the MSTPCRC registers, see section 8, Low Power Consumption.



Downloaded from Arrow.com.

# 26. ROM (Flash Memory for Code Storage)

The RX610 has two flash-memory modules: a maximum 2-Mbyte ROM for storing code and a 32-Kbyte data flash block for storing data.

This section covers the flash memory for code storage. For the data flash, see section 27, Data Flash (Flash Memory for Data Storage).

## 26.1 Overview

Table 26.1 lists the specifications of the ROM, and figure 26.1 show a block diagram of the ROM, data-flash memory (data flash), and related modules.

**Table 26.1 Specifications of the ROM**

| Item | | Specifications |
|---|---|---|
| Two types of memory mats | | • User mat: 2 Mbytes, 1.5 Mbytes, 1 Mbyte, or 768 Kbytes*1<br>• User boot mat: 16 Kbytes |
| High-speed reading | | A read operation takes one cycle of ICLK |
| Programming/erasing method | | • The chip incorporates a dedicated sequencer (FCU) for programming of the ROM and data flash.<br>• Programming and erasing the ROM and data flash are handled by issuing commands to the FCU. |
| BGO (background operation) | | • The CPU is able to execute program code from areas other than the ROM or data flash while the ROM is being programmed or erased.<br>• Execution of program code from the ROM is possible while the data flash memory is being programmed or erased. |
| Suspension and resumption | | • The CPU is able to execute program code from the ROM during suspension of programming or erasure.<br>• Programming and erasure of the ROM can be restarted (resumed) after suspension. |
| Units of programming and erasure | | • Unit of programming for the user mat and user boot mat: 256 bytes<br>• Units of erasure for the user mat: 8 Kbytes (8 blocks), 64 Kbytes (9 blocks), 128 Kbytes (11 blocks)<br>• Unit of erasure for the user boot mat: 16 Kbytes |
| On-board programming (three types) | Boot mode | • The user mat and user boot mat are programmable via the SCI.<br>• The bit rate for SCI transfer between the host and RX610 is automatically adjusted. |
| | User boot mode | Booting up from the user boot mat and programming of the user mat |
| | User program | Programming of the user mat under program control |
| Off-board programming | | A PROM programmer can be used to program the user mat and user boot mat. |
| Protection | Software-controlled protection | The FENTRYR.FENTRY1*2, FENTRYR.FENTRY0, FWEPROR.FLWE[1:0], and lock bits can be used to prevent unintentional programming. |
| | Error protection | Prevention of further programming or erasure after the detection of abnormal operations during programming or erasure |
| Times for programming and erasure, durability (number of times reprogramming is poss ble) | | See section 28, Electrical Characteristics. |



Downloaded from Arrow.com.

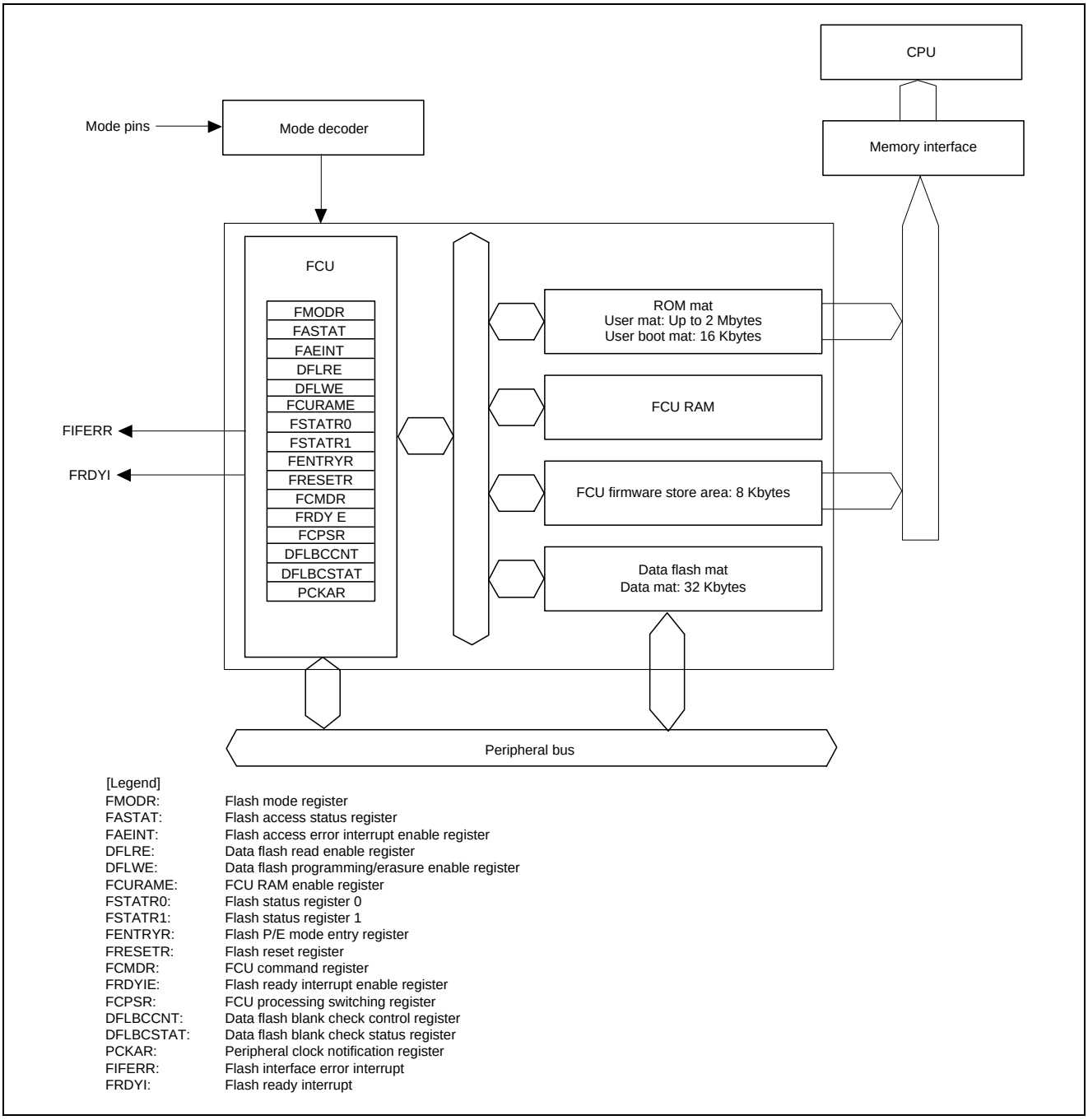
Notes: 1. Each product has different ROM sizes.

| Product Code | ROM Size | ROM Addresses |
|---|---|---|
| R5F56108 | 2 Mbytes | FFE0 0000h to FFFF FFFFh |
| R5F56107 | 1.5 Mbytes | FFE8 0000h to FFFF FFFFh |
| R5F56106 | 1 Mbyte | FFF0 0000h to FFFF FFFFh |
| R5F56104 | 768 Kbytes | FFF4 0000h to FFFF FFFFh |

2. Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.




[Legend]

| | |
|---|---|
| FMODR: | Flash mode register |
| FASTAT: | Flash access status register |
| FAEINT: | Flash access error interrupt enable register |
| DFLRE: | Data flash read enable register |
| DFLWE: | Data flash programming/erasure enable register |
| FCURAME: | FCU RAM enable register |
| FSTATR0: | Flash status register 0 |
| FSTATR1: | Flash status register 1 |
| FENTRYR: | Flash P/E mode entry register |
| FRESETR: | Flash reset register |
| FCMDR: | FCU command register |
| FRDYIE: | Flash ready interrupt enable register |
| FCPSR: | FCU processing switching register |
| DFLBCCNT: | Data flash blank check control register |
| DFLBCSTAT: | Data flash blank check status register |
| PCKAR: | Peripheral clock notification register |
| FIFERR: | Flash interface error interrupt |
| FRDYI: | Flash ready interrupt |

**Figure 26.1 Block Diagram of ROM**

Input and output pins associated with the ROM are listed in table 26.2.

**Table 26.2 Input and Output Pins Associated with the ROM**

| Pin Name | I/O | Description |
|---|---|---|
| P05/RxD4 | Input | Used in boot mode to receive data via SCI4 (for host communications) |
| P04/TxD4 | Output | Used in boot mode to transmit data from SCI4 (for host communications) |



Downloaded from Arrow.com.

## 26.2 Register Descriptions

Table 26.3 lists the registers related to ROM. Although some registers have bits related to data flash, this section deals only with the bits related to ROM. For details on the bits related to the data flash, see section 27.2, Register Descriptions in the data flash section.

The registers related to the ROM are initialized by a reset.

**Table 26.3 Registers Related to ROM**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Flash mode register | FMODR | 00h | 007F C402h | 8 |
| Flash access status register | FASTAT | 00h | 007F C410h | 8 |
| Flash access error interrupt enable register | FAEINT | 9Bh | 007F C411h | 8 |
| Flash ready interrupt enable register | FRDYIE | 00h | 007F C412h | 8 |
| FCU RAM enable register | FCURAME | 0000h | 007F C454h | 16 |
| Flash status register 0 | FSTATR0 | 80h | 007F FFB0h | 8 |
| Flash status register 1 | FSTATR1 | 0xh | 007F FFB1h | 8 |
| Flash P/E mode entry register | FENTRYR | 0000h | 007F FFB2h | 16 |
| Flash protection register | FPROTR | 0000h | 007F FFB4h | 16 |
| Flash reset register | FRESETR | 0000h | 007F FFB6h | 16 |
| FCU command register | FCMDR | FFFFh | 007F FFBAh | 16 |
| FCU processing switching register | FCPSR | 0000h | 007F FFC8h | 16 |
| Flash P/E status register | FPESTAT | 0000h | 007F FFCCh | 16 |
| Peripheral clock notification register | PCKAR | 0000h | 007F FFE8h | 16 |
| Flash write erase protection register | FWEPROR | 02h | 0008 C289h | 8 |



Downloaded from Arrow.com.

### 26.2.1 Flash Mode Register (FMODR)

Address: 007F C402h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | FRDMD | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4 | FRDMD | FCU Read Mode Select | 0: Memory Area Read Method<br>Memory area read mode is set to read a lock bit of ROM in ROM lock bit read mode.<br>1: Register Read Method<br>Register read mode is set to read a lock bit of ROM using the lock bit read 2 command. | R/W |
| b7 to b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

FMODR is a register to specify the method for the reading of lock bits.

When on-chip ROM is disabled, the data read from FMODR is 00h and writing is disabled.

FMODR is initialized by a reset.

#### FRDMD Bit (FCU Read Mode Select)

This bit is used to specify the method for the reading of lock bits.

If the blank checking command for the data flash is to be used, this bit has to be set for the register read mode (see section 27, Data Flash (Flash Memory for Data Storage)).



Downloaded from Arrow.com.

## 26.2.2 Flash Access Status Register (FASTAT)

Address: 007F C410h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ROMAE | – | – | CMDLK | DFLAE | – | DFLRPE | DFLWPE |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DFLWPE | Data Flash Programming/Erasure Protection Violation | See section 27, Data Flash (Flash Memory for Data Storage). | R/(W)* |
| b1 | DFLRPE | Data Flash Read Protection Violation | See section 27, Data Flash (Flash Memory for Data Storage). | R/(W)* |
| b2 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b3 | DFLAE | Data Flash Access Violation | See section 27, Data Flash (Flash Memory for Data Storage). | R/(W)* |
| b4 | CMDLK | FCU Command Lock | 0: FCU is not in the command-locked state<br>1: FCU is in the command-locked state | R |
| b6, b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | ROMAE | ROM Access Violation | 0: No ROM access error<br>1: ROM access error | R/(W)* |

Note: * Only 0 can be written after reading 1 to clear the flag.

FASTAT is a register to check if the access to the ROM/data flash is allowed.

When on-chip ROM is disabled, the data read from FASTAT is 00h and writing is disabled. When one of the bits in FASTAT is set to 1, the FCU is placed in the command-locked state (see section 26.8.2, Error Protection). To clear the command-locked state, a status clear command must be issued to the FCU after setting FASTAT to 10h.

FASTAT is initialized by a reset.

### CMDLK Bit (FCU Command Lock)

This bit indicates that the FCU is in the command-locked state (see section 26.8.2, Error Protection).

[Setting condition]

- After the FCU detects an error and enters the command-locked state

[Clearing condition]

- After the FCU processes a status clear command under conditions where FASTAT is set to 10h



Downloaded from Arrow.com.

### ROMAE Bit (ROM Access Violation)

This bit indicates whether a ROM access violation occurred.

When the ROMAE bit is set to 1, the ILGLERR bit in FSTATR0 is set to 1, placing the FCU in the command-locked state.

[Setting conditions]

- A read command is issued for ROM programming/erasure addresses 00E0 0000h to 00EF FFFFh when the FCU is in ROM P/E normal mode and the FENTRYR.FENTRY1 bit* is set to 1.
- A read command is issued for ROM programming/erasure addresses 00F0 0000h to 00FF FFFFh when the FCU is in ROM P/E normal mode and the FENTRYR.FENTRY0 bit is set to 1.
- A command is issued for ROM programming/erasure addresses 00E0 0000h to 00EF FFFFh when the FENTRY1 bit is set to 0.
- A command is issued for ROM programming/erasure addresses 00F0 0000h to 00FF FFFFh when the FENTRY0 bit is set to 0.
- A read command is issued for ROM read addresses FFE0 0000h to FFFF FFFFh when FENTRYR is set to other than 0000h.

[Clearing condition]

- When 0 is written after reading 1

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

### 26.2.3 Flash Access Error Interrupt Enable Register (FAEINT)

Address: 007F C411h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ROMAEIE | — | — | CMDLKIE | DFLAEIE | — | DFLRPEIE | DFLWPEIE |
| Value after reset: | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DFLWPEIE | Data Flash Programming/Erasure Protection Violation Interrupt Enable | See section 27, Data Flash (Flash Memory for Data Storage). | R/W |
| b1 | DFLRPEIE | Data Flash Read Protection Violation Interrupt Enable | See section 27, Data Flash (Flash Memory for Data Storage). | R/W |
| b2 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b3 | DFLAEIE | Data Flash Access Violation Interrupt Enable | See section 27, Data Flash (Flash Memory for Data Storage). | R/W |
| b4 | CMDLKIE | FCU Command Lock Interrupt Enable | 0: FIFERR interrupt requests disabled when the CMDLK bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the CMDLK bit in FASTAT is set to 1 | R/W |
| b6, b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | ROMAEIE | ROM Access Violation Interrupt Enable | 0: FIFERR interrupt requests disabled when the ROMAE bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the ROMAE bit in FASTAT is set to 1 | R/W |

FAEINT is a register to enable and disable the flash interface error interrupt (FIFERR).

When on-chip ROM is disabled, the data read from FAEINT is 00h and writing is disabled.

FAEINT is initialized by a reset.

#### CMDLKIE Bit (FCU Command Lock Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when an FCU command lock occurs and the CMDLK bit in FASTAT is set to 1.

#### ROMAEIE Bit (ROM Access Violation Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when a ROM access violation occurs and the ROMAE bit in FASTAT is set to 1.



Downloaded from Arrow.com.

### 26.2.4 FCU RAM Enable Register (FCURAME)

Address: 007F C454h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | KEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | FCRME |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FCRME | FCU RAM Enable | 0: Access to the FCU RAM disabled<br>1: Access to the FCU RAM enabled | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | KEY[7:0] | Key Code | This bit is used to enable or disable rewriting of the FCRME bit. | R/(W)* |

Note: * Write data is not retained.

FCURAME is a register to enable and disable an access to the FCU RAM area.

Only specific values written to the upper byte in word access are valid. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from FCURAME is 00h and writing is disabled.

FCURAME is initialized by a reset.

#### FCRME Bit (FCU RAM Enable)

This bit is used to enable and disable an access to the FCU RAM.

Data written to the FCRME bit is valid only when it is written in word access and the KEY[7:0] bits are C4h. When programming data to the FCU RAM, set FENTRYR to 0000h and stop the FCU.

#### KEY[7:0] Bits (Key Code)

These bits are used to enable or disable rewriting of the FCRME bit.

Data written to the KEY[7:0] bits is not retained.



Downloaded from Arrow.com.

## 26.2.5 Flash Status Register 0 (FSTATR0)

Address: 007F FFB0h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FRDY | ILGLERR | ERSERR | PRGERR | SUSRDY | – | ERSSPD | PRGSPD |
| Value after reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | PRGSPD | Programming Suspend Status | 0: Other than the status described below<br>1: During programming suspend processing or programming suspended | R |
| b1 | ERSSPD | Erasure Suspend Status | 0: Other than the status described below<br>1: When erasure suspend processing or erasure suspended | R |
| b2 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b3 | SUSRDY | Suspend Ready | 0: P/E suspend commands cannot be received<br>1: P/E suspend commands can be received | R |
| b4 | PRGERR | Programming Error | 0: Programming terminates normally<br>1: An error occurs during programming | R |
| b5 | ERSERR | Erasure Error | 0: Erasure terminates normally<br>1: An error occurs during erasure | R |
| b6 | ILGLERR | Illegal Command Error | 0: FCU detects no illegal command or ROM/data flash access<br>1: FCU detects an illegal command or ROM/data flash access | R |
| b7 | FRDY | Flash Ready | 0: During programming/erasure,<br>During suspending programming/erasure,<br>During the lock bit read 2 command processing,<br>During the blank check processing of data flash<br>(See section 27, Data Flash (Flash Memory for Data Storage)).<br>1: Processing described above is not performed | R |

FSTATR0 is a register to check the FCU status.

When on-chip ROM is disabled, the data read from FSTATR0 is 00h.

FSTATR0 is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

### PRGSPD Bit (Programming Suspend Status)

This bit is used to indicate that the FCU enters the programming suspend processing state or programming suspended state (see section 26.7, Suspending Operation).

[Setting condition]

- The FCU has initiated a write suspend command.

[Clearing condition]

- The FCU has accepted a resume command.



Downloaded from Arrow.com.

### ERSSPD Bit (Erasure Suspend Status)

This bit is used to indicate that the FCU enters the erasure suspend processing state or erasure suspended state (see section 26.7, Suspending Operation).

[Setting condition]

- The FCU has initiated an erasure suspend command.

[Clearing condition]

- The FCU has accepted a resume command.

### SUSRDY Bit (Suspend Ready Status)

This bit is used to indicate whether the FCU can receive a P/E suspend command.

[Setting condition]

- After starting programming/erasure process, the FCU enters a state in which P/E suspend commands can be received.

[Clearing conditions]

- The FCU has accepted a P/E suspend command.
- During programming/erasure process, the FCU enters the command-locked state.

### PRGERR Bit (Programming Error)

This bit is used to indicate the result of the ROM/data flash programming process by the FCU.

When the PRGERR bit is set to 1, the FCU is placed in the command-locked state (see section 26.8.2, Error Protection).

[Setting condition]

- An error occurs during programming.
- A programming command is issued to areas protected by a lock bit.

[Clearing condition]

- After the FCU processes a status clear command

### ERSERR Bit (Erasure Error)

This bit is used to indicate the result of the ROM/data flash erasure process by the FCU.

When the ERSERR bit is set to 1, the FCU is placed in the command-locked state (see section 26.8.2, Error Protection).

[Setting condition]

- An error occurs during erasure.
- A block erase command is issued to areas protected by a lock bit.

[Clearing condition]

- After the FCU processes a status clear command



Downloaded from Arrow.com.

### ILGLERR Bit (Illegal Command Error)

This bit is used to indicate that the FCU detects any illegal command or ROM/data flash access.

When the ILGLERR bit is set to 1, the FCU is placed in the command-locked state (see section 26.8.2, Error Protection).

[Setting conditions]

- The FCU detects an illegal command.
- The FCU detects an illegal ROM/data flash access
  (one of the ROMAE, DFLAE, DFLRPE, and DFLWPE bits in FASTAT is 1).
- The setting of FENTRYR is invalid.

[Clearing condition]

- After the FCU processes a status clear command under conditions where FASTAT is set to 10h

### FRDY Bit (Flash Ready)

This bit is used to check the processing status of the FCU.



Downloaded from Arrow.com.

### 26.2.6 Flash Status Register 1 (FSTATR1)

Address: 007F FFB1h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FCUERR | – | – | FLOCKST | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | x | x |

x: Undefined

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | — | Reserved | The read value is undefined and these bits cannot be modified. | R |
| b3, b2 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b4 | FLOCKST | Lock Bit Status | 0: Protected<br>1: Not protected | R |
| b6, b5 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |
| b7 | FCUERR | FCU Error | 0: No error occurs in the FCU processing<br>1: An error occurs in the FCU processing | R |

FSTATR1 is a register to check the FCU status.

When on-chip ROM is disabled, the data read from FSTATR1 is 00h.

FSTATR1 is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

#### FLOCKST Bit (Lock Bit Status)

This bit is to reflect the read data of a lock bit when using the lock bit read 2 command.

When the FRDY bit in FSTATR0 is set to 1 after a lock bit read 2 command is issued, valid data is stored in the FLOCKST bit. The value of the FLOCKST bit is retained until the completion of the next lock bit read 2 command.

#### FCUERR Bit (FCU Error)

This bit is used to indicate that an error occurs in the FCU internal processing.

When the FCUERR bit is set to 1, set the FRESET bit in FRESETR to 1 to initialize the FCU. Additionally, recopy the FCU firmware from the FCU firmware area to the FCU RAM area.



Downloaded from Arrow.com.

## 26.2.7 Flash Ready Interrupt Enable Register (FRDYIE)

Address: 007F C412h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | FRDYIE |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FRDYIE | Flash Ready Interrupt Enable | 0: FRDYI interrupt requests disabled<br>1: FRDYI interrupt requests enabled | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

FRDYIE is a register to enable and disable the flash ready interrupt (FRDYI) output.

When on-chip ROM is disabled, the data read from FRDYIE is 00h and writing is disabled.

FRDYIE is initialized by a reset.

### FRDYIE Bit (Flash Ready Interrupt Enable)

This bit is to enable/disable a FRDYI interrupt request when programming/erasure is completed.

If the FRDYIE bit is set to 1, a flash ready interrupt request (FRDYI) is generated when execution of the FCU command has completed (FSTATR0.FRDY bit changes from 0 to 1).



Downloaded from Arrow.com.

## 26.2.8 Flash P/E Mode Entry Register (FENTRYR)

Address: 007F FFB2h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | FEKEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FENTRYD | – | – | – | – | – | FENTRY1 | FENTRY0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FENTRY0 | ROM P/E Mode Entry 0 | 0: ROM within 1 Mbyte*[1] is in ROM read mode<br>1: ROM within 1 Mbyte*[1] is in ROM P/E mode | R/W |
| b1 | FENTRY1*[1] | ROM P/E Mode Entry 1 | 0: ROM more than 1 Mbyte is in ROM read mode<br>1: ROM more than 1 Mbyte is in ROM P/E mode | R/W |
| b6 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | FENTRYD | Data Flash P/E Mode Entry | See section 27, Data Flash (Flash Memory for Data Storage). | R/W |
| b15 to b8 | FEKEY[7:0] | Key Code | These bits enable or disable rewriting of the FENTRYD, FENTRY1*[2] and FENTRY0 bits. | R/(W)*[3] |

Notes: 1. ROM within 1 Mbyte: Address for reading, FFF0 0000h to FFFF FFFFh
Address for writing/erasing, 00F0 0000h to 00FF FFFFh
ROM more than 1 Mbyte: Address for reading, FFE0 0000h to FFEF FFFFh
Address for writing/erasing, 00E0 0000h to 00EF FFFFh

2. Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

3. Write data is not retained.

FENTRYR is a register to place the ROM/data flash in P/E mode.

To place the ROM/data flash in P/E mode so that the FCU can accept commands, one of the FENTRYD, FENTRY1*, and FENTRY0 bits must be set to 1. Note that if more than one of these bits is set to 1, the ILGLERR bit in FSTATR0 is set and the FCU enters the command-locked state.

In case of access to the FENTRYR register to select ROM reading mode, read the FENTRYR register to confirm that the new setting is in place before proceeding to read the ROM.

Only specific values written to the upper byte in word access are valid. Any other writing causes the register to be initialized. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from FENTRYR is 0000h and writing is disabled.

FENTRYR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

### FENTRY0 Bit (ROM P/E Mode Entry 0)

This bit is used to place 1 Mbyte of ROM (read addresses: FFF0 0000h to FFFF FFFFh, programming/erasure addresses 00F0 0000h to 00FF FFFFh) in P/E mode.

[Writing-enable conditions (when all of the following conditions are met)]

- On-chip ROM is enabled.
- The FRDY bit in FSTATR0 is set to 1.
- AAh is written to the FEKEY[7:0] bits in word access.

[Setting condition]

- When the writing-enable conditions are met, FENTRYR is set to 0000h, and 1 is written to the FENTRY0 bit

[Clearing conditions]

- Data is written in byte access.
- Data is written in word access when the FEKEY[7:0] bits are other than AAh.
- When the writing-enable conditions are met, 0 is written to the FENTRY0 bit.
- When the writing-enable conditions are met and FENTRYR is other than 0000h, data is written to FENTRYR.

### FENTRY1 Bit* (ROM P/E Mode Entry 1)

This bit is used to place 1 Mbyte of ROM (read addresses: FFE0 0000h to FFEF FFFFh, programming/erasure addresses 00E0 0000h to 00EF FFFFh) in P/E mode.

[Writing-enable conditions (when all of the following conditions are met)]

- On-chip ROM is enabled.
- The FRDY bit in FSTATR0 is set to 1.
- AAh is written to the FEKEY[7:0] bits in word access.

[Setting condition]

- When the writing-enable conditions are met, FENTRYR is set to 0000h, and 1 is written to the FENTRY1 bit*

[Clearing conditions]

- Data is written in byte access.
- Data is written in word access when the FEKEY[7:0] bits are other than AAh.
- When the writing-enable conditions are met, 0 is written to the FENTRY1 bit*.
- When the writing-enable conditions are met and FENTRYR is other than 0000h, data is written to FENTRYR.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

### FEKEY[7:0] Bits (Key Code)

These bits enable or disable rewriting of the FENTRYD, FENTRY1*, and FENTRY0 bits.

Data written to the FEKEY[7:0] bits is not retained.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

**Table 26.4 Correspondence of FENTRY1 and FENTRY0 Bits in Each Product**

| Product Code | ROM Size | FENTRY1 Bit*[1] | FENTRY0 Bit | FCU Mode | Target Addresses |
|---|---|---|---|---|---|
| R5F56108 | 2 Mbytes | 0 | 0 | ROM read mode | Read addresses: FFE0 0000h to FFFF FFFFh |
| | | 0 | 1 | ROM P/E mode | Programming/erasure addresses: 00F0 0000h to 00FF FFFFh |
| | | | | | Access disabled addresses: FFE0 0000h to FFFF FFFFh |
| | | 1 | 0 | | Programming/erasure addresses: 00E0 0000h to 00EF FFFFh |
| | | | | | Access disabled addresses: FFE0 0000h to FFFF FFFFh |
| | | 1 | 1 | Do not set. | — |
| R5F56107 | 1.5 Mbytes | 0 | 0 | ROM read mode | Read addresses: FFE8 0000h to FFFF FFFFh |
| | | 0 | 1 | ROM P/E mode | Programming/erasure addresses: 00F0 0000h to 00FF FFFFh |
| | | | | | Access disabled addresses: FFE8 0000h to FFFF FFFFh |
| | | 1 | 0 | | Programming/erasure addresses: 00E8 0000h to 00EF FFFFh |
| | | | | | Access disabled addresses: FFE8 0000h to FFFF FFFFh |
| | | 1 | 1 | Do not set. | — |
| R5F56106 | 1 Mbyte | Unusable*[2] | 0 | ROM read mode | Read addresses: FFF0 0000h to FFFF FFFFh |
| | | Unusable*[2] | 1 | ROM P/E mode | Programming/erasure addresses: 00F0 0000h to 00FF FFFFh |
| | | | | | Access disabled addresses: FFF0 0000h to FFFF FFFFh |
| R5F56104 | 768 Kbytes | Unusable*[2] | 0 | ROM read mode | Read addresses: FFF4 0000h to FFFF FFFFh |
| | | Unusable*[2] | 1 | ROM P/E mode | Programming/erasure addresses: 00F4 0000h to 00FF FFFFh |
| | | | | | Access disabled addresses: FFF4 0000h to FFFF FFFFh |

Notes: 1. Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

2. This is a reserved bit in the R5F56106 and R5F56104. The write value should always be 0.



Downloaded from Arrow.com.

## 26.2.9 Flash Protection Register (FPROTR)

Address: 007F FFB4h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | FPKEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | FPROTCN |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FPROTCN | Lock Bit Protection Cancel | 0: Protection with a lock bit enabled<br>1: Protection with a lock bit disabled | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | FPKEY[7:0] | Key Code | These bits are used to enable or disable rewriting of the FPROTCN bit. | R/(W)* |

Note: * Write data is not retained.

FPROTR is a register used to enable/disable the programming/erasure protection function with a lock bit.

Only specific values written to the upper byte in word access are valid. Any other writing causes the register to be initialized. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from FPROTR is 0000h and writing is disabled.

FPROTR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

### FPROTCN Bit (Lock Bit Protection Cancel)

This bit is used to enable/disable the programming/erasure protection with a lock bit.

[Setting condition]

- The FPKEY[7:0] bits are set to 55h, and the FPROTCN bit is set to 1 in word access when the value of FENTRYR is other than 0000h.

[Clearing conditions]

- Data is written in byte access.
- Data is written in word access when the FPKEY[7:0] bits are other than 55h.
- The FPKEY[7:0] bits are set to 55h, and the FPROTCN bit is set to 0 in word access.
- The value of FENTRYR is 0000h.

### FPKEY[7:0] Bits (Key Code)

These bits are used to enable or disable rewriting of the FPROTCN bit.

Data written to the FPKEY[7:0] bits is not retained.



Downloaded from Arrow.com.

## 26.2.10 Flash Reset Register (FRESETR)

Address: 007F FFB6h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | FRKEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | FRESET |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FRESET | Flash Reset | 0: FCU is not reset<br>1: FCU is reset | R/W |
| b7 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | FRKEY[7:0] | Key Code | These bits are used to enable or disable rewriting of the FRESET bit. | R/(W)* |

Note: * Write data is not retained.

FRESETR is a register to initialize the FCU.

Only specific values written to the upper byte in word access are valid. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from FRESETR is 0000h and writing is disabled.

FRESETR is initialized by a reset.

### FRESET Bit (Flash Reset)

When the FRESET bit is set to 1, programming/erasure operations for the ROM/data flash are forcibly terminated, and the FCU is initialized.

High voltage is applied to the memory of the ROM/data flash during programming/erasure. To ensure time required for dropping the voltage applied to the memory, keep the FRESET bit set to 1 for tRESW2 (see section 28, Electrical Characteristics) when initializing the FCU. While the FRESET bit is kept 1, prohibit the ROM/data flash from being read. Additionally, when the FRESET bit is set to 1, the FCU commands cannot be used because FENTRYR is initialized.

Writing of the FRESET bit is enabled only in word access and when the FRKEY[7:0] bits are CCh.

### FRKEY[7:0] Bits (Key Code)

These bits are used to enable or disable rewriting of the FRESET bit.

Data written to the FRKEY[7:0] bits is not retained.



Downloaded from Arrow.com.

## 26.2.11 FCU Command Register (FCMDR)

Address: 007F FFBAh

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | CMDR[7:0] | | | | | | | |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | PCMDR[7:0] | | | | | | | |
| Value after reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | PCMDR[7:0] | Precommand | Store the command immediately before the last command received by the FCU. | R |
| b15 to b8 | CMDR[7:0] | Command | Store the last command received by the FCU. | R |

FCMDR is a register to store commands received by the FCU.

When on-chip ROM is disabled, the data read from FCMDR is 0000h and writing is disabled.

FCMDR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

Table 26.5 lists the states of FCMDR after receiving each command. For details on the blank check processing, see section 27.6, Programming and Erasing the Data Flash Memory.

**Table 26.5 States of FCMDR after Receiving Each Command**

| Command | CMDR | PCMDR |
|---|---|---|
| Normal mode transition | FFh | Previous command |
| Status read mode transition | 70h | Previous command |
| Lock bit read mode transition (lock bit read 1) | 71h | Previous command |
| Peripheral clock notification command | E9h | Previous command |
| Programming | E8h | Previous command |
| Block erase | D0h | 20h |
| P/E suspend | B0h | Previous command |
| P/E resume | D0h | Previous command |
| Status register clear | 50h | Previous command |
| Lock bit read 2 blank check | D0h | 71h |
| Lock bit programming | D0h | 77h |



Downloaded from Arrow.com.

## 26.2.12 FCU Processing Switching Register (FCPSR)

Address: 007F FFC8h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | ESUSPMD |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | ESUSPMD | Erasure Suspend Mode | 0: suspension priority mode<br>1: Erasure priority mode | R/W |
| b15 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

FCPSR is a register to select the method of suspending the FCU erasure processing.

When on-chip ROM is disabled, the data read from FCPSR is 0000h and writing is disabled.

FCPSR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

### ESUSPMD Bit (Erasure Suspend Mode)

This bit is to select the erasure suspend mode for when a P/E suspend command is issued while the FCU executes the erasure processing for the ROM/data flash (see section 26.7, Suspending Operation).



Downloaded from Arrow.com.

## 26.2.13 Flash P/E Status Register (FPESTAT)

Address: 007F FFCCh

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | PEERRST[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | PEERRST[7:0] | P/E Error Status | 01h: Programming error against areas protected by a lock bit<br>02h: Programming error due to sources other than the lock bit protection<br>11h: Erasure error against areas protected by a lock bit<br>12h: Erasure error due to sources other than the lock bit protection<br>(Values other than above are reserved) | R |
| b15 to b8 | — | Reserved | These bits are always read as 0 and cannot be modified. | R |

FPESTAT is a register to indicate the result of the programming/erasure processing for the ROM/data flash.

When on-chip ROM is disabled, the data read from FPESTAT is 0000h and writing is disabled.

FPESTAT is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

### PEERRST[7:0] Bits (P/E Error Status)

These bits are used to indicate the reason of an error that occurs during the programming/erasure processing for the ROM/data flash.

The value of the PEERRST[7:0] bits is valid only when the ERSERR bit or PRGERR bit in FSTATR0 is 1. The value of the reason of the past error is retained in the PEERRST[7:0] bits when the ERSERR bit and PRGERR bit is 0.



Downloaded from Arrow.com.

## 26.2.14 Peripheral Clock Notification Register (PCKAR)

Address: 007F FFE8h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | PCKA[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b7 to b0 | PCKA[7:0] | Peripheral Clock Notification | These bits are used to set the peripheral clock (PCLK) at the programming/erasure for the ROM/data flash. | R/W |
| b15 to b8 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

PCKAR is a register to notify the sequencer of the frequency setting data of the peripheral clock (PCLK) at the programming/erasure for the ROM/data flash. This setting is used to control programming and erasure times.

When on-chip ROM is disabled, the data read from PCKAR is 0000h and writing is disabled.

PCKAR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

### PCKA[7:0] Bits (Peripheral Clock Notification)

These bits are used to set the peripheral clock (PCLK) at the programming/erasure for the ROM/data flash.

Set the PCKA[7:0] bits to the PCLK frequency and issue a peripheral clock notification command before programming/erasure. Do not change the frequency during the programming/erasure processing for the ROM/data flash.

Calculate the setting value as follows:

- Convert an operating frequency represented in MHz units to binary and write it to the PCKA[7:0] bits.

  For example, when the operating frequency of the peripheral clock is 35.9 MHz, the setting value is calculated as follows:

- Round up 35.9
- Convert 36 to binary and set the upper bits and lower bits of the PCKA[7:0] bits to 00h and 24h (0010 0100b).

Notes: 1. When the PCKA[7:0] bits are set to values outside the range from 8 to 50 MHz, do not issue a programming command to the ROM/data flash.

2. When the PCKA[7:0] bits are set to a frequency that is different from the FCLK, the data of the ROM/E2 data flash may be damaged.

3. Please note that programming time depends on the frequency to some extent even if the PCKA[7:0] bits are used.



Downloaded from Arrow.com.

## 26.2.15 Flash Write Erase Protection Register (FWEPROR)

Address: 0008 C289h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | FLWE[1:0] | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b1, b0 | FLWE[1:0] | Flash Write/Erase | b1 and b0<br>0 0: Write/erase disabled<br>0 1: Write/erase enabled<br>1 0: Write/erase disabled (initial value)<br>1 1: Write/erase disabled | R/W |
| b7 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

FWEPROR is a readable/writable register to protect the execution of the flash write/erase with software.

FWEPROR is initialized in software standby mode or deep software standby.

### FLWE[1:0] Bits (Flash Write/Erase)

These bits protect the execution of the flash write/erase with software.

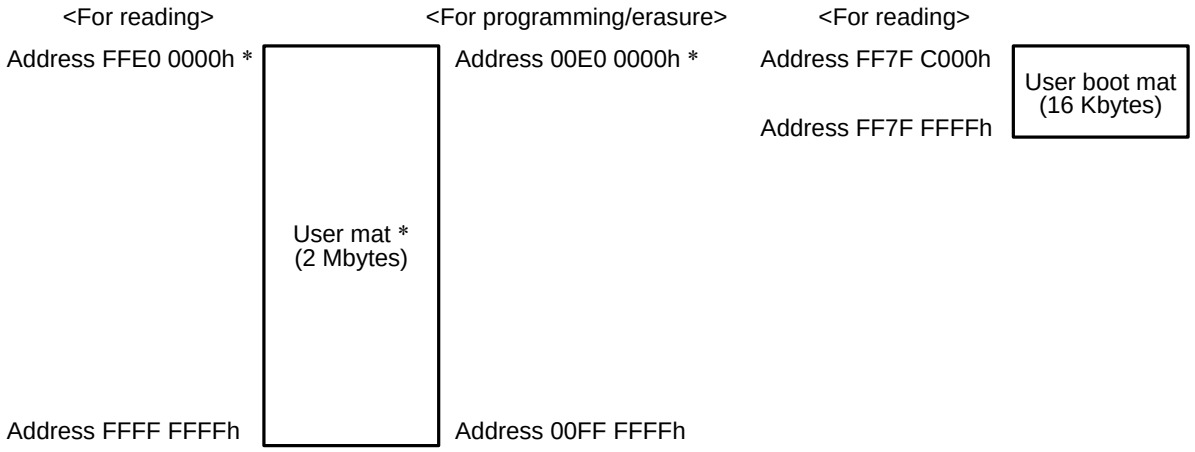


Downloaded from Arrow.com.

## 26.3 Configuration of Memory Mats for the ROM

The ROM of products in the RX610 Group is configured of a maximum 2-Mbyte user mat and a 16-Kbyte user boot mat. The address ranges occupied by these mats are shown in figure 26.2.

Note that for the user mat, the address range for reading differs from the address range for programming and erasure.




Note: * The read addresses and programming/erasure addresses differ in each product.

| Product Code | Read Address | Programming/Erasure Address | Memory Space |
|---|---|---|---|
| R5F56108 | FFE0 0000h | 00E0 0000h | 2 Mbytes |
| R5F56107 | FFE8 0000h | 00E8 0000h | 1.5 Mbytes |
| R5F56106 | FFF0 0000h | 00F0 0000h | 1 Mbyte |
| R5F56104 | FFF4 0000h | 00F4 0000h | 768 Kbytes |

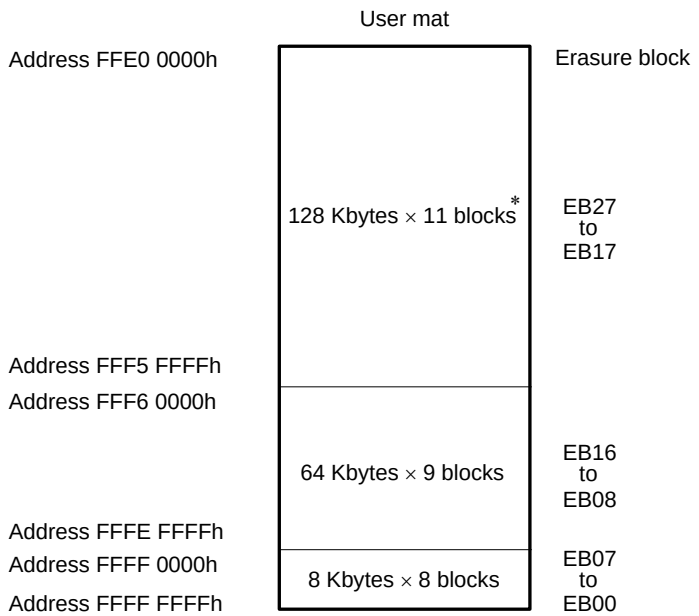
**Figure 26.2 Memory Mat Configuration of ROM**

## 26.4 Block Configuration

The configuration of erasure blocks for the user mat is shown in figure 26.3. As units of erasure, the user mat is divided into 8 blocks of 8 Kbytes each, 9 blocks of 64 Kbytes each, and 11 blocks of 128 Kbytes each. For programming, the user mat consists of the 256-byte units starting from 00h as the lower-order byte of the address.



Downloaded from Arrow.com.




Note: * The erasure blocks differ in each product.

| Product Code | Address | Number of 128-Kbyte Erasure Blocks | Erasure Blocks |
|---|---|---|---|
| R5F56108 | FFE0 0000h | 11 | EB17 to EB27 |
| R5F56107 | FFE8 0000h | 7 | EB17 to EB23 |
| R5F56106 | FFF0 0000h | 3 | EB17 to EB19 |
| R5F56104 | FFF4 0000h | 1 | EB17 |

**Figure 26.3 Configuration of Erasure Blocks for the User Mat**

## 26.5 Operating Modes Associated with the ROM

Figure 26.4 is a diagram of the operating-mode transitions for the RX610 Group.

On release from the reset state, transitions are in accord with the levels on the MD0 and MD1 pins, as shown in figure 26.4.

For more information on the connections between the settings of the levels on the MD0 and MD1 pins and the operating mode for the RX610 Group, refer to section 3, Operating Modes.



Downloaded from Arrow.com.

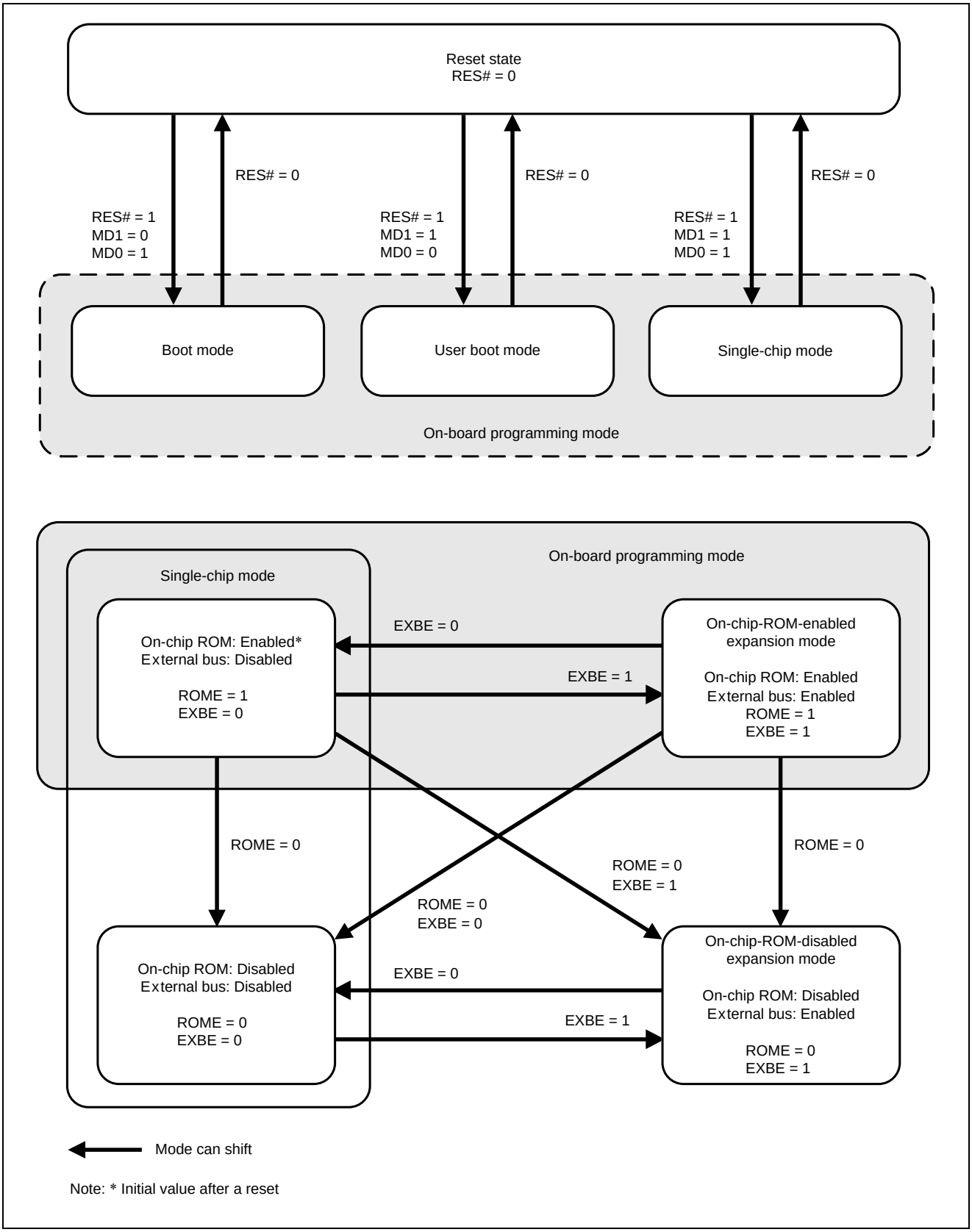



**Figure 26.4 Transitions between Operating Modes in Terms of the ROM**



Downloaded from Arrow.com.

Reading, programming, and erasing of the ROM in an on-board device can proceed if the device is in boot, user-boot, or single-chip mode (with on-chip ROM enabled), or in on-chip-ROM-enabled expansion mode.

Which mats are programmable and erasable, the mat from which booting up proceeds after a reset, etc., differs with the mode. The differences between modes are indicated in table 26.6.

**Table 26.6 Differences between Modes**

| Item | Boot Mode | User Boot Mode | Single-Chip Mode (with On-chip ROM Enabled) or On-chip-ROM-Enabled Expansion Mode |
|---|---|---|---|
| Environment for programming and erasure | On-board programming | On-board programming | On-board programming |
| Programmable and erasable mat | User mat<br>User boot mat | User mat | User mat |
| Division into erasure blocks | Possible*1 | Possible | Poss ble |
| Target mat for booting after a reset | Mat containing the embedded program*2 | User boot mat | User mat |

Notes: 1. The entire ROM may be erased at the time of booting up. Specified blocks can subsequently be erased. For details, refer to section 26.10.2, ID Code Protection.

2. Not available to users.

- Programming and erasure of the user boot mat are only possible in boot mode.
- In boot mode, a host is able to program or read out the user mat, user boot mat, or data mat via an SCI.
- After booting up from the user boot mat in user boot mode, programming and erasure of the user mat or data mat via the desired interface is possible.
- In boot mode, on-chip RAM is employed for the embedded program for use in boot mode. For this reason, preserving the contents of on-chip RAM is not possible in this case.



Downloaded from Arrow.com.

## 26.6 Programming and Erasing the ROM

The ROM is programmed and erased by issuing commands to a dedicated sequencer (FCU) for programming and erasure. The FCU has five modes. For programming and erasure, the mode is changed and then commands for programming and erasure are issued.

The mode transitions required to program or erase the ROM and the system of commands are described below. The descriptions apply in common to boot, user-boot, and single-chip mode (with on-chip ROM enabled), and to on-chip-ROM-enabled expansion mode.

### 26.6.1 FCU Modes

The FCU has five modes or sets of modes. Transitions between modes are caused by modifying FENTRYR or issuing FCU commands. Figure 26.5 is a diagram of the FCU mode transitions.

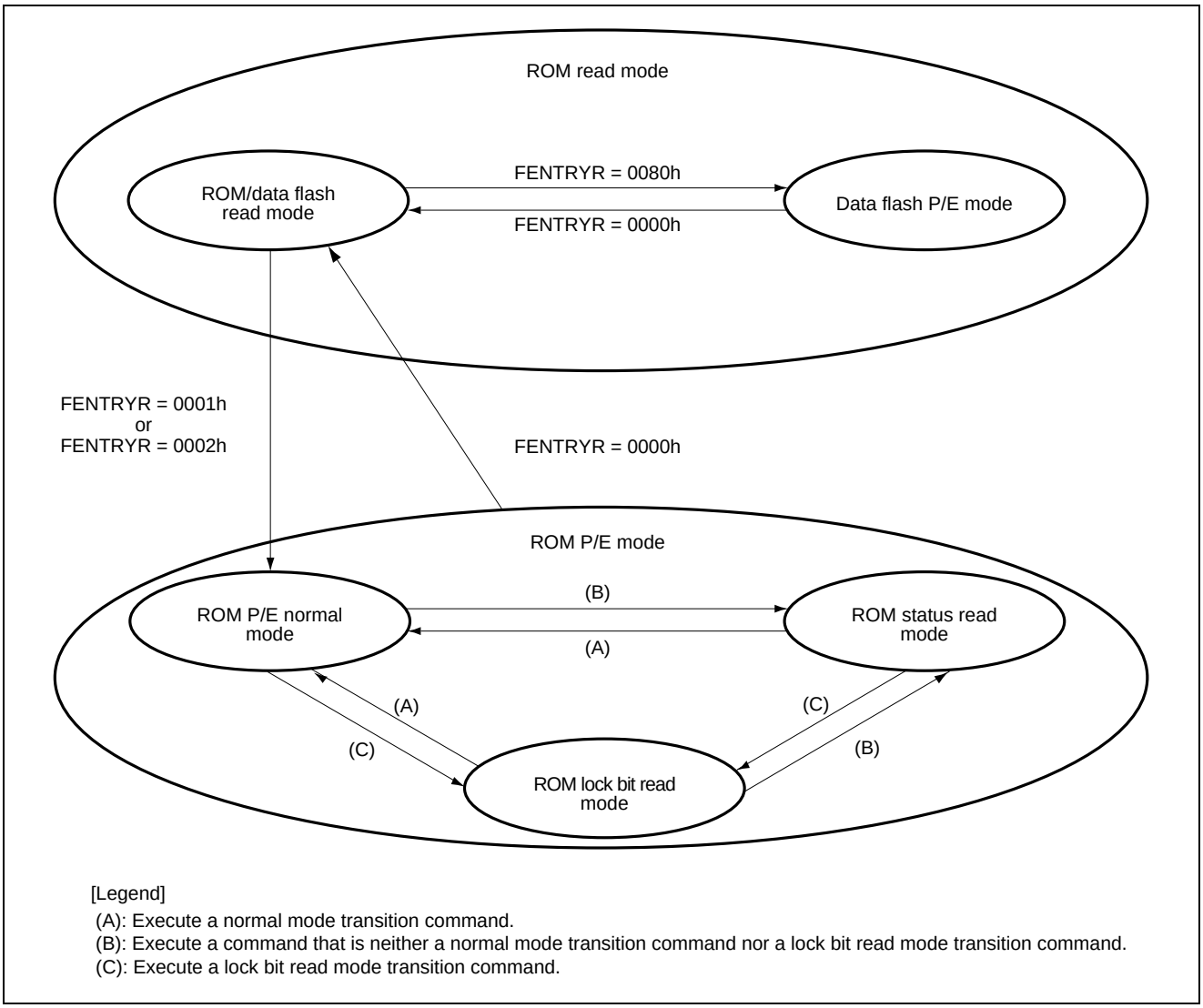



**Figure 26.5 Mode Transitions of the FCU (Associated with the ROM)**



Downloaded from Arrow.com.

### 26.6.1.1 ROM Read Modes

The ROM read modes are for high-speed reading of the ROM. Access to an address for reading can be accomplished in one cycle of ICLK.

ROM/data-flash read mode and data-flash P/E mode are the two kinds of ROM reading mode.

#### (1) ROM/Data Flash Read Mode

This mode is for reading the ROM and data-flash memory. The FCU does not accept commands.

The FCU enters this mode when the FENTRY1* and FENTRY0 bits in FENTRYR are set to 0, and the FENTRYD bit in FENTRYR is set to 0.

#### (2) Data Flash P/E Modes

These modes are for programming and erasure of the data-flash memory. High-speed reading of the ROM is also possible. Although the FCU accepts FCU commands related to the data-flash memory in this mode, it does not accept FCU commands related to the ROM. The FCU enters these modes when the FENTRY1* and FENTRY0 bits in FENTRYR are set to 0, or the FENTRYD bit in FENTRYR is set to 1.

For details on the data flash P/E modes, see section 27.6.1, FCU Modes

### 26.6.1.2 ROM P/E Modes

The ROM P/E modes are for programming and erasure of the ROM. High-speed reading of the ROM is not possible in these modes. Read access to an address within the range for reading causes a ROM-access violation, and the FCU enters the command-locked state (see section 26.8.2, Error Protection).

ROM P/E normal mode, ROM status read mode, and ROM lock-bit read mode are the three ROM P/E modes.

#### (1) ROM P/E Normal Mode

The transition to ROM P/E normal mode is the first transition in the process of programming or erasing the ROM. The FCU enters this mode when the FENTRYD bit in FENTRYR is set to 0, and either the FENTRY1* or FENTRY0 bit in FENTRYR is set to 1 in ROM read mode, or when the normal mode transition command is received in ROM P/E modes. Table 26.9 lists the acceptable commands in this mode.

Read access to an address within the range for programming and erasure while the FENTRY1* and FENTRY0 bits in FENTRYR are set to 1 causes a ROM-access violation, and the FCU enters the command-locked state (see section 26.8.2, Error Protection).

#### (2) ROM Status Read Mode

The ROM status read mode is for reading information on the state of the ROM. The FCU enters this mode when a command other than the normal mode transition and lock bit read mode transition command is received in ROM P/E modes.

ROM status read mode encompasses the states where the FRDY bit in FSTATR0 is 0 and the command-locked state after an error has occurred. Table 26.9 lists the acceptable commands in this mode.

Read access to an address within the range for programming and erasure while the FENTRY1* and FENTRY0 bits in FENTRYR are 1 will actually read out the value of FSTATR0.



Downloaded from Arrow.com.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

### (3) ROM Lock-Bit Read Mode

The ROM lock-bit read mode is for reading the values of the lock bits of the ROM. The FCU enters this mode when a lock-bit read mode transition command is received in ROM P/E modes. Table 26.9 lists the acceptable commands in this mode.

In read access to an address within the range for programming and erasure while the FENTRY1* and FENTRY0 bits in FENTRYR are 1, all bits of the value read out have the value of the lock bit of the erasure block that includes the accessed address.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

## 26.6.2 FCU Commands

FCU commands consist of commands for mode transitions of the FCU and commands for programming and erasure. Table 26.7 lists the FCU commands for use with the ROM.

**Table 26.7 FCU Commands for Use with the ROM**

| Command | Description |
|---|---|
| P/E normal mode transition | Changes the mode to normal mode (see section 26.6.3, Connections between FCU Modes and Commands) |
| Status read mode transition | Changes the mode to status read mode (see section 26.6.3, Connections between FCU Modes and Commands) |
| Lock bit read mode transition (lock bit read 1) | Changes the mode to lock bit read mode (see section 26.6.3, Connections between FCU Modes and Commands) |
| Peripheral clock notification | Sets the frequency of the peripheral clock |
| Programming | ROM programming (in 256-byte units) |
| Block erasure | ROM erasure (in block units, with the lock bit being erased simultaneously) |
| P/E suspension | Suspends programming/erasure |
| P/E resumption | Resumes programming/erasure |
| Status register clearing | Clears the ILGLERR, ERSERR and PRGERR bits in FSTATR0 and releases the FCU from the command locked state |
| Lock bit read 2/blank checking | Reads the lock bit of a specified erasure block (the value of the lock bit is reflected in the FLOCKST bit of FSTATR1)/blank checking of the data-flash memory |
| Lock bit programming | Programs the lock bit of a specified erasure block |

The lock bit read 2 command is also used as the blank-checking command for the data-flash memory. That is, when a lock bit read 2 command is issued for the data flash, blank checking is executed for the data-flash memory (see section 27, Data Flash (Flash Memory for Data Storage)).

Commands for the FCU are issued by write access to addresses within the range for programming and erasure. Table 26.8 shows the formats of the FCU commands. Write access as listed in table 26.8 and in accord with certain conditions causes the FCU to execute processing for the corresponding command. For details on the conditions for the acceptance of the individual FCU commands, see section 26.6.3, Connections between FCU Modes and Commands. For how to use the FCU commands, see section 26.6.4, FCU Command Usage.



Downloaded from Arrow.com.

**Table 26.8 FCU Command Formats**

| Command | Number of bus cycles | First Cycle | | Second Cycle | | Third Cycle | | 4th to 5th Cycles | | 6th Cycle | | 7th to 130th Cycles | | 131st Cycle | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Address | Data | Address | Data | Address | Data | Address | Data | Address | Data | Address | Data | Address | Data |
| P/E normal mode transition | 1 | RA | FFh | — | — | — | — | — | — | — | — | — | — | — | — |
| Status read mode transition | 1 | RA | 70h | — | — | — | — | — | — | — | — | — | — | — | — |
| Lock bit read mode transi ion (lock bit read 1) | 1 | RA | 71h | — | — | — | — | — | — | — | — | — | — | — | — |
| Peripheral clock notification | 6 | RA | E9h | RA | 03h | RA | 0F0Fh | RA | 0F0Fh | RA | D0h | — | — | — | — |
| Programming | 131 | RA | E8h | RA | 80h | WA | WDn | RA | WDn | RA | WDn | RA | WDn | RA | D0h |
| Block erasure | 2 | RA | 20h | BA | D0h | — | — | — | — | — | — | — | — | — | — |
| P/E suspension | 1 | RA | B0h | — | — | — | — | — | — | — | — | — | — | — | — |
| P/E resump ion | 1 | RA | D0h | — | — | — | — | — | — | — | — | — | — | — | — |
| Status register clearing | 1 | RA | 50h | — | — | — | — | — | — | — | — | — | — | — | — |
| Lock bit read 2 | 2 | RA | 71h | BA | D0h | — | — | — | — | — | — | — | — | — | — |
| Lock bit programming | 2 | RA | 77h | BA | D0h | — | — | — | — | — | — | — | — | — | — |

[Legend] Address column RA: ROM programming/erasure address
When the FENTRY0 bit in FENTRYR is 1: An address from 00F0 0000h to 00FF FFFFh
When the FENTRY1* bit in FENTRYR is 1: An address from 00E0 0000h to 00EF FFFFh
WA: ROM programming-destination address
Start address for programming of 256 bytes of data
BA: ROM erasure block address
An address within the target erasure block (specified as an address in the range for programming and erasure)
Data column WDn: nth word of data for programming (n = 1 to 128)

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

### 26.6.3 Connections between FCU Modes and Commands

The sets of FCU commands that can be accepted in each of the FCU modes are fixed. Furthermore, which commands are acceptable in a given FCU mode also depends on the state of the FCU.

Issuing of an FCU command must follow checking of the FCU's state after transitions of the FCU mode.

Commands that are acceptable in the various FCU modes and states are listed in table 26.9. Issuing a command that is not currently acceptable leads to the FCU being placed in the command-locked state (see section 26.8.2, Error Protection).

Issuing of an FCU command must follow checking of the values of the FRDY, ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and of the FCUERR bit in FSTATR1 after transitions of the FCU mode. Furthermore, the CMDLK bit in FASTAT can be checked to see if an error has occurred. The value of the CMDLK bit in FASTAT is the logical OR of the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and the FCUERR bit in FSTATR1.

Table 26.9 Acceptable Commands and the State and Mode (ROM P/E Mode) of the FCU

| | P/E Normal Mode | | | Status Read Mode | | | | | | Lock-Bit Read Mode | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Programming suspended | Erasure suspended | Other state | Programming or erasure | Processing to suspend programming or erasure | Lock bit read 2 processing | Programming suspended | Erasure suspended | Command-locked state | Other state | Programming suspended | Erasure suspended | Other state |
| FSTATR0.FRDY bit | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0/1 | 1 | 1 | 1 | 1 |
| FSTATR0.SUSRDY bit | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| FSTATR0.ERSSPD bit | 0 | 1 | 0 | 0 | 0/1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| FSTATR0.PRGSPD bit | 1 | 0 | 0 | 0 | 0/1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| FASTAT.CMDLK bit | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Normal mode transition | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Status read transition | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Lock-bit read transition (lock bit read 1) | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Peripheral clock setting | X | X | A | X | X | X | X | X | X | A | X | X | A |
| Programming | X | * | A | X | X | X | X | * | X | A | X | * | A |
| Block erasure | X | X | A | X | X | X | X | X | X | A | X | X | A |
| P/E suspension | X | X | X | A | X | X | X | X | X | X | X | X | X |
| P/E resumption | A | A | X | X | X | X | A | A | X | X | A | A | X |
| Status register clearing | A | A | A | X | X | X | A | A | A | A | A | A | A |
| Lock bit read 2 | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Lock bit programming | X | * | A | X | X | X | X | * | X | A | X | * | A |

[Legend]

A: Acceptable

*: Only programming is acceptable for blocks other than the block where erasure was suspended

X: Not acceptable



Downloaded from Arrow.com.

### 26.6.4 FCU Command Usage

The set of FCU commands consists of commands for FCU mode transitions, actually programming or erasing the ROM, error processing, and suspension and resumption. The following passages describe the various commands. For a description of the modes and states where the respective commands are acceptable, see section 26.6.3, Connections between FCU Modes and Commands.

#### 26.6.4.1 Mode Transitions

This subsection covers the commands for mode transitions. For an illustration of the various transitions between modes, see figure 26.5.

**(1) Switching to ROM P/E Mode**

A transition to ROM P/E mode is required before executing an FCU command for the ROM becomes possible.

Setting the FENTRY0 or FENTRY1* bit in FENTRYR to 1 causes a transition to ROM P/E mode for programming and erasure of the corresponding address range.

Before actually proceeding to program or erase the ROM, enable programming and erasure by writing 01h as a byte to FWEPROR (see section 26.2.15, Flash Write Erase Protection Register (FWEPROR)).

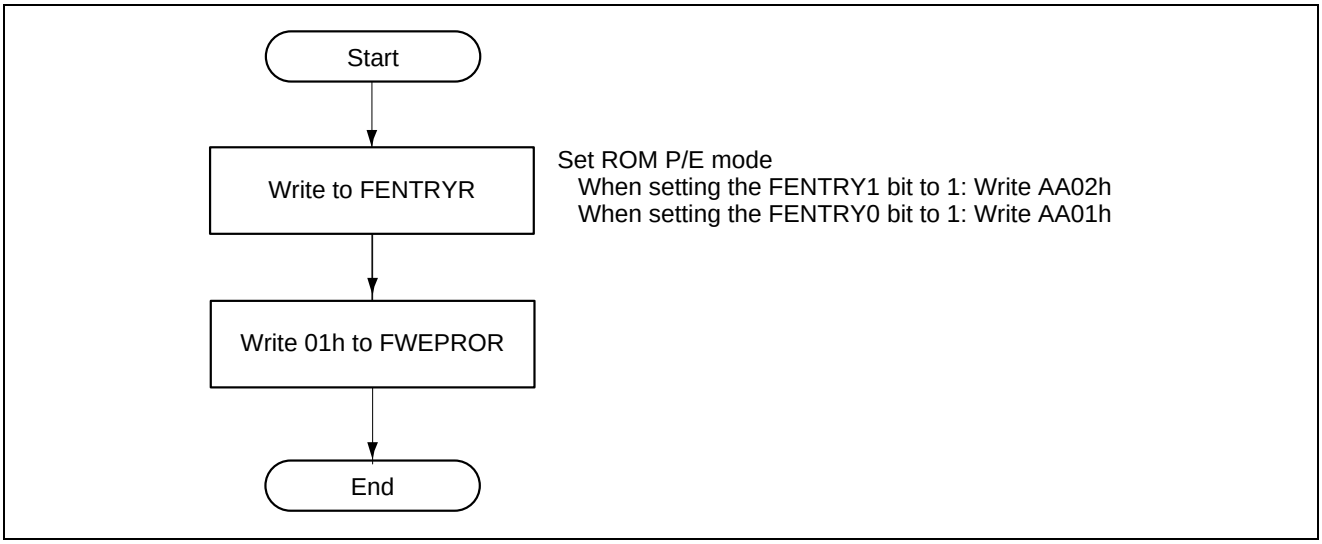



**Figure 26.6 Procedure for Transition to ROM P/E Mode**

**(2) Switching to ROM Read Mode**

High-speed reading of the ROM requires clearing of the FENTRY0 or FENTRY1* bit in FENTRYR, which places the FCU in ROM read mode.

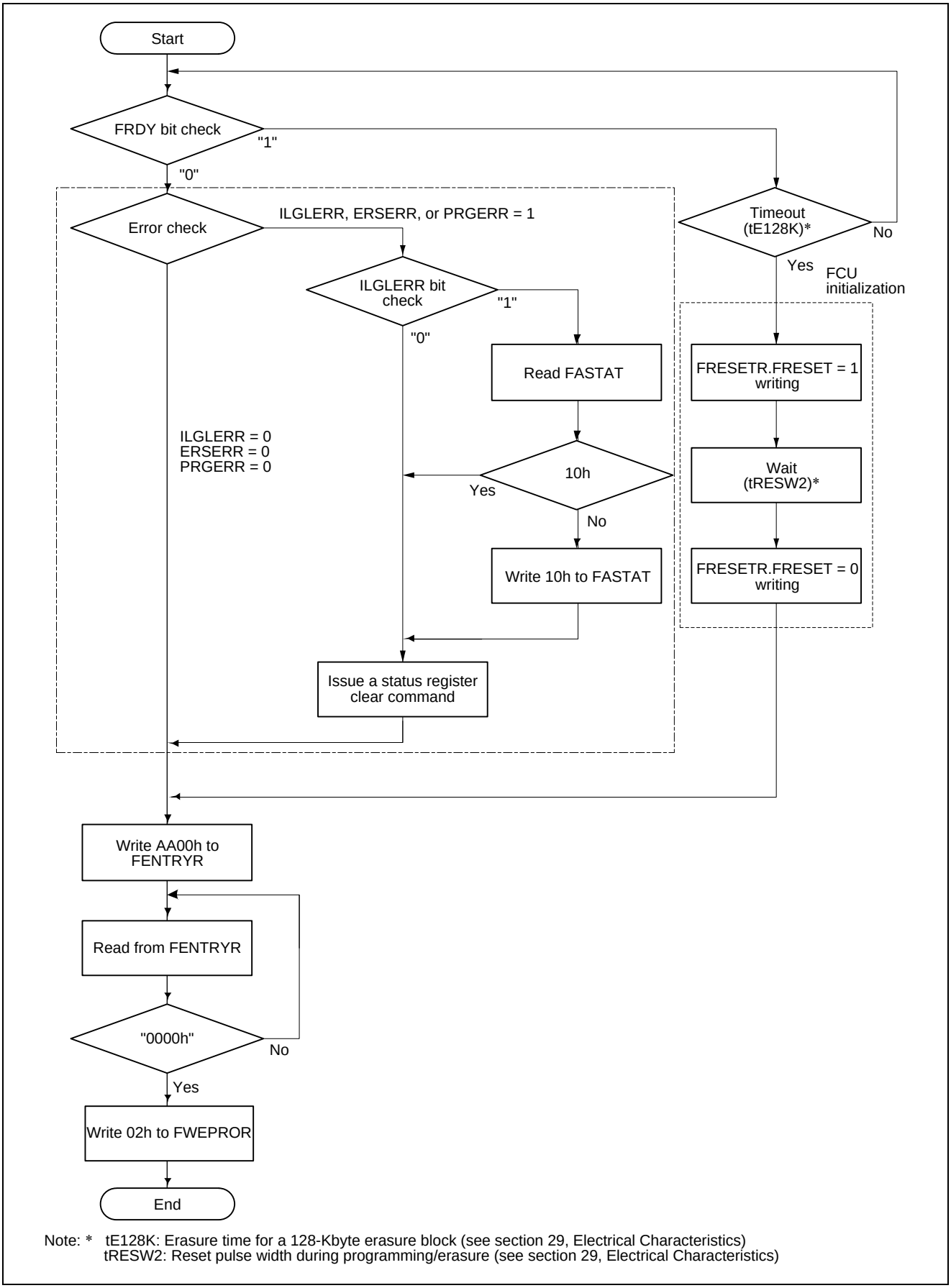
Writing of 02h to FWEPROR is also required to disable programming and erasure (see section 26.2.15, Flash Write Erase Protection Register (FWEPROR).

Before switching the FCU from P/E mode to read mode, ensure that all processing of FCU commands has been completed and that the FCU has not detected an error.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.




Note: * tE128K: Erasure time for a 128-Kbyte erasure block (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.7 Procedure for Transition to ROM Read Mode**



Downloaded from Arrow.com.

### (3) Switching to ROM P/E Normal Mode

Two methods are available for the transition to ROM P/E normal mode: setting the FENTRYR register while the FCU is in ROM read mode (see section 26.6.1, FCU Modes), or issuing the normal mode transition command while the FCU is in ROM P/E mode (see figure 26.8). The normal mode transition command is issued by writing FFh as a byte to an address in the range for programming and erasure of the ROM.

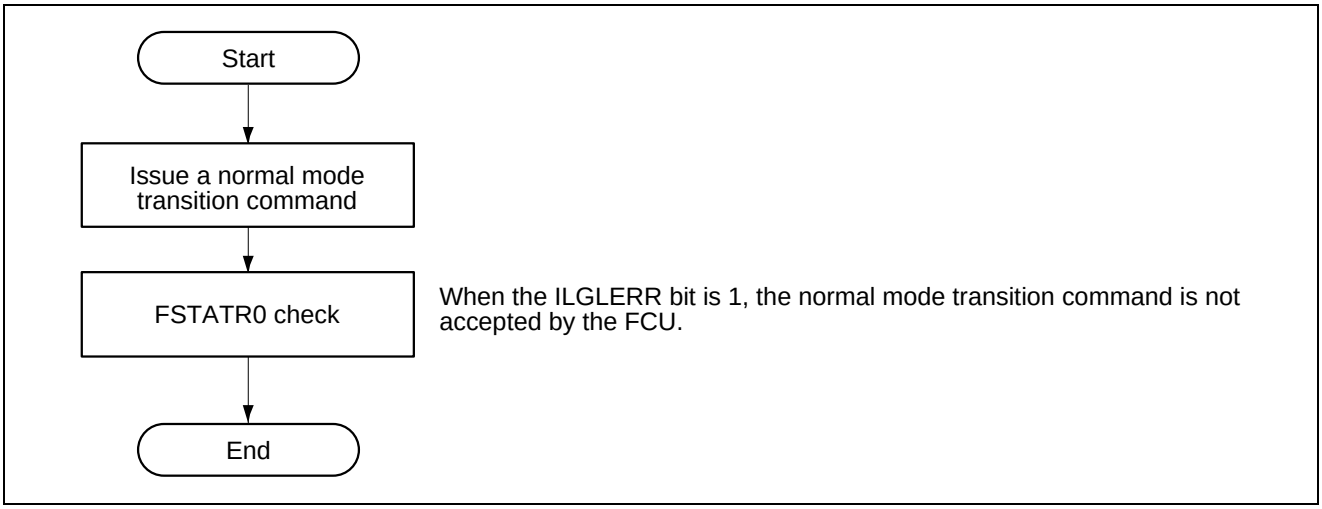



**Figure 26.8 Procedure for Transition to ROM P/E Normal Mode**

### (4) Switching to ROM Status Read Mode

Issuing an FCU command other than a normal mode transition or lock bit read mode transition command places the FCU in ROM status read mode. The same transition can be obtained by issuing the status read mode transition command. Figure 26.9 shows the procedure for checking the register FSTATR0 as an example. In the example, the status read mode transition command is issued to place the FCU in ROM status read mode, and the value of FSTATR0 is obtained by read access to the area for programming and erasure and then checked.

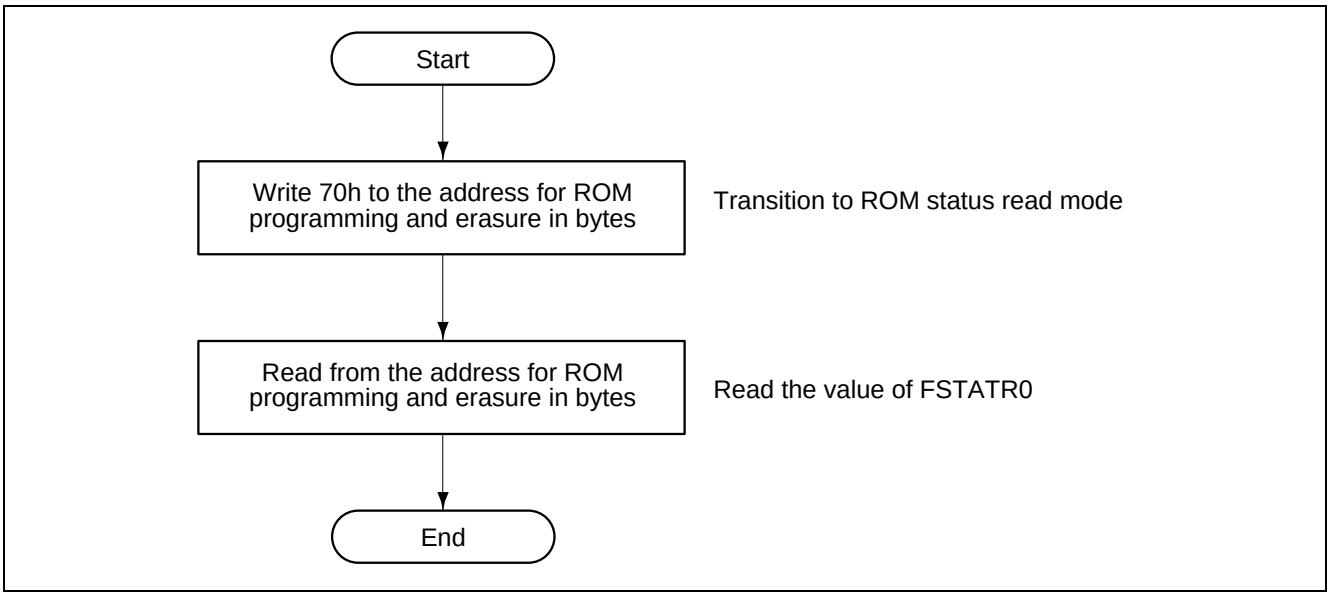



**Figure 26.9 Procedure for Transition to ROM Status Read Mode and the Status Checking**



Downloaded from Arrow.com.

### (5) Switching to ROM Lock-Bit Read Mode

Clearing the FRDMD bit in FMODR (memory area method) issues a lock bit read mode transition (lock bit read 1) command. After the transition to ROM lock bit read mode, lock bit value are obtained by read access to the area for ROM programming and erasure. All bits of a value thus read out have the value of the lock bit of the erasure block that contains the accessed address (figure 26.10).

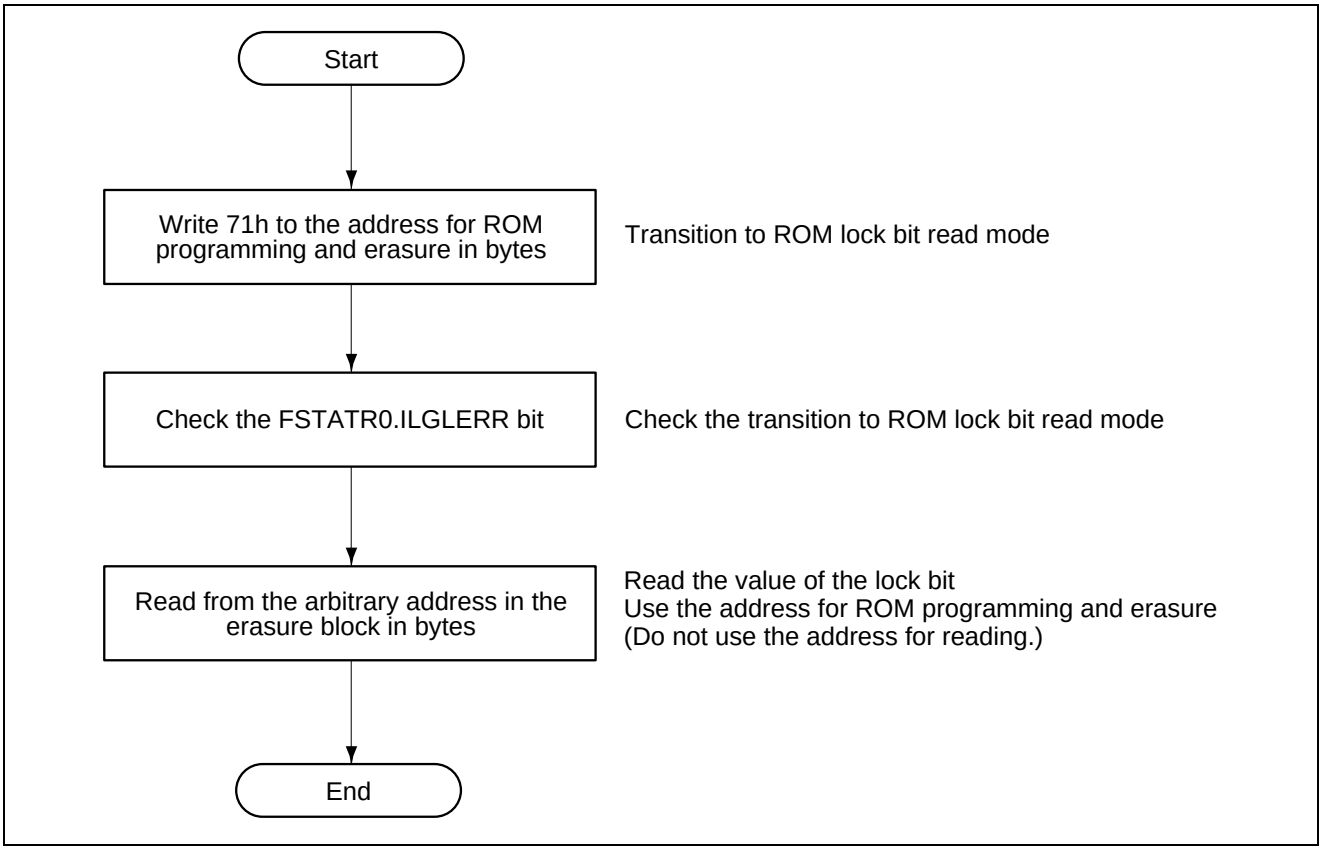



**Figure 26.10 Procedure for Transition to ROM Lock-Bit Read Mode and the Lock-Bit Read Method**

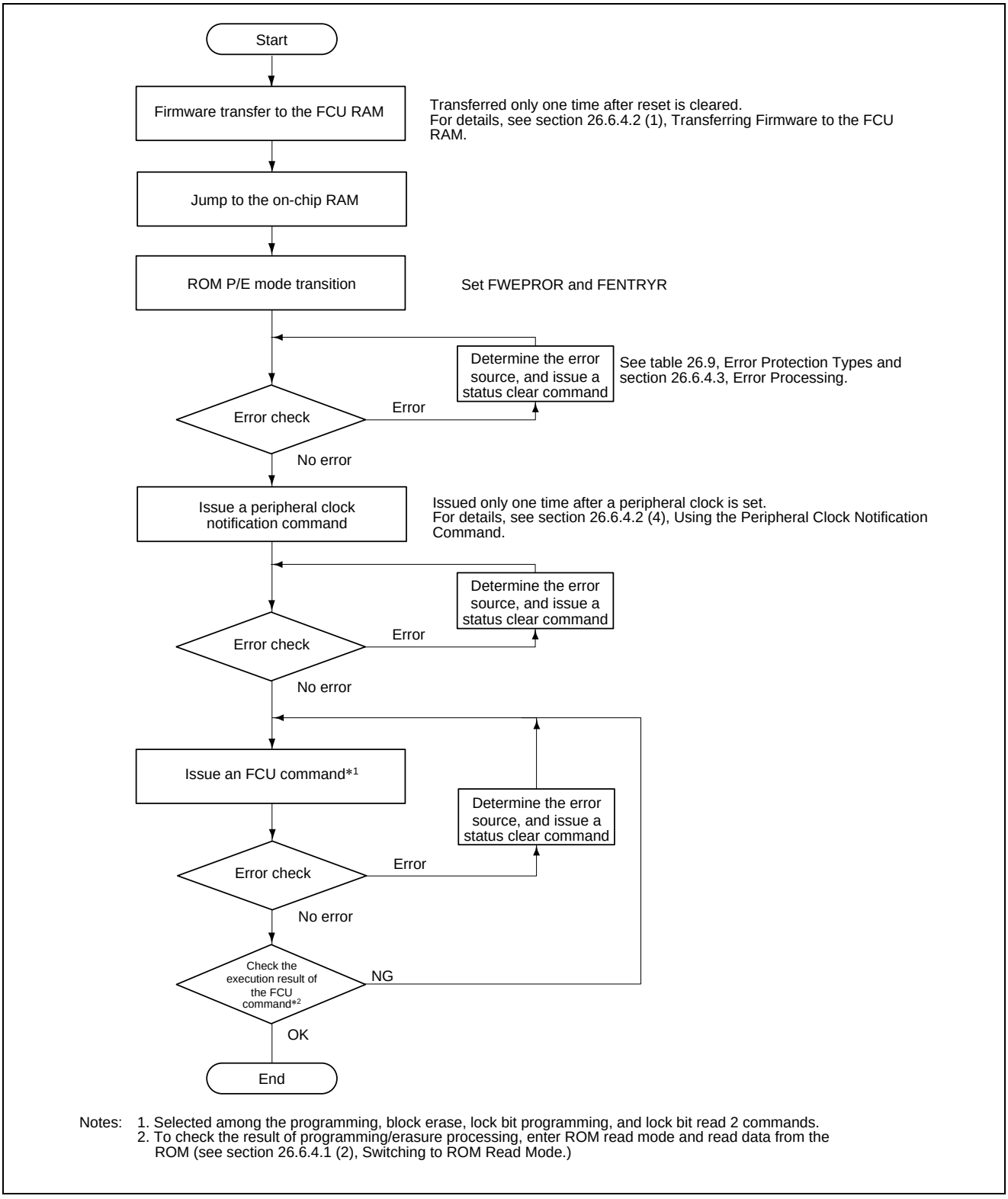


Downloaded from Arrow.com.

### 26.6.4.2 Programming and Erasure Procedures

The following passages describe the flow of procedures for programming or erasing the ROM. For details on the acceptance of commands by the FCU, see section 26.6.3, Connections between FCU Modes and Commands.

Figure 26.11 is a simple flowchart of the procedure for executing FCU commands.




Notes: 1. Selected among the programming, block erase, lock bit programming, and lock bit read 2 commands.
2. To check the result of programming/erasure processing, enter ROM read mode and read data from the ROM (see section 26.6.4.1 (2), Switching to ROM Read Mode.)

**Figure 26.11 Simple Flowchart of the Procedure for Programming and Erasure**



Downloaded from Arrow.com.

### (1) Transferring Firmware to the FCU RAM

FCU commands can only be used if the FCU RAM holds the firmware for the FCU. The FCU RAM does not hold the FCU firmware immediately after the chip has been booted up, so the firmware must be copied from the FCU firmware area to the FCU RAM. Furthermore, when the FCUERR bit in FSTATR1 is set to 1, the FCU must be reset and the firmware recopied because the firmware stored in the FCU RAM may have been corrupted.

Figure 26.12 shows the flow of the procedure for transferring firmware to the FCU RAM. Before writing data to the FCU RAM, set FENTRYR to 0000h and stop the FCU. See section 12, DMA Controller (DMAC), for information on setting up the DMAC to handle data transfer.

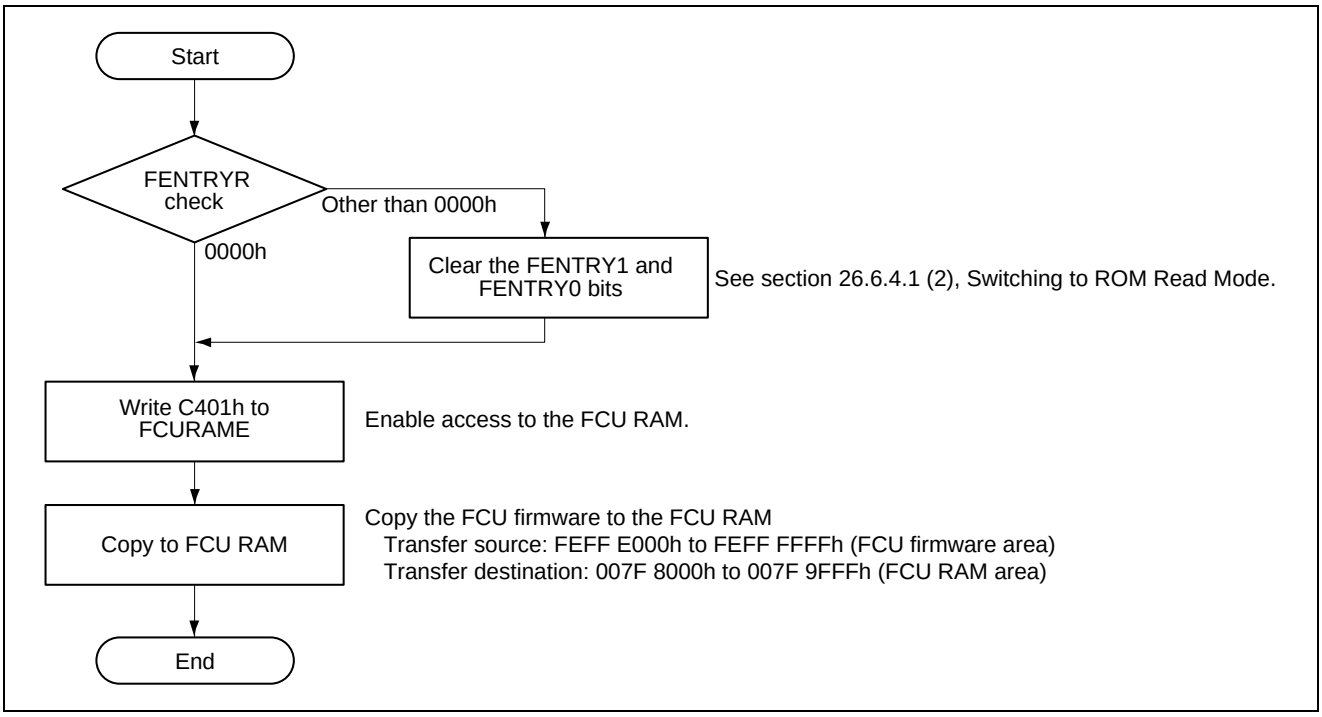



**Figure 26.12 Procedure for Firmware Transfer to FCU RAM**

### (2) Jumping to Locations in On-chip RAM

Since fetching instructions from the ROM is not possible while the ROM is being programmed or erased, code has to be executed from an area other than the ROM. Copy the required program code to on-chip RAM and then make execution jump to the address where the code starts in on-chip RAM.

### (3) Transition to ROM P/E Mode

The FCU is placed in ROM P/E mode by the settings of the FENTRY1* or FENTRY0 bit in FENTRYR and of FWEPROR. Details are given in section 26.6.4.1 (3)Switching to ROM P/E Normal Mode.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

### (4) Using the Peripheral Clock Notification Command

The peripheral clock is used in programming and erasing the ROM, so the frequency of this clock has to be set in the PCKAR. Frequencies in the range from 8 to 50 MHz are selectable. Do not set frequencies out of this range.

The peripheral clock notification command is used after the PCKAR setting has been made. In the first and second cycles for the peripheral clock notification command, respectively, the values E9h and 03h are written to the address range for programming and erasure of the ROM. Word-unit writing is used in the third to fifth cycles of the command. Accordingly, make sure that the addresses used are aligned with four-byte boundaries. After 0F0Fh has been written three times (as a word) to the address range for programming and erasure of the ROM, the process of the FCU setting the frequency of the peripheral clock starts once the value D0h has been written as a byte in the sixth cycle. The FRDY bit in FSTATR0 can be used to check whether or not the settings have been completed.

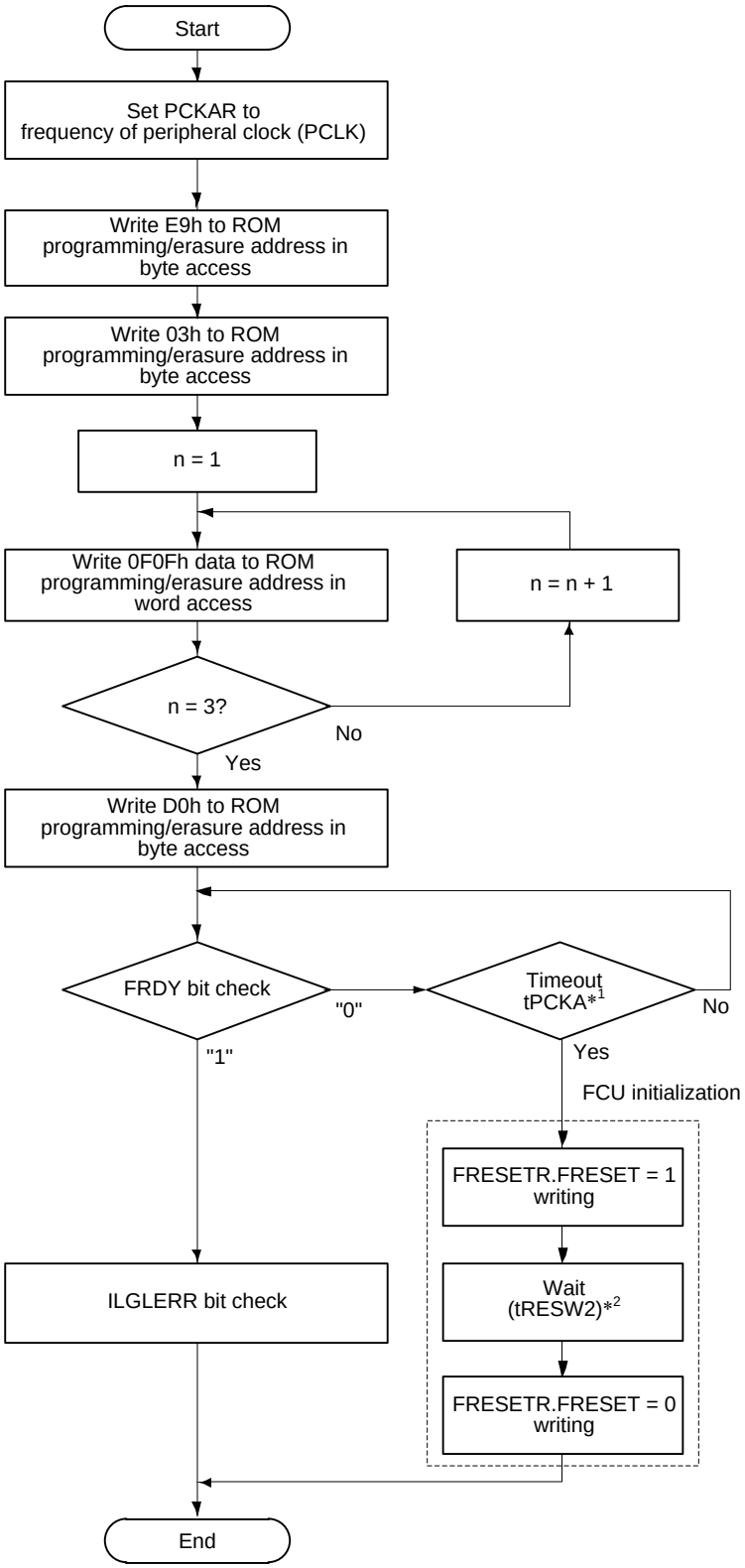
Addresses that can be used in the first to sixth cycles differ according to the settings of the FENTRY0 and FENTRY1* bits in FENTRYR. Ensure that the addresses suit the settings of these bits. If issuing of the command is attempted with an erroneous combination of the setting of the bits and specified addresses, the FCU will detect the error and enter the command-blocked state (see section 26.8.2, Error Protection).

Furthermore, if the setting for the peripheral clock in use will not be changed from this setting after release from the reset state, this setting is also valid for the next FCU command.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.




Notes: 1. tPCKA: 60 µs when PCLK = 50 MHz, 120 µs when PCLK = 25 MHz
2. tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.13 Using the Peripheral Clock Notification Command**



Downloaded from Arrow.com.

### (5) Programming

The programming command is used to write data to the ROM.

In the first and second cycles for the peripheral clock notification command, respectively, the values E8h and 80h are written to the address range for programming and erasure of the ROM. In the third cycle, write the actual data to be programmed, as a word unit, to the target address for programming. For this first address, always use an address that is on a 256-byte boundary. In the fourth to the 130th cycles, write the data for programming in 127 word-unit rounds to the address range for programming and erasure of the ROM. Once the value D0h has been written as a byte to the address range for programming and erasure of the ROM in the 131st cycle, the FCU begins the actual process of programming the ROM. The FRDY bit in FSTATR0 can be used to check whether or not the programming has been completed.

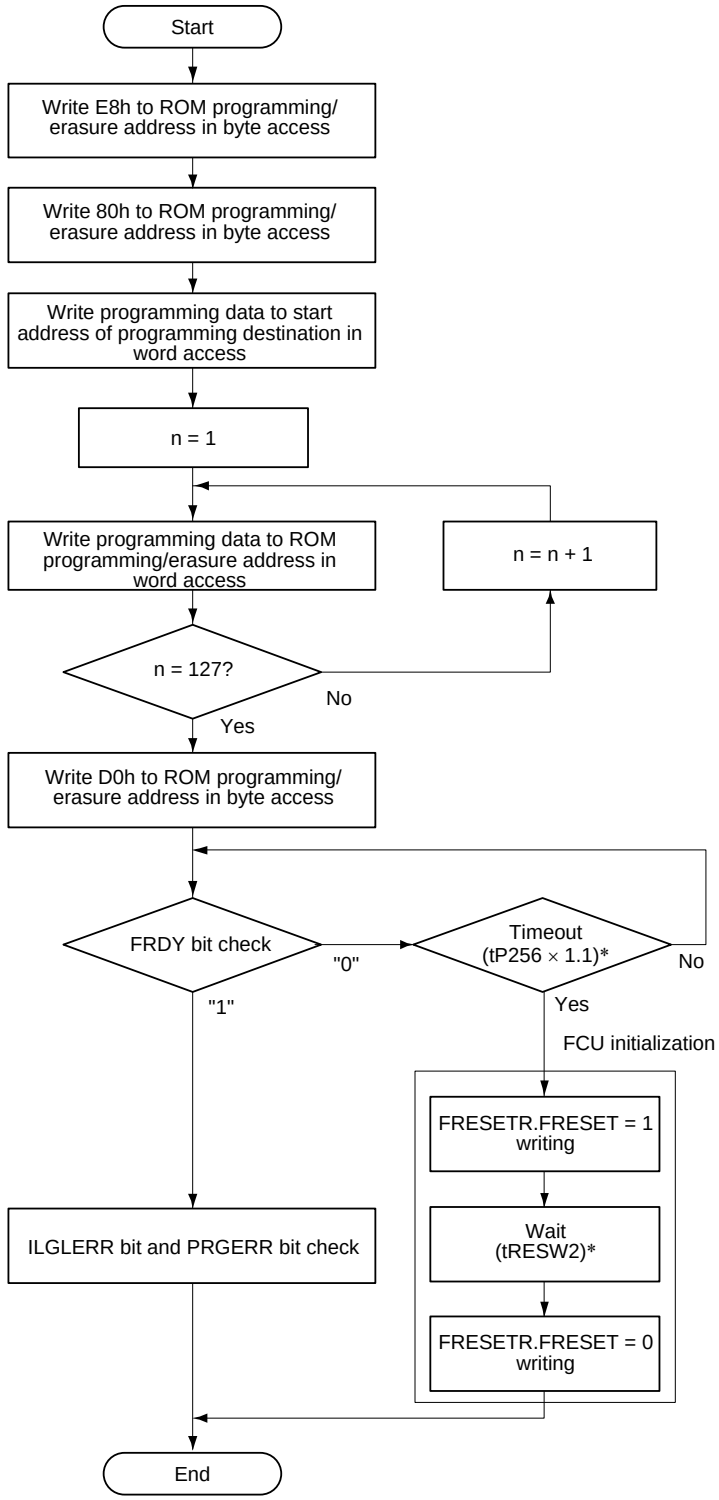
Addresses that can be used in the first to 131st cycles differ according to the settings of the FENTRY0 and FENTRY1* bits in FENTRYR. Ensure that the addresses suit the settings of these bits. If issuing of the command is attempted with an erroneous combination of the setting of the bits and specified addresses, the FCU will detect the error and enter the command-blocked state (see section 26.8.2, Error Protection).

In cases where the target range in the third to 130th cycles includes addresses that do not require programming, use FFFFh as the data for programming to those addresses. To execute a programming with lock bit protection disabled, proceed with programming after setting the FPROTCN bit in FPROTR.

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.




Note: * tP256: Programming time for 256-byte data (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.14 Procedure for ROM Programming**



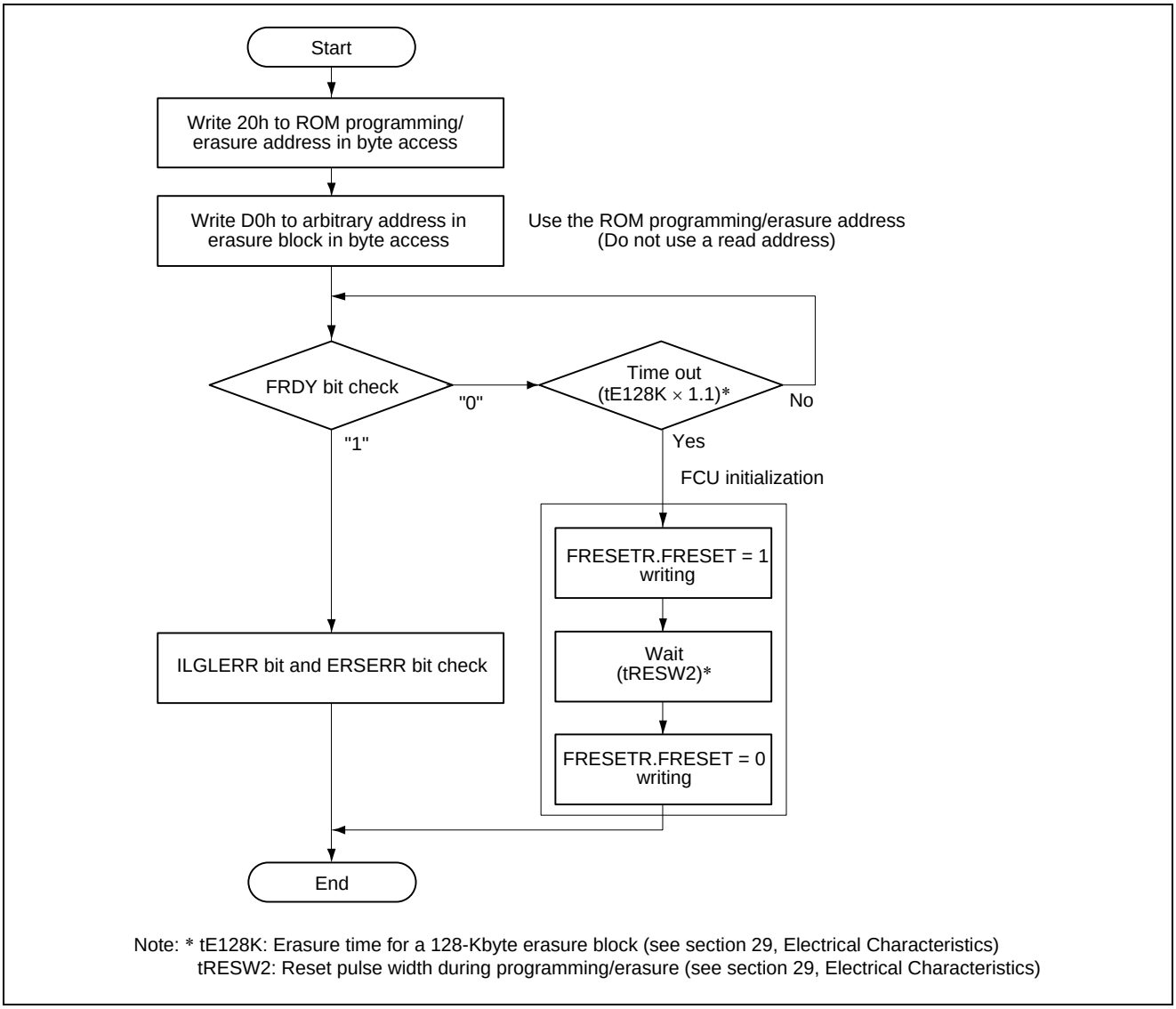
Downloaded from Arrow.com.

### (6) Erasure

To erase data from the ROM, use the block erase command.

Write 20h to the ROM programming/erasure address in byte access in the first cycle of the block erase command. When D0h is written to an arbitrary address in an erasure target block in byte access in the second cycle, the FCU start the erasure processing for the ROM. Whether erasure is completed can be checked with the FRDY bit in FSTATR0.

To execute an erasure with lock bit protection disabled, set the FPROTCN bit in FPROTR before erasure.




Note: * tE128K: Erasure time for a 128-Kbyte erasure block (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.15 Procedure for ROM Erasure**



Downloaded from Arrow.com.

### (7) Writing to/Erasing Lock Bit

Each erasure block in the user mat includes a lock bit. To write to a lock bit, use the lock bit programming command. In the first cycle of the lock bit programming command, 77h is programmed to the ROM programming/erasure address in byte access. When D0h is programmed to an arbitrary address in an erasure block whose lock bit is to be programmed in the second cycle in byte access, the FCU start the programming processing of the lock bit. Whether programming is completed can be checked with the FRDY bit in FSTATR0.

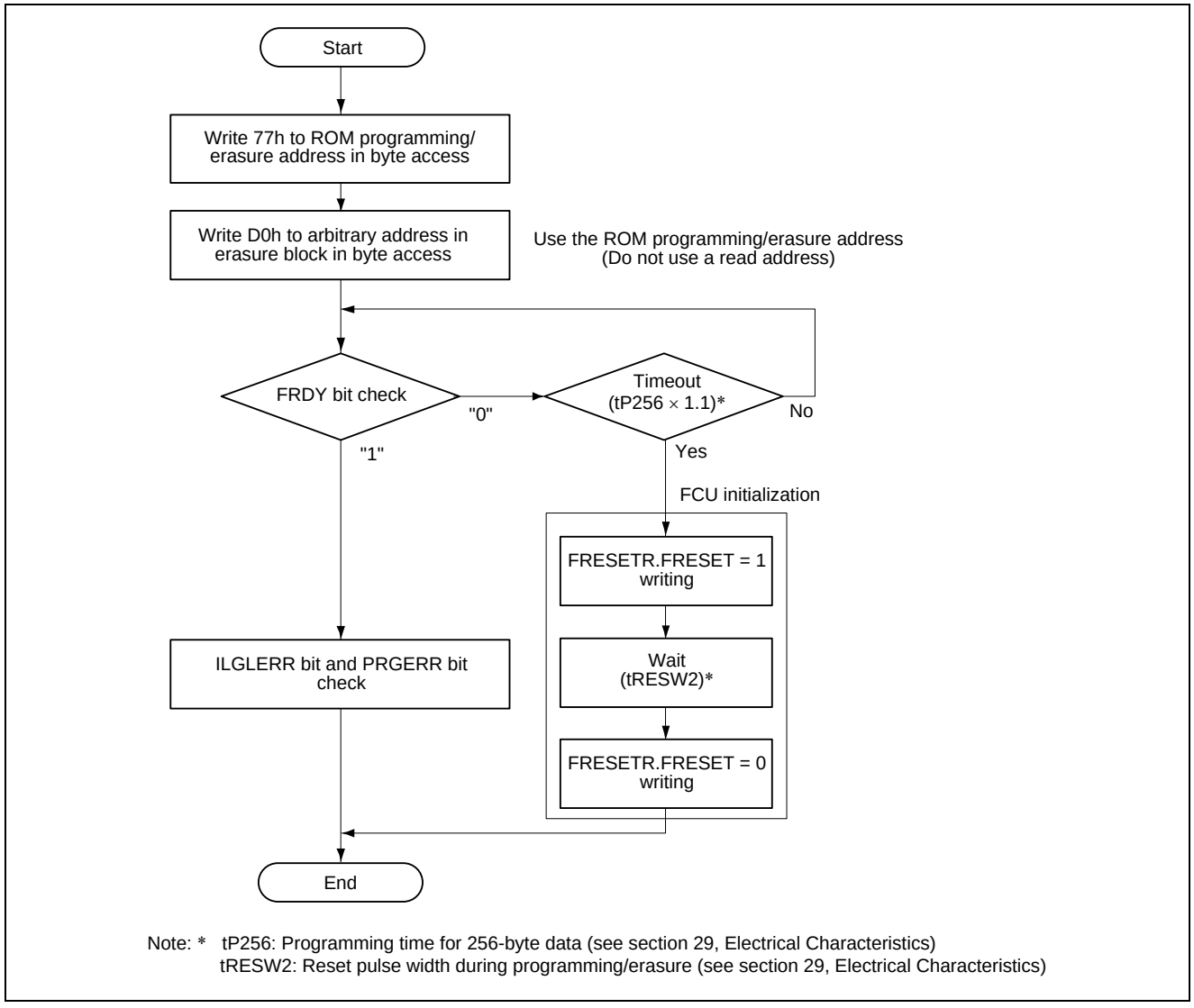



**Figure 26.16 Program Setting of the Lock Bit**

To erase a lock bit, use the block erase command.

When the FPROTCN bit in FPROTR is 0, erasure blocks whose lock bit is set to 0 cannot be erased. When erasing a lock bit, issue a block erase command with the FPROTCN bit set to 1. Using the block erase command erases all data in the erasure block. It is impossible to erase only a lock bit.



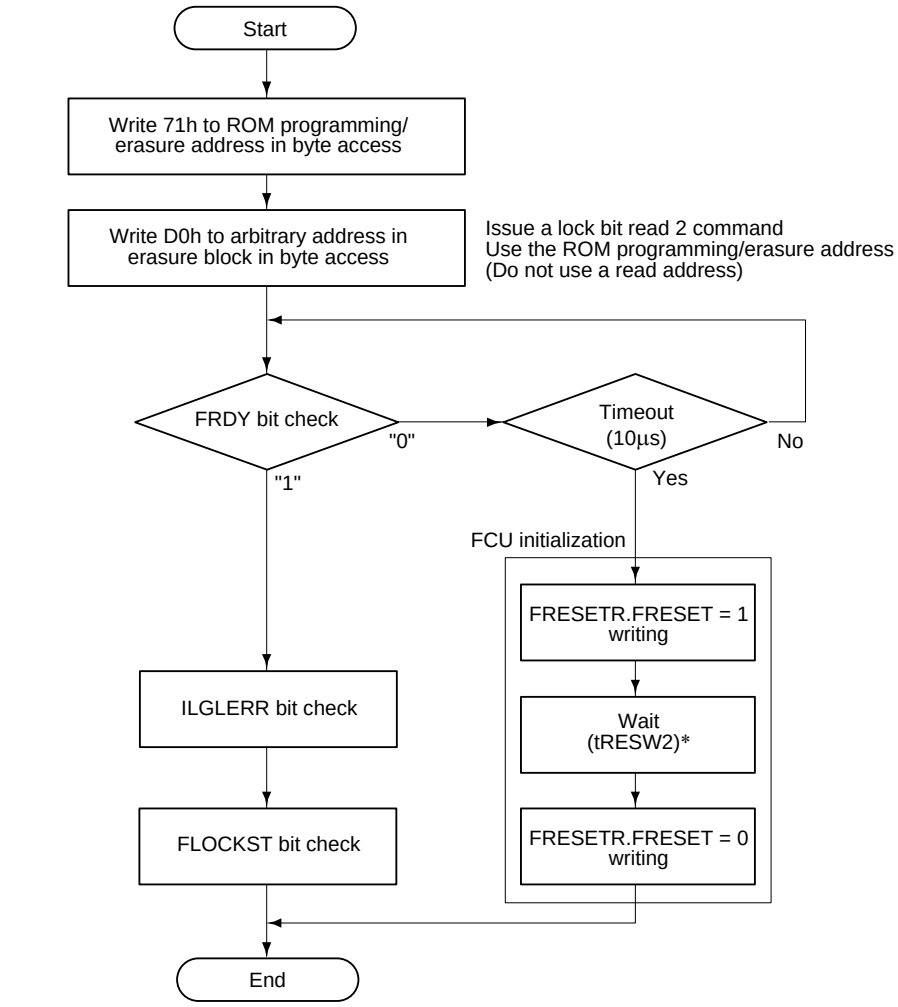
Downloaded from Arrow.com.

### (8) Reading Lock Bits

Lock bits can be read out by either reading from a memory area or reading from a register.

The lock bit read 2 command is issued in the case of the register-reading method (i.e. when the FRDMD bit in FMODR is 1). This command is issued to an address within the block for which the lock bit is to be read out; the address range is that for programming and erasing the ROM. In the first and second cycles of the lock bit read 2 command, the values 71h and D0h are written as bytes; once these values have been written, the value of the lock bit for the specified erasure block is copied to the FLOCKST bit in FSTATR1.

In the case of the memory area reading method (i.e. when the FRDMD bit in FMODR is 0), the FCU is placed in lock-bit reading mode, and the lock bit is obtained by reading from an address within the address range for programming and erasure of the ROM. Details are given in section 26.6.4.1 (5)Switching to ROM Lock-Bit Read Mode.




Note: * tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.17 Procedure for Reading Lock Bit in Register Read Mode**



Downloaded from Arrow.com.

### 26.6.4.3 Error Processing

The following passages describe the flow of error processing. For details on errors, see section 26.8, Protection.

#### (1) Checking Flash Status Register 0 (FSTATR0)

To check FSTATR0, read FSTATR0 directly or read the ROM programming/erasure address in ROM status read mode.

For the reading in ROM status read mode, see section 26.6.4.1 (4)Switching to ROM Status Read Mode.

#### (2) Clearing Flash Status Register 0 (FSTATR0)

To clear the ILGLERR, ERSERR and PRGERR bits in FSTATR0, use the status register clear command.

When one of the ILGLERR, ERSERR and PRGERR bits in FSTATR0 is 1, the FCU is placed in the command-locked state and receives no FCU commands other than the status register clear command. If the ILGLERR is 1, also check the values of the ROMAE, DFLAE, DFLRPE, and DFLWPE bits in FASTAT. Even if issuing a status register clear command without clearing these bits, the ILGLERR bit is not cleared.

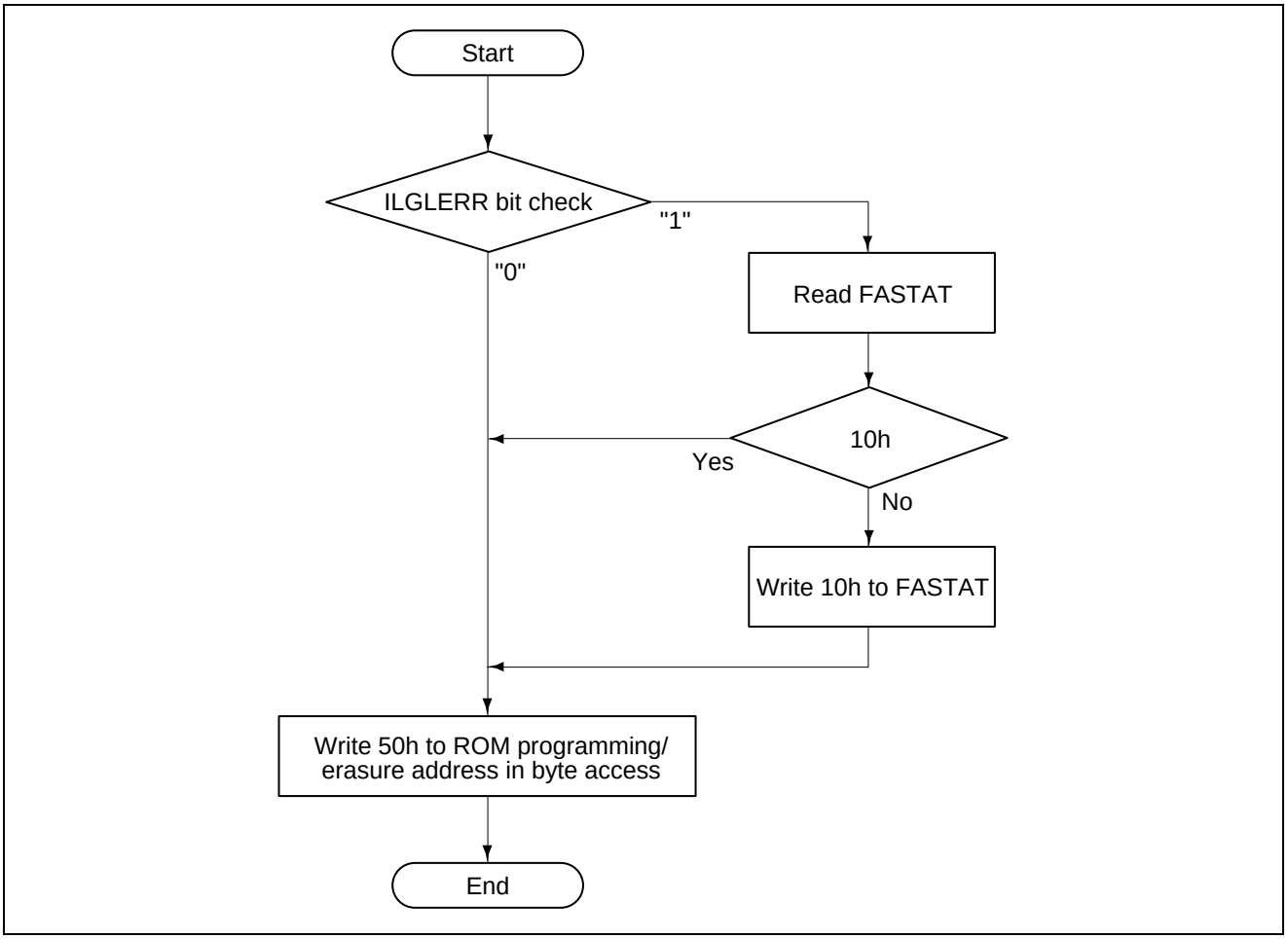



**Figure 26.18 Procedure for Clearing FSTATR0**



Downloaded from Arrow.com.

#### (3) Initializing the FPU

When a timeout leads to the FRDY bit in FSTATR0 not being set to 1 after an FCU command has been issued, FRESETR must be used to initialize the FCU. This is also necessary when the FCUERR bit in FSTATR1 has been set. In either case, maintain the FRESET bit in FRESETR at logical one over a period of at least tRESW2 (see section 28, Electrical Characteristics). Disable reading from the ROM and data-flash memory over this period. In addition, while the FRESET bit is 1, FCU commands are disabled because the FENTRYR register is initialized. Restart the processing from the start, as shown in figure 26.11.

### 26.6.4.4 Suspension and Resumption

#### (1) Suspending Programming or Erasure

To suspend programming/erasure for the ROM, use the P/E suspend command.

When issuing a P/E suspend command, check that the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and the FCUERR bit in FSTATR1 are 0, and the execution of programming/erasure is normally performed. To confirm that the suspend command can be received, also check that the SUSRDY bit in FSTATR0 is 1. After issuing a P/E suspend command, read FSTATR0 and FSTATR1 to confirm that no error occurs.

If an error occurs during programming/erasure, at least one of the ILGLERR, PRGERR, ERSERR, and FCUERR bits is set to 1. When programming/erasure processing has finished during the interval from when it is checked that the SUSRDY bit is 1 to when a P/E suspend command is received, the ILGLERR bit is set to 1 because the issued P/E suspend command is detected as an illegal command.

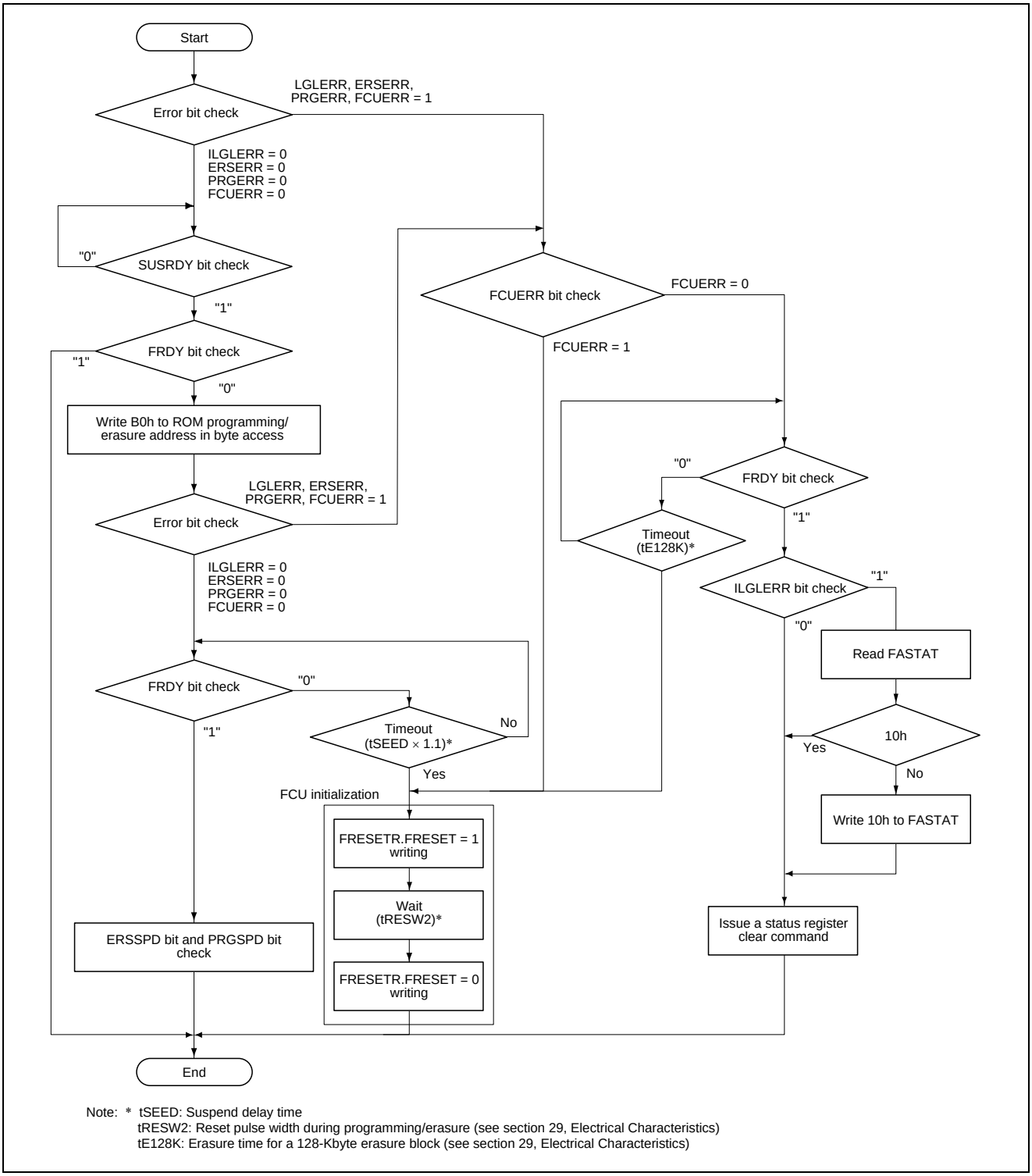
When programming/erasure processing has finished simultaneously with the reception of a P/E suspend command, no error occurs and the suspended state is not entered (the FRDY bit in FSTATR0 is 1 and the ERSSPD and PRGSPD bits in FSTATR0 are 0). When a P/E suspend command is received and then the programming/erasure suspend processing finishes normally, the FCU enters the suspended state, the FRDY bit is set to 1, and the ERSSPD or PRGSPD bit is set to 1. After issuing a P/E suspend command, check that the ERSSPD or PRGSPD bit is 1 and the FCU enters the suspended state, and then decide the subsequent flow. When issuing a P/E resume command in the subsequent flow although the FCU does not enter the suspended state, an illegal command error occurs and the FCU is placed in the command-locked state (see section 26.8.2, Error Protection).

If the erasure suspended state is entered, programming to blocks other than an erasure target can be performed. Additionally, the programming and erasure suspended states can change to ROM read mode by clearing FENTRYR.

For details on FCU operations at the reception of a P/E suspend command, see section 26.7, Suspending Operation.



Downloaded from Arrow.com.




Note: * tSEED: Suspend delay time
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)
tE128K: Erasure time for a 128-Kbyte erasure block (see section 29, Electrical Characteristics)

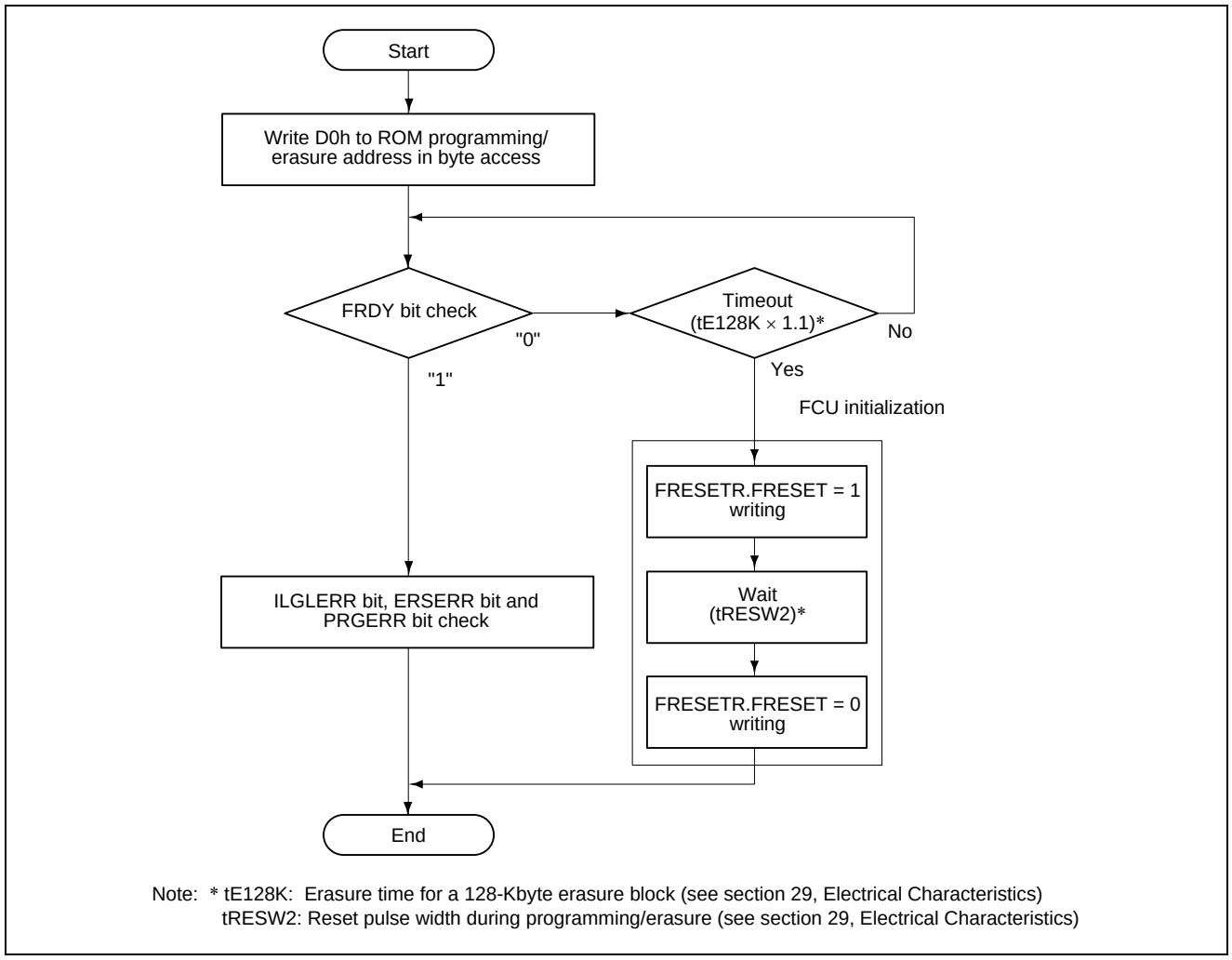
**Figure 26.19 Procedure for Programming/Erasure Suspension**



Downloaded from Arrow.com.

### (2) Resuming Programming or Erasure

To resume a suspended programming/erasure processing, use the P/E resume command. When the settings of FENTRYR are changed during suspension, reset FENTRYR to the value immediately before a P/E suspend command is issued, and then issue a P/E resume command.




Note: * tE128K: Erasure time for a 128-Kbyte erasure block (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 26.20 Procedure for Resuming Programming or Erasure**



Downloaded from Arrow.com.

## 26.7 Suspending Operation

The ROM cannot be read out during programming/erasure. The ROM can be read out by suspending the ROM programming/erasure with the P/E suspend command. The P/E suspend command includes one programming mode and two erasure modes (suspension priority mode and erasure priority mode). The P/E resume command that resumes suspended programming/erasure processing is also provided.

### 26.7.1 Suspension during Programming

When issuing a P/E suspend command during the ROM programming/erasure, the FCU suspends programming processing. Figure 26.21 shows the suspend operation of programming.

When receiving a programming-related command, the FCU clears the FRDY bit in FSTATR0 to 0 to start programming. If the FCU enters the state in which the P/E suspend command can be received after starting programming, the SUSRDY bit in FSTATR0 is set to 1. When a P/E suspend command is issued, the FCU receives the command and clears the SUSRDY bit to 0. If the FCU receives a P/E suspend command while a programming pulse is being applied, the FCU continues applying the pulse. After specified pulse application time, the FCU finishes pulse application, and starts the programming suspend processing and sets the PRGSPD bit in FSTATR0 to 1. When the suspend processing finishes, the FCU sets the FRDY bit to 1 to enter the programming suspended state. If receiving a P/E resume command in the programming suspended state, the FCU clears the FRDY and PRGSPD bits to 0 and resumes programming.

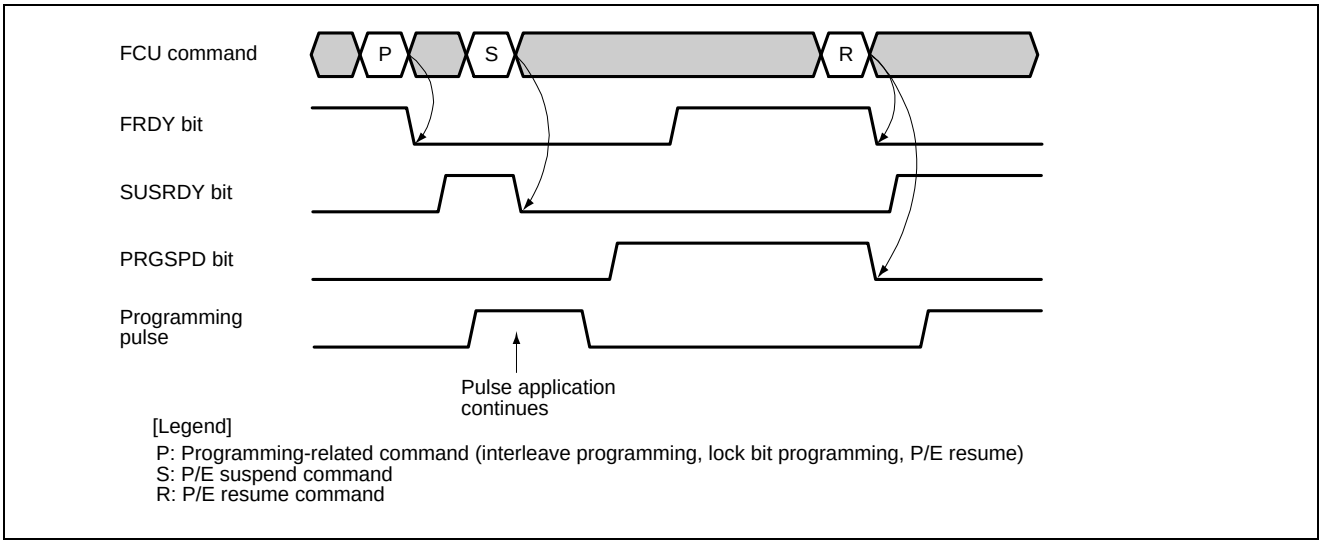



**Figure 26.21 Suspension during Programming**



Downloaded from Arrow.com.

### 26.7.2 Suspension during Erasure (Suspension Priority Mode)

Figure 26.22 shows the suspend operation of erasure when the erasure suspend mode is set to the suspension priority mode (ESUSPMD bit in FCPSR is 0).

When receiving an erasure-related command, the FCU clears the FSTATR0.FRDY bit to 0 to start erasure. If the FCU enters the state in which the P/E suspend command can be received after starting erasure, the FSTATR0.SUSRDY bit is set to 1. When a P/E suspend command is issued, the FCU receives the command and clears the SUSRDY bit to 0. When receiving a suspend command during erasure, the FCU starts the suspend processing and sets the ERSSPD bit in FSTATR0 to 1 even if it is applying an erasure pulse. When the suspend processing finishes, the FCU sets the FRDY bit to 1 to enter the erasure suspended state. If receiving a P/E resume command in the erasure suspended state, the FCU clears the FRDY and ERSSPD bits to 0 and resumes erasure.

The setting of the erasure suspend mode affects the control method of erasure pulses. In suspension priority mode, when receiving a P/E suspend command while erasure pulse A that has never been suspended in the past is being applied, the FCU suspends the application of erasure pulse A and enters the erasure suspended state. When receiving a P/E suspend command while reapplying erasure pulse A after erasure is resumed by a P/E resume command, the FCU continues applying erasure pulse A. After specified pulse application time, the FCU finishes erasure pulse application and enters the erasure suspended state. When the FCU receives a P/E resume command next and erasure pulse B starts to be newly applied, and then the FCU receives a P/E suspend command again, the application of erasure pulse B is suspended. In suspension priority mode, delay due to suspend can be minimized because the application of an erasure pulse is suspended one time per pulse and priority is given to the suspension processing.

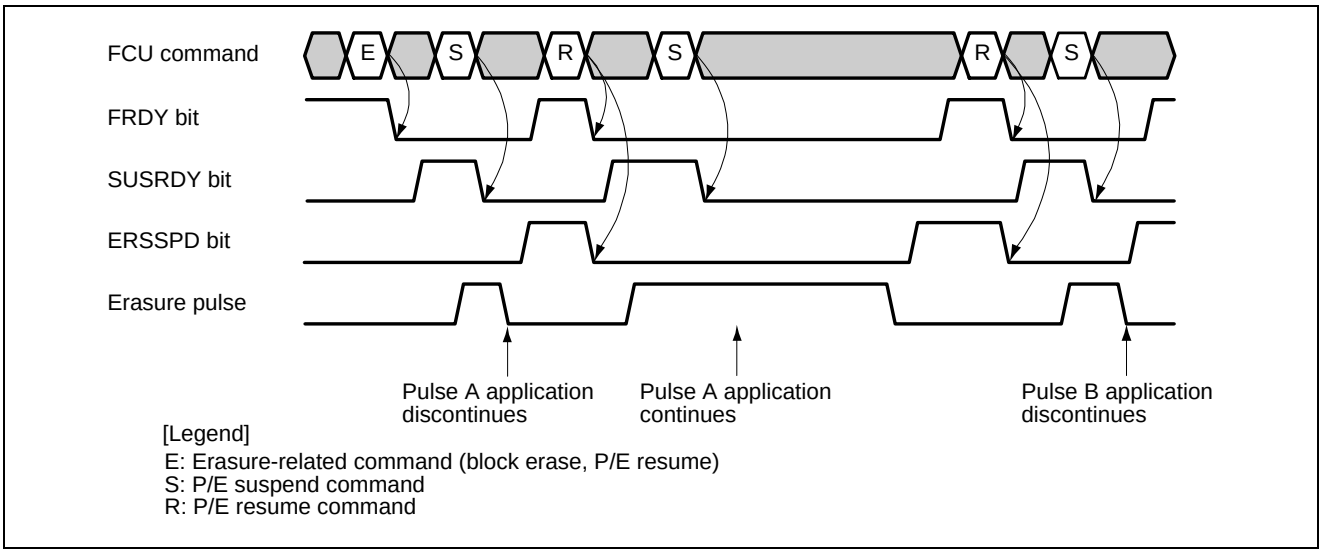



**Figure 26.22 Suspension during Erasure (Suspension Priority Mode)**



Downloaded from Arrow.com.

### 26.7.3 Suspension during Erasure (Erasure Priority Mode)

Figure 26.23 shows the suspend operation of erasure when the erasure suspend mode is set to the erasure priority mode (ESUSPMD bit in FCPSR is 1). The control method of erasure pulses in erasure priority mode is the same as that of programming pulses for the programming suspend processing.

If the FCU receives a P/E suspend command while an erasure pulse is being applied, the FCU definitely continues applying the pulse. In this mode, required time for the whole erasure processing can be reduced as compared with the suspension priority mode because the reapplication of erasure pulses does not occur when a P/E resume command issued.

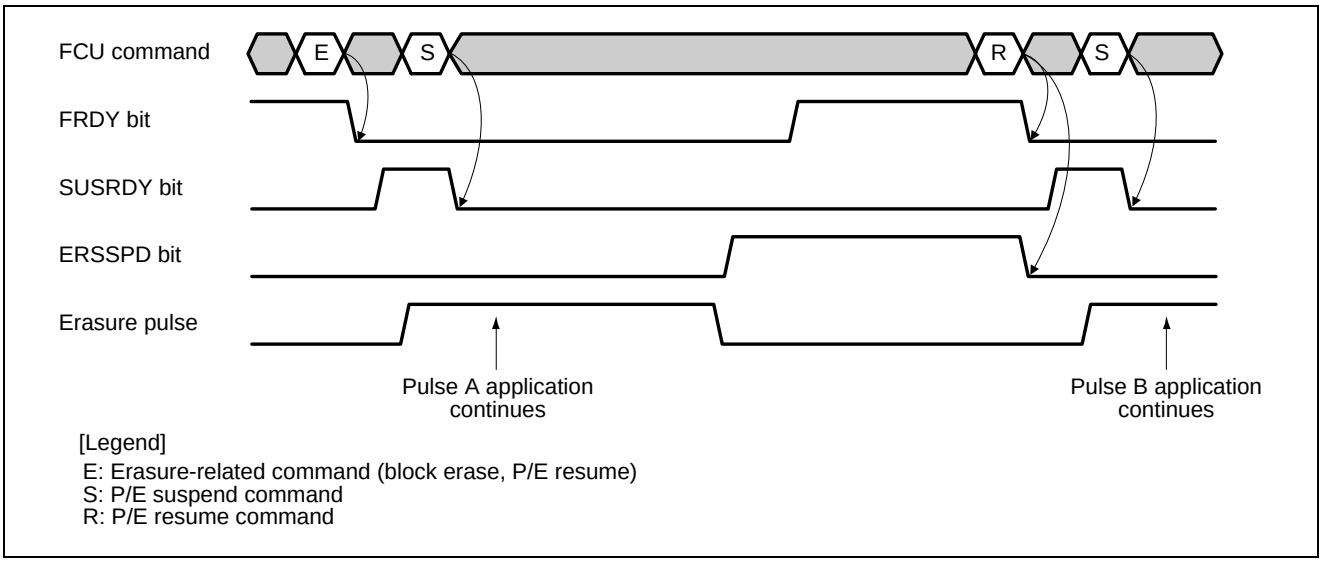



**Figure 26.23 Suspension during Erasure (Erasure Priority Mode)**



Downloaded from Arrow.com.

## 26.8 Protection

Protection against programming/erasure for the ROM includes software protection and error protection.

### 26.8.1 Software Protection

With the software protection, the ROM programming/erasure is prohibited by the settings of the control register or user mat lock bit. When the software protection is violated and a ROM programming/erasure-related command is issued, the FCU detects an error and enters the command-locked state.

#### (1) Protection through FWEPROR

If the FLWE[1:0] bits in FWEPROR are not set to 01b, programming cannot be performed in any of the modes.

#### (2) Protection through FENTRYR

When the FENTRY1* and FENTRY0 bits in FENTRYR are 0, ROM read mode is selected. Because the FCU command cannot be received in ROM read mode, ROM programming/erasure is prohibited. When an FCU command is issued in ROM read mode, the FCU detects an illegal command error and is placed in the command-locked state (see section 26.8.2, Error Protection).

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.

#### (3) Protection through Lock Bit

Each erasure block in the user mat includes a lock bit. When the FPROTCN bit in FPROTR is 0, erasure blocks whose lock bit is set to 0 are prohibited from being programmed/erased. To program or erase erasure blocks whose lock bit is set to 0, set the FPROTCN bit to 1. When the lock bit protection is violated and a ROM programming/erasure-related command is issued, the FCU detects a programming/erasure error and enters the command-locked state (see section 26.8.2, Error Protection).

### 26.8.2 Error Protection

With the error protection, FCU command issuance errors, prohibited access occurrences, and FCU malfunctions are detected, and an FCU command is prohibited from being received (command-locked state). When the FCU enters the command-locked state (FASTAT.CMDLK bit is 1), one or several of the status bits (FSTATR0.ILGLERR, ERSERR, and PRGERR bits, FSTATR1.FCUERR bit, and FASTST.ROMAE bit) are set to 1 and programming and erasure of the ROM are prohibited. To clear the command-locked state, a status register clear command must be issued with FASTAT set to 10h.

While the CMDLKIE bit in FAEINT is set to 1, if the FCU is placed in the command-locked state (CMDLK bit in FASTAT is set to 1), a flash interface error (FIFERR) interrupt occurs. While the ROMAEIE bit in FAEINT is set to 1, if the ROMAE bit in FASTAT is set to 1, an FIFERR interrupt occurs.

Table 26.10 lists the relationship between the contents of the ROM-related error protection and status bit values (ILGLERR, ERSERR, PRGERR bits in FSTATR0, FCUERR bit in FSTATR1, ROMAE bit in FASTAT) at error detection. If a command other than the suspend command is issued during programming/erasure and the FCU enters the command-locked state, it continues the programming/erasure. In this state, it is impossible to issue a P/E suspend command and suspend programming/erasure. When a command is issued in the command-locked state, the ILGLERR bit is set to 1.



Downloaded from Arrow.com.

**Table 26.10 Error Protection Types (Types Dedicated to ROM and Types Common to ROM and Data Flash)**

| Type | Description | ILGLERR | ERSERR | PRGERR | FCUERR | ROMAE | CMDLK |
|---|---|---|---|---|---|---|---|
| FENTRYR setting error | More than one bit is set to 1 among the FENTRYD, FENTRY1*, and FENTRY0 bits in FENTRYR | 1 | 0 | 0 | 0 | 0 | 1 |
| | The FENTRYR setting at suspension disagrees with that at resume | 1 | 0 | 0 | 0 | 0 | 1 |
| Illegal command error | Undefined code is specified in the first cycle of an FCU command | 1 | 0 | 0 | 0 | 0 | 1 |
| | Other than D0h is specified in the last cycle of a multi-cycle FCU command | 1 | 0 | 0 | 0 | 0 | 1 |
| | A command other than the suspend command is issued during programming/erasure | 1 | 0 | 0 | 0 | 0 | 1 |
| | A suspend command is issued during processing other than programming/erasure | 1 | 0 | 0 | 0 | 0 | 1 |
| | A suspend command is issued in the suspended state | 1 | 0 | 0 | 0 | 0 | 1 |
| | A resume command is issued in other than the suspended state | 1 | 0 | 0 | 0 | 0 | 1 |
| | A programming/erasure-related (programming/lock bit programming/block erase) command is issued in the programming suspended state | 1 | 0 | 0 | 0 | 0 | 1 |
| | A block erase command is issued in the erasure suspended state | 1 | 0 | 0 | 0 | 0 | 1 |
| | A programming or lock bit programming command is issued to an erasure suspend target area in the erasure suspended state | 1 | 0 | 0 | 0 | 0 | 1 |
| | Other than 80h is specified in the second cycle of the programming command | 1 | 0 | 0 | 0 | 0 | 1 |
| | A command is issued in the command-locked state | 1 | 0/1 | 0/1 | 0/1 | 0/1 | 1 |
| Erasure error | An error occurs during erasure | 0 | 1 | 0 | 0 | 0 | 1 |
| | When the FPROTCN bit in FPROTR is 0, a block erase command is issued to a erasure block whose lock bit is set to 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Programming error | An error occurs during programming | 0 | 0 | 1 | 0 | 0 | 1 |
| | When the FPROTCN bit in FPROTR is 0, a programming or lock bit programming command is issued to a erasure block whose lock bit is set to 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| FCU error | An error occurs during FCU internal processing | 0 | 0 | 0 | 1 | 0 | 1 |
| ROM access violation | When the FENTRY1* bit in FENTRYR is 1 and the FCU is in P/E normal mode, a read command is issued for addresses 00E0 0000h to 00EF FFFFh | 1 | 0 | 0 | 0 | 1 | 1 |
| | When the FENTRY0 bit in FENTRYR is 1 and the FCU is in P/E normal mode, a read command is issued for addresses 00F0 0000h to 00FF FFFFh | 1 | 0 | 0 | 0 | 1 | 1 |
| | When the FENTRY1* bit in FENTRYR is 0, a command is issued for addresses 00E0 0000h to 00EF FFFFh | 1 | 0 | 0 | 0 | 1 | 1 |
| | When the FENTRY0 bit in FENTRYR is 0, a command is issued for addresses 00F0 0000h to 00FF FFFFh | 1 | 0 | 0 | 0 | 1 | 1 |
| | A read command is issued for addresses FFE0 0000h to FFFF FFFFh when FENTRYR is set to other than 0000h | 1 | 0 | 0 | 0 | 1 | 1 |

Note: * Cannot be used in a product whose ROM size is equal to or smaller than 1 Mbyte.



Downloaded from Arrow.com.

## 26.9 User Boot Mode

When the reset clearing is executed with the MD1 and MD0 pins set to user boot mode, the FCU enters user boot mode. The reset vector at this time is address FF7F FFFCh in the user boot mat. For other vector tables, see the normal vector table (see section 10, Interrupt Control Unit (ICU)).

In user boot mode, software for programming with an arbitrary interface can be prepared, and the user mat or data mat can be programmed/erased by issuing an FCU command. Programming for the user boot mat should be performed only in boot mode.

## 26.10 Boot Mode

### 26.10.1 System Configuration

In boot mode, the host sends control commands and data for programming, and the user mat, user boot mat, and data mat are programmed or erased accordingly. An on-chip SCI handles transfer between the host and RX610 in asynchronous mode. Tools for the transmission of control commands and the data for programming must be prepared in the host.

When the RX610 is activated in boot mode, the program on the mat that holds the embedded program is executed. This program automatically adjusts the bit rate of the SCI and controls programming/erasure by receiving control commands from the host.

Figure 26.24 shows the system configuration for operations in boot mode.

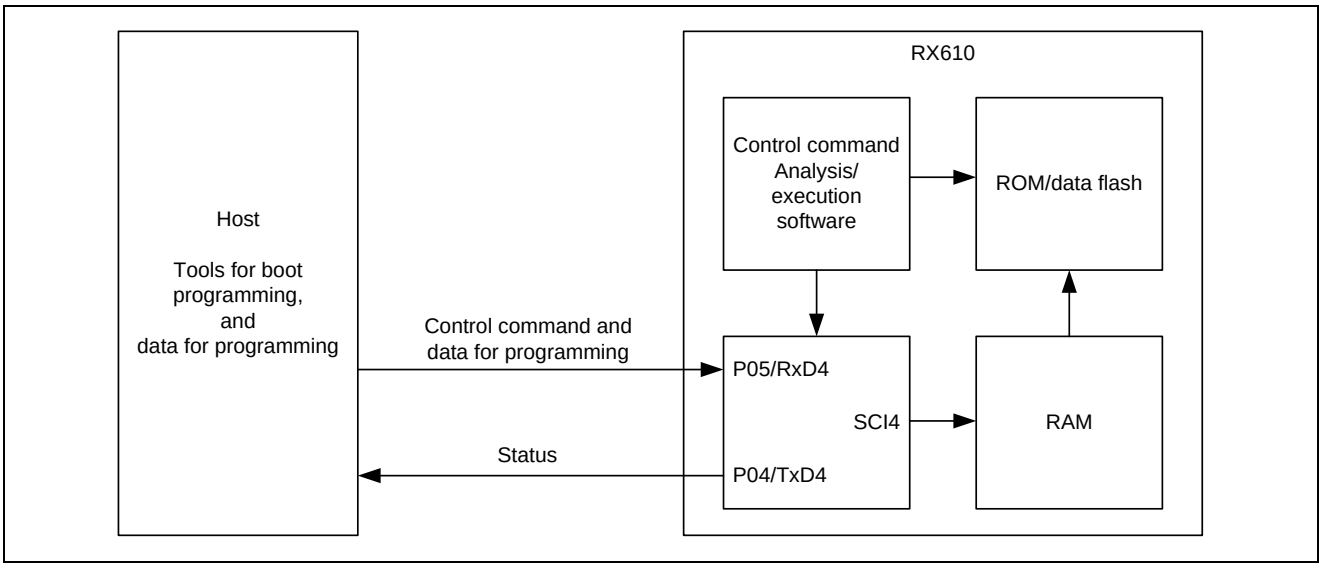



**Figure 26.24 System Configuration for Operations in Boot Mode**



Downloaded from Arrow.com.

## 26.10.2 ID Code Protection

This function is used to prohibit reading/programming/erasure from the host.

Using the control code and ID code written in the ROM, ID code protection is enabled or disabled and ID code protection is judged. When ID code protection is enabled, the code sent from the host is compared with the control code and ID code in the ROM to determine whether they match, and reading/programming/erasure will be enabled only when the two match.

The control code and ID code in the ROM consists of four 32-bit words. Figure 26.25 shows the configuration of the control code and ID code. The ID code should be set in 32-bit units.

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| FFFF FFA0h | Control code | ID code 1 | ID code 2 | ID code 3 |
| FFFF FFA4h | ID code 4 | ID code 5 | ID code 6 | ID code 7 |
| FFFF FFA8h | ID code 8 | ID code 9 | ID code 10 | ID code 11 |
| FFFF FFACh | ID code 12 | ID code 13 | ID code 14 | ID code 15 |

**Figure 26.25 Configuration of Control Code and ID Code in ROM**

### (1) Control Code

The control code determines whether ID code protection is or is not enabled and the method of authentication to use with the host. Table 26.11 shows how the control code determines the method of authentication



Downloaded from Arrow.com.

**Table 26.11 Specifications for ID Code Protection**

| Control Code | ID Code | State of Protection | Operations at the Time of SCI Connection |
|---|---|---|---|
| 45h | As desired | Protection enabled (authentication method 1) | Matching ID code: ID code protection is lifted, and this is followed by a transition to the state of waiting for a host command.<br>Non-matching ID code: Up to two more transitions to the ID code protection waiting state; complete erasure if a non-matching ID code is received for a third time. |
| 52h | Sequences other than 50h, 72h, 6Fh, 74h, 65h, 63h, 74h, FFh, ..., FFh | Protection enabled (authentication method 2) | Matching ID code: ID code protection is lifted, and this is followed by a transition to the state of waiting for a host command.<br>Non-matching ID code: Further transition to the ID code protection waiting state |
| | 50h, 72h, 6Fh, 74h, 65h, 63h, 74h, FFh, ..., FFh | Protection enabled (authentication method 3) | Always judged to be a non-matching ID code. |
| Other than the above | — | Protection disabled | Erasure of all blocks |

### (2) ID Code

The ID code can be set to any desired value. However, if the control code is 52h and the ID code is 50h, 72h, 6Fh, 74h, 65h, 63h, 74h, FFh, ..., FFh (from the ID code 1 field), there is no determination of matching and the ID code is always considered to be non-matching. Accordingly, reading, programming, and erasure from the host are prohibited.

### (3) Example of Assembler Directives for Setting an ID Code

The following assembler directives set up a control code of 45h and an ID code of 01h, 02h, 03h, 04h, 05h, 06h, 07h, 08h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh (from the ID code 1 field).

```
.SECTION    ID_CODE,CODE
.ORG        0FFFFFFA0h
.LWORD      45010203h
.LWORD      04050607h
.LWORD      08090A0Bh
.LWORD      0C0D0E0Fh
```



Downloaded from Arrow.com.

## 26.10.3 State Transitions in Boot Mode

Figure 26.26 is a diagram of the state transitions in boot mode.

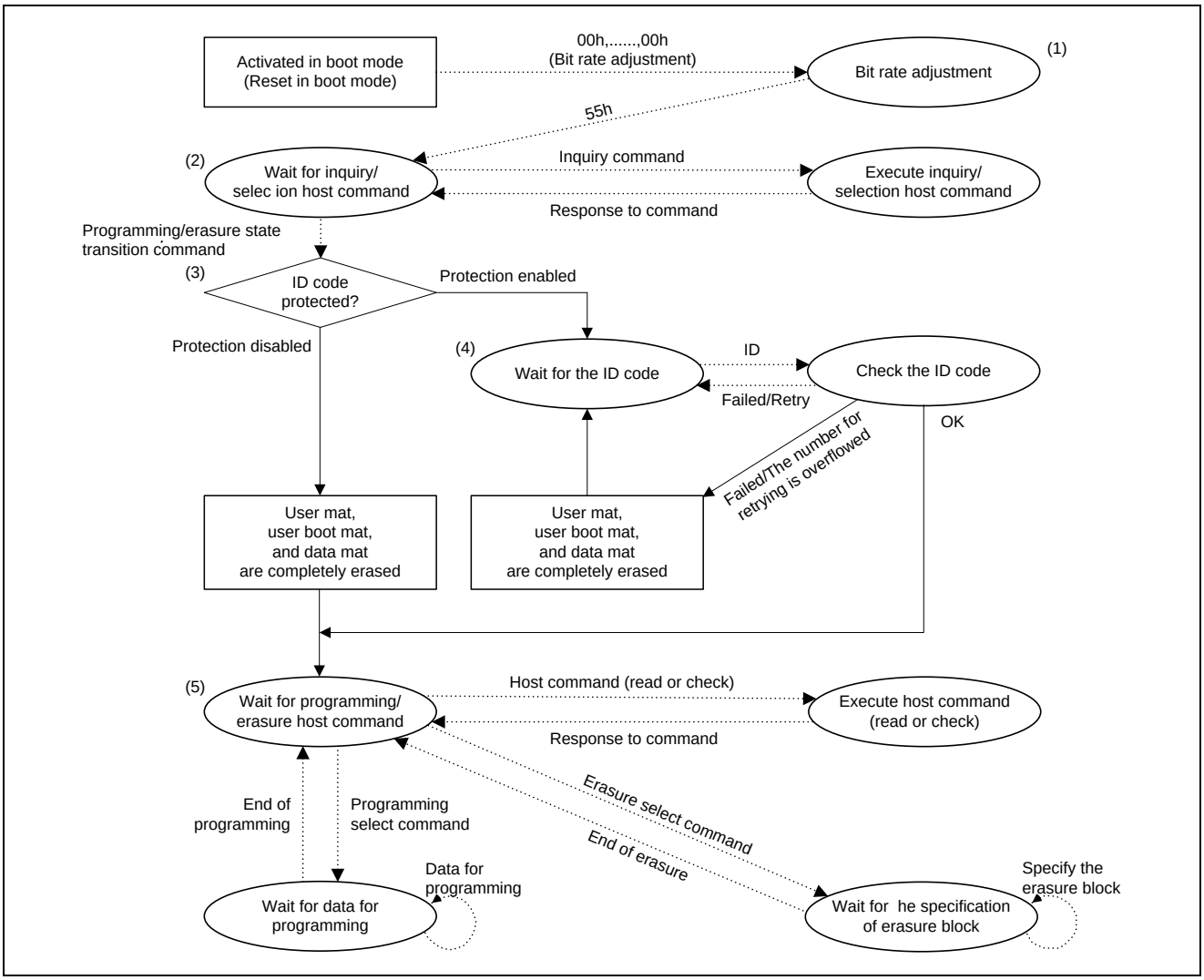



**Figure 26.26 State Transitions in Boot Mode**



Downloaded from Arrow.com.

### (1) Matching the Bit Rates

When the RX610 is activated in boot mode, the bit rate of the SCI is automatically adjusted to match that of the host. On completion of this adjustment, the RX610 transmits the value 00h to the host. On subsequent correct reception of the value 55h sent from the host, the RX610 enters the state of waiting for a host command for inquiry or selection. For details on matching of the bit rates, see section 26.10.4, Automatic Adjustment of the Bit Rate.

### (2) Waiting for a Host Command for Inquiry or Selection

This state is for inquiries on mat size, mat configuration, the addresses where mats start, the state of support etc., and for selection of the device, clock mode, and bit rate. The RX610 receives a programming/erasure state transition command issued by the host and then enters the state to determine whether ID code protection is enabled or disabled. For the inquiry/selection host commands, see section 26.10.5, Inquiry/Selection Host Command Wait State.

### (3) Judging ID Code Protection

This state is for determining whether ID code protection is enabled or disabled. The control code and ID code written in the ROM are used to determine whether ID code protection is enabled or disabled. When enabled, the state of waiting for the ID code is entered. When disabled, the user mat, user boot mat, and data mat are all completely erased, and the state of waiting for programming and erasure commands from the host is entered. For details on the control code and ID code, see section 26.10.2, ID Code Protection.

### (4) Waiting for an ID Code

This state is for waiting for the control code and ID code to be sent from the host. The control code and ID code sent by the host are compared with the code stored in the ROM, and the state of waiting for programming and erasure commands from the host is entered if the two match. If they do not match, the next transition is back to the state of waiting for an ID code. However, if the ID codes fail to match three times in a row and also the state of protection is authentication method 1, the ROM is completely erased, and the state of waiting for an ID code is entered again. A reset is required to release the system from this state due to non-matching ID codes. For details on the control code and ID code, see section 26.10.2, ID Code Protection.

### (5) Waiting for a Host Command for Programming or Erasure

In this state, programming and erasure proceed in accordance with commands from the host. In response to the reception of a command, the RX610 enters the state of waiting for the data to use in programming, waiting for specification of the erasure block to be erased, or executing the processing of commands for reading, fetching and so on.

When the RX610 receives a programming selection command, it enters the state of waiting for the data to use in programming. After the host has issued the programming selection command, the process continues with the address where programming is to start and then the data for programming. Setting of FFFF FFFFh as the address where programming is to start indicates the completion of programming, and the next transition is from the state of waiting for the data to use in programming to the state of waiting for programming and erasure commands.

When the RX610 receives a programming selection command, it enters the state of waiting for specification of the erasure block to be erased. After the host has issued the programming selection command, the process continues with the number of the erasure block to be erased. Setting of FFh as the number of the erasure block indicates the completion of erasure, and the next transition is from the state of waiting for specification of the erasure block to the state of waiting for programming and erasure commands. Since the user mat, user boot mat, and data mat are all completely erased during the interval between booting up in boot mode and transition to the state of waiting for programming and erasure



Downloaded from Arrow.com.

commands, explicit execution of erasure is not necessary unless newly programmed data are to be erased without a further reset.

Other than the programming and erasure commands, commands from the host for execution in this state include those for sum checking of the user mat and user boot mat, blank checking (to confirm erasure), reading from memory, and acquiring state information.

### 26.10.4 Automatic Adjustment of the Bit Rate

When the RX610 is booted up in boot mode, asynchronous transfer by the SCI is used to measure the periods at low level of consecutive bytes with value 00h that are sent from the host. While the period at low level is being measured, set the host's SCI parameters to eight-bit data, one stop bit, no parity, and a transfer rate of 9,600 bps or 19,200 bps. The RX610 calculates the host's SCI bit rate from the measured periods at low level, adjusts its own bit rate accordingly, and then sends a 00h byte to the host. If reception of the value 00h by the host is normal, the host responds by sending the value 55h to the RX610. If normal reception of 00h by the host is not possible, the RX610 is re-booted in boot mode, and then repeats the process of automatically adjusting the bit rate. If reception of the value 55h by the RX610 is normal, it responds by sending E6h to the host, and if normal reception of 55h by the RX610 is not possible, it responds by sending FFh to the host.

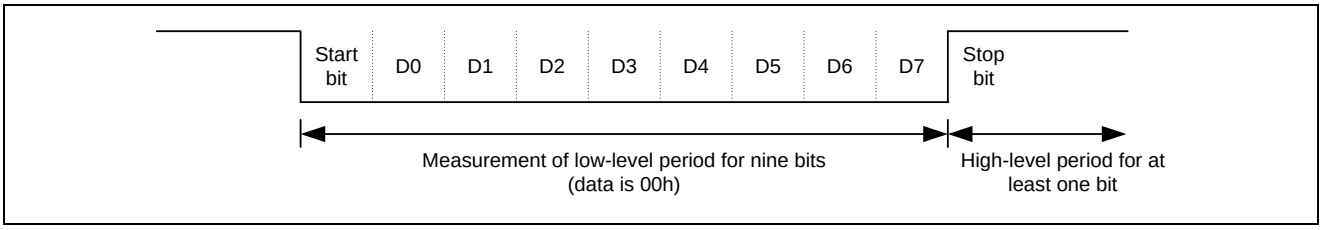



**Figure 26.27 Transfer Format Used by the SCI in Automatic Adjustment of the Bit Rate**

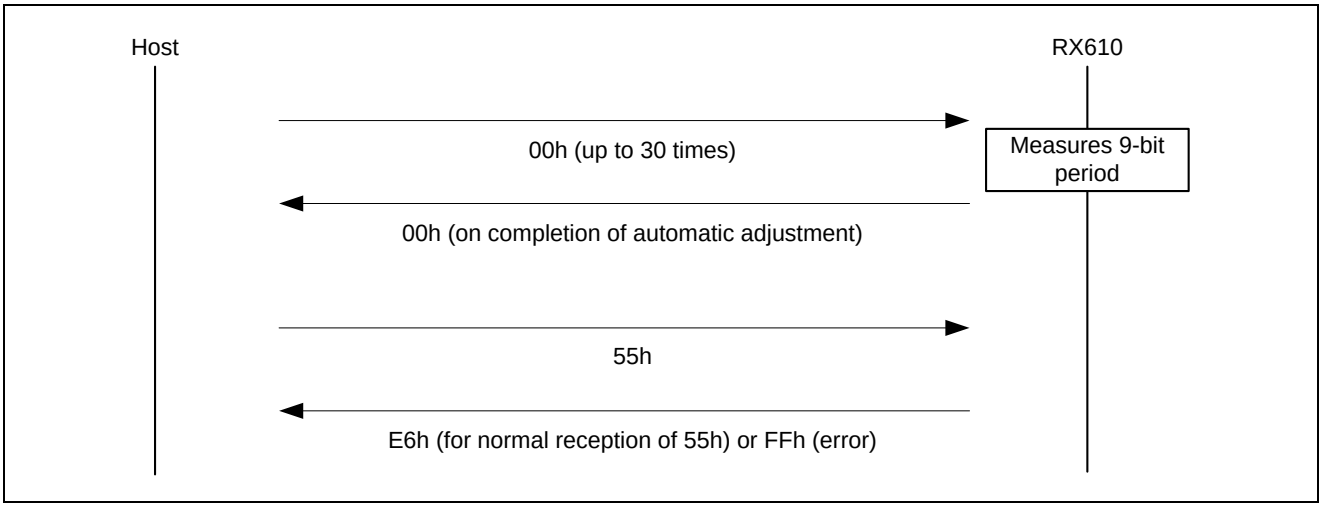



**Figure 26.28 Sequence of Transfer between the Host and RX610**

Since the bit rate of the RX610's SCI module depends on the frequency of the peripheral clock, adjustment to match the bit rate of the host will not be possible under some conditions. Accordingly, ensure that SCI transfer is under the conditions given in table 26.12.



Downloaded from Arrow.com.

**Table 26.12 Conditions for Automatic Bit-Rate Adjustment to be Possible**

| Bit Rate of the SCI in the Host | Range of Frequency for the EXTAL Signal |
|---|---|
| 9,600 bps | 8 to 14 MHz |
| 19,200 bps | 8 to 14 MHz |

## 26.10.5 Inquiry/Selection Host Command Wait State

Table 26.13 shows the host commands available in the inquiry/selection host command wait state. The embedded program status inquiry command can also be used in the programming/erasure host command wait state. The other commands can only be used in the inquiry/selection host command wait state.

**Table 26.13 Inquiry/Selection Host Commands**

| Host Command Name | Function |
|---|---|
| Supported device inquiry | Inquires regarding the device codes and the product codes for the embedded programs |
| Device selection | Selects a device code |
| Clock mode inquiry | Inquires regarding the clock mode |
| Clock mode selection | Notifies the selected clock mode |
| Multiplication ratio inquiry | Inquires regarding the number of clock types, the number of multiplication/division ratios, and the multiplication /division ratios |
| Operating frequency inquiry | Inquires regarding the number of clock types and the maximum and minimum operating frequencies |
| User boot mat information inquiry | Inquires regarding the number of user boot mats and the start and end addresses |
| User mat information inquiry | Inquires regarding the number of user mats and the start and end addresses |
| Erasure block information inquiry | Inquires regarding the number of blocks and the start and end addresses |
| Programming size inquiry | Inquires regarding the size of programming data |
| New bit rate selection | Modifies the bit rate of SCI communications between the host and RX610 |
| Programming/erasure state transition | Enters the state for determining ID code protection |
| Embedded program status inquiry | Inquires regarding the processing state |

If the host has sent an undefined command, the RX610 returns a response indicating a command error in the format shown below. The command field holds the first byte of the undefined command sent from the host.

| | | |
|---|---|---|
| Error response | 80h | Command |

In the inquiry/selection host command wait state, send selection commands from the host in the order of device selection, clock mode selection, and new bit rate selection to set up the RX610 according to the responses to inquiry commands. Note that the supported device inquiry and clock mode inquiry commands are the only inquiry commands that can be sent before the clock mode selection command; other inquiry commands must not be issued before the clock mode selection command. If commands are issued in an incorrect order, the RX610 returns a response indicating a command error. Figure 26.29 shows an example of the procedure to use inquiry/selection host commands.



Downloaded from Arrow.com.

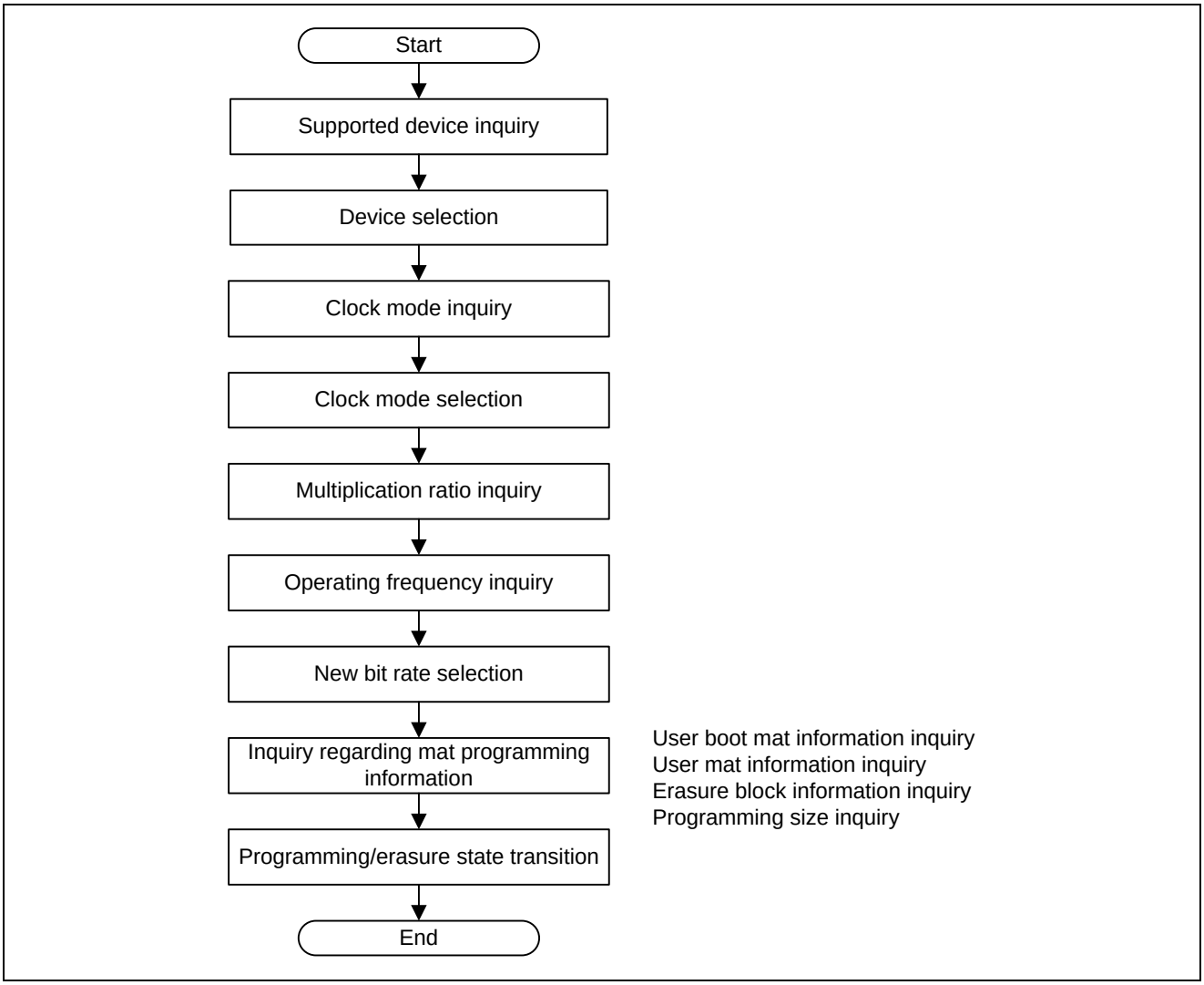



**Figure 26.29 Example of Procedure to Use Inquiry/Selection Host Commands for User Mat/User Boot Mat**

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to the RX610 and the "response" indicates a response sent from the RX610 to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by the RX610 becomes 00h.



Downloaded from Arrow.com.

### (1) Supported Device Inquiry

In response to a supported device inquiry command sent from the host, the RX610 returns the information concerning the devices supported by the embedded program for boot mode. If the supported device inquiry command comes after the host has selected a device, the RX610 only returns the information concerning the selected device.

Command

| 20h |
|---|

Response

| 30h | Size | Device count |
|---|---|---|
| Character count | Device code | Product code |
| Character count | Device code | Product code |
| : | : | : |
| Character count | Device code | Product code |
| SUM | | |

[Legend]

Size (1 byte): Total number of bytes in the device count, character count, device code, and product code fields

Device count (1 byte): Number of device types supported by the embedded program for boot mode

Character count (1 byte): Number of characters included in the device code and product code fields

Device code (4 bytes): ASCII code for the product name of the chip

Product code (n bytes): ASCII code for the supported device

SUM (1 byte): Checksum (in response)



Downloaded from Arrow.com.

### (2) Device Selection

In response to a device selection command sent from the host, the RX610 checks if the selected device is supported. When the selected device is supported, the RX610 specifies this device as the device for use and returns a response (06h). If the selected device is not supported or the sent command is illegal, the RX610 returns an error response (90h).

Even when 01h has been returned as the number of supported devices in response to a supported device inquiry command, issue a device selection command to specify the device code that has been returned as the result of the inquiry.

| | | | | |
|---|---|---|---|---|
| Command | 10h | Size | Device code | SUM |
| Response | 06h | | | |
| Error response | 90h | Error | | |

[Legend]

Size (1 byte): Number of characters in the device code field (fixed at four)

Device code (4 bytes): ASCII code for the product name of the chip (one of the device codes returned in response to the supported device inquiry command)

SUM (1 byte): Checksum

Error (1 byte): Error code
11h: Checksum error (illegal command)
21h: Incorrect device code error

### (3) Clock Mode Inquiry

In response to a clock mode inquiry command sent from the host, the RX610 returns the supported clock modes. If the clock mode inquiry command comes after the host has selected a clock mode, the RX610 only returns the information concerning the selected clock mode.

| | | | | |
|---|---|---|---|---|
| Command | 21h | | | |
| Response | 31h | Size | | |
| | Mode | Mode | . . . | Mode |
| | SUM | | | |

[Legend]

Size (1 byte): Total number of bytes in the mode count and mode fields

Mode (1 byte): Supported clock mode (for example, 01h indicates clock mode 1)

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (4) Clock Mode Selection

In response to a clock mode selection command sent from the host, the RX610 checks if the selected clock mode is supported. When the selected mode is supported, the RX610 specifies this clock mode for use and returns a response (06h). If the selected mode is not supported or the sent command is illegal, the RX610 returns an error response (91h).

Be sure to issue a clock mode selection command only after issuing a device selection command. Even when 00h or 01h has been returned as the number of supported clock modes in response to a clock mode inquiry command, issue a clock mode selection command to specify the clock mode that has been returned as the result of the inquiry.

| | | | | |
|---|---|---|---|---|
| Command | 11h | Size | Mode | SUM |
| Response | 06h | | | |
| Error response | 91h | Error | | |

[Legend]

Size (1 byte): Number of characters in the mode field (fixed at 1)

Mode (1 byte): Clock mode (same mode as the response to the clock mode inquiry command)

SUM (1 byte): Checksum

Error (1 byte): Error code
11h: Checksum error (illegal command)
21h: Incorrect clock mode error



Downloaded from Arrow.com.

### (5) Multiplication Ratio Inquiry

In response to a multiplication ratio inquiry command sent from the host, the RX610 returns the clock types, the number of multiplication/division ratios, and the multiplication division ratios supported.

Command

| 22h |
|---|

Response

| 32h | Size | Clock type count | | |
|---|---|---|---|---|
| Multiplication ratio count | Multiplication ratio | Multiplication ratio | . . . | Multiplication ratio |
| Multiplication ratio count | Multiplication ratio | Multiplication ratio | . . . | Multiplication ratio |
| : | : | : | . . . | : |
| Multiplication ratio count | Multiplication ratio | Multiplication ratio | . . . | Multiplication ratio |
| SUM | | | | |

[Legend]

Size (1 byte): Total number of bytes in the clock type count, multiplication ratio type, and multiplication ratio fields

Clock type count (1 byte): Number of clock types (for example, 02h indicates two clock types; that is, a system clock and a peripheral clock)

Multiplication ratio count (1 byte): Number of supported multiplication/division ratios (for example, 04h indicates that four multiplication ratios are supported for the system clock (x1, x2, x4, and x8))

Multiplication ratio (1 byte): A positive value indicates a multiplication ratio (for example, 04h = 4 = multiplication by 4)
A negative value indicates a division ratio (for example, FEh = -2 = division by 2)

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (6) Operating Clock Frequency Inquiry

In response to an operating clock frequency inquiry command sent from the host, the RX610 returns the minimum and maximum frequencies for each clock.

| | | | |
|---|---|---|---|
| Command | 23h | | |
| Response | 33h | Size | Clock type count |
| | Minimum frequency | | Maximum frequency |
| | Minimum frequency | | Maximum frequency |
| | : | | : |
| | Minimum frequency | | Maximum frequency |
| | SUM | | |

[Legend]

Size (1 byte): Total number of bytes in the clock type count, minimum frequency, and maximum frequency fields

Clock type count (1 byte): Number of clock types (for example, 02h indicates two clock types; that is, a system clock and a peripheral clock)

Minimum frequency (2 bytes): Minimum value of the operating frequency (for example, 07D0h indicates 20.00 MHz).
This value should be calculated by multiplying the frequency value (MHz) to two decimal places by 100.

Maximum frequency (2 bytes): Maximum value of the operating frequency represented in the same format as the minimum frequency

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (7) User Boot Mat Information Inquiry

In response to a user boot mat information inquiry command sent from the host, the RX610 returns the number of user boot mat areas and their addresses.

| | | | |
|---|---|---|---|
| Command | 24h | | |
| Response | 34h | Size | Area count |
| | Area start address | | |
| | Area end address | | |
| | Area start address | | |
| | Area end address | | |
| | : | | |
| | Area start address | | |
| | Area end address | | |
| | SUM | | |

[Legend]

Size (1 byte): Total number of bytes in the area count, area start address, and area end address fields

Area count (1 byte): Number of user boot mat areas (consecutive areas are counted as one area)

Area start address (4 bytes): Start address of a user boot mat area

Area end address (4 bytes): End address of a user boot mat area

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (8) User Mat Information Inquiry

In response to a user mat information inquiry command sent from the host, the RX610 returns the number of user mat areas and their addresses.

| | | | |
|---|---|---|---|
| Command | 25h | | |
| Response | 35h | Size | Area count |
| | Area start address | | |
| | Area end address | | |
| | Area start address | | |
| | Area end address | | |
| | : | | |
| | Area start address | | |
| | Area end address | | |
| | SUM | | |

[Legend]

Size (1 byte): Total number of bytes in the area count, area start address, and area end address fields

Area count (1 byte): Number of user mat areas (consecutive areas are counted as one area)

Area start address (4 bytes): Start address of a user mat area

Area end address (4 bytes): End address of a user mat area

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (9) Erasure Block Information Inquiry

In response to an erasure block information inquiry command sent from the host, the RX610 returns the number of erasure blocks in the user mat and their addresses.

| | | | |
|---|---|---|---|
| Command | 26h | | |
| Response | 36h | Size | Block count |
| | Block start address | | |
| | Block end address | | |
| | Block start address | | |
| | Block end address | | |
| | : | | |
| | Block start address | | |
| | Block end address | | |
| | SUM | | |

[Legend]

Size (1 byte): Total number of bytes in the block count, block start address, and block end address fields

Block count (1 byte): Number of erasure blocks in the user mat

Block start address (4 bytes): Start address of an erasure block

Block end address (4 bytes): End address of an erasure block

SUM (1 byte): Checksum

### (10) Programming Size Inquiry

In response to a programming size inquiry command sent from the host, the RX610 returns the programming size.

| | | | | |
|---|---|---|---|---|
| Command | 27h | | | |
| Response | 37h | Size | Programming size | SUM |

[Legend]

Size (1 byte): Number of characters included in the programming size field (fixed at two)

Programming size (2 bytes): Programming unit (bytes)

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (11) New Bit Rate Selection

In response to a new bit rate selection command sent from the host, the RX610 checks if the on-chip SCI can be set to the selected new bit rate. When the SCI can be set to the new bit rate, the RX610 returns a response (06h) and sets the SCI to the new bit rate. If the SCI cannot be set to the new bit rate or the sent command is illegal, the RX610 returns an error response (BFh). Upon reception of response 06h, the host waits for a one-bit period in the previous bit rate with which the new bit rate selection command has been sent, and then sets the host's bit rate to the new one. After that, the host sends confirmation data (06h) in the new bit rate, and the RX610 returns a response (06h) to the confirmation data.

Be sure to issue a new bit rate selection command only after a clock mode selection command.

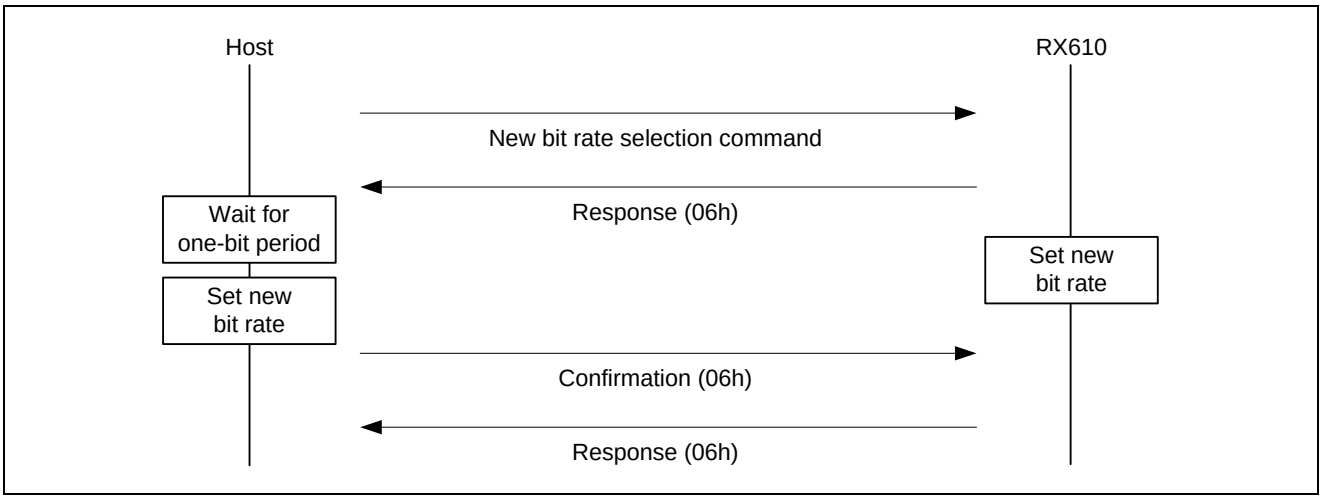



**Figure 26.30 New Bit Rate Selection Sequence**

| | | | | |
|---|---|---|---|---|
| Command | 3Fh | Size | Bit rate | Input frequency |
| | Clock type count | Multiplication ratio 1 | Multiplication ratio 2 | |
| | SUM | | | |
| Response | 06h | | | |
| Error response | BFh | Error | | |
| Confirmation | 06h | | | |
| Response | 06h | | | |



Downloaded from Arrow.com.

[Legend]

| | |
|---|---|
| Size (1 byte): | Total number of bytes in the bit rate, input frequency, clock type count, and multiplication ratio fields |
| Bit rate (2 bytes): | New bit rate (for example, 00C0h indicates 19200 bps)<br>1/100 of the new bit rate value should be specified. |
| Input frequency (2 bytes): | Clock frequency input to the RX610 (for example, 04E2h indicates 12.50 MHz)<br>This value should be calculated by multiplying the input frequency value to two decimal places by 100. |
| Clock type count (1 byte): | Number of clock types (for example, 02h indicates two clock types; that is, a system clock and a peripheral clock) |
| Multiplication ratio 1 (1 byte): | Multiplication/division ratio of the input frequency to obtain the system clock (ICLK)<br>A positive value indicates a multiplication ratio (for example, 04h = 4 = multiplication by 4)<br>A negative value indicates a division ratio (for example, FEh = -2 = division by 2) |
| Multiplication ratio 2 (1 byte): | Multiplication/division ratio of the input frequency to obtain the peripheral clock (PCLK)<br>This value is represented in the same format as multiplication ratio 1 |
| SUM (1 byte): | Checksum |
| Error: | Error code<br>11h: Checksum error<br>24h: Bit rate selection error<br>25h: Input frequency error<br>26h: Multiplication ratio error<br>27h: Operating frequency error |

- Bit rate selection error

  A bit rate selection error occurs when the bit rate selected through a new bit rate selection command cannot be set for the SCI of the RX610 within an error of 4%. The bit rate error can be obtained by the following equation from the bit rate (B) selected through a new bit rate selection command, the input frequency ($f_{EX}$), multiplication ratio 2 ($M_{P\phi}$), the bit rate register (BRR) setting (N) in the SCI, and the CKS[1:0] bit value (n) in the serial mode register (SMR).

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

- Input frequency error

  An input frequency error occurs when the input frequency specified through a new bit rate selection command is outside the range from the minimum to maximum input frequencies for the clock mode selected through a clock mode selection command.



Downloaded from Arrow.com.

- Multiplication ratio error

  A multiplication ratio error occurs when the multiplication ratio specified through a new bit rate selection command does not match the clock mode selected through a clock mode selection command. To check the selectable multiplication ratios, issue a multiplication ratio inquiry command.

- Operating frequency error

  An operating frequency error occurs when the RX610 cannot operate at the operating frequencies selected through a new bit rate selection command. The RX610 calculates the operating frequencies from the input frequency and multiplication ratios specified through a new bit rate selection command and checks if each calculated frequency is within the range from the minimum to maximum frequencies for the respective clock. To check the minimum and maximum operating frequencies for each clock, issue an operating clock frequency inquiry command.

### (12) Programming/Erasure State Transition

In response to a programming/erasure state transition command sent from the host, the RX610 determines whether ID code protection is enabled or disabled using the control code and ID code written in the ROM. When ID code protection is enabled, the RX610 returns a response (16h) and waits for the ID code. When ID code protection is disabled, the RX610 erases the entire area of each of the user mat, user boot mat, and data mat. After completing erasure, the RX610 returns a response (06h) and waits for a programming/erasure host command. If the RX610 has failed to complete erasure due to an error, it returns an error response (sends C0h and 51h in that order).

Do not issue a programming/erasure state transition command before device selection, clock mode selection, and new bit rate selection commands.

| | | |
|---|---|---|
| Command | 40h | |
| Response | ACK | |
| Error response | C0h | 51h |

[Legend]

ACK (1 byte): ACK code
06h: ID code protection is disabled
16h: ID code protection is enabled



Downloaded from Arrow.com.

### (13) Embedded Program Status Inquiry

In response to an embedded program status inquiry command sent from the host, the RX610 returns its current status. The embedded program status inquiry command can be issued in both the inquiry/selection host command wait state and programming/erasure host command wait state.

| | | | | |
|---|---|---|---|---|
| Command | 4Fh | | | |
| Response | 5Fh | Size | Status | Error |

[Legend]

Size (1 byte): Total number of bytes in the status and error fields (fixed at two)

Status (1 byte): Current status in the RX610 (see table 26.14)

Error (1 byte): Error status in the RX610 (see table 26.15)

**Table 26.14 Status Code**

| Code | Description |
|---|---|
| 11h | Waiting for device selection |
| 12h | Waiting for clock mode selection |
| 13h | Waiting for bit rate selection |
| 1Fh | Waiting for transition to programming/erasure host command wait state (bit rate has been selected) |
| 31h | Erasing the user mat and user boot mat |
| 3Fh | Waiting for a programming/erasure host command |
| 4Fh | Waiting for reception of programming data |
| 5Fh | Waiting for erasure block selection |

**Table 26.15 Error Code**

| Code | Description |
|---|---|
| 00h | No error |
| 11h | Checksum error |
| 21h | Incorrect device code error |
| 22h | Incorrect clock mode error |
| 24h | Bit rate selection error |
| 25h | Input frequency error |
| 26h | Multiplication ratio error |
| 27h | Operating frequency error |
| 29h | Block number error |
| 2Ah | Address error |
| 2Bh | Data size error |
| 51h | Erasure error |
| 52h | Incomplete erasure error |
| 53h | Programming error |
| 54h | Selection error |
| 80h | Command error |
| FFh | Bit rate adjustment verification error |



Downloaded from Arrow.com.

## 26.10.6 ID Code Wait State

Table 26.16 shows the host command available in the ID code wait state.

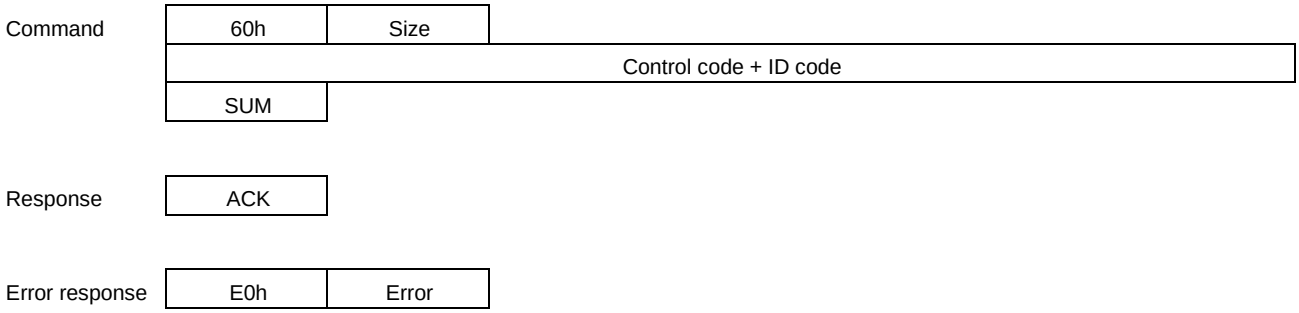
**Table 26.16 ID Code Check Host Command**

| Host Command Name | Function |
|---|---|
| ID code check | Performs the ID code check |

If the host has sent an undefined command, the RX610 returns a response indicating a command error. For the format of this response, see section 26.10.5, Inquiry/Selection Host Command Wait State.

### (1) ID Code Check

In response to an ID code check command sent from the host, the RX610 compares the code sent from the host with the control code and ID code in the ROM and returns the result.




[Legend]

Size (1 byte): Number of bytes in the ID code field (fixed at 16)

ID code (16 bytes): Control code (1 byte) + ID code (15 bytes)

SUM (1 byte): Checksum

ACK (1 byte): ACK code
26h: Returns the response for a programming/erasure state transition command

Error (1 byte): Error code
11h: Checksum error
61h: ID code mismatch
63h: ID code mismatch (erasure error)
An error has occurred during erasure triggered by an ID code mismatch.



Downloaded from Arrow.com.

### 26.10.7 Programming/Erasure Host Command Wait State

Table 26.17 shows the host commands available in the programming/erasure host command wait state.

**Table 26.17 Programming/Erasure Host Commands**

| Host Command Name | Function |
|---|---|
| User boot mat programming selection | Selects the program for user boot mat programming |
| User mat programming selection | Selects the program for user mat programming |
| 256-byte programming | Programs 256 bytes of data |
| Erasure selection | Selects the erasure program |
| Block erasure | Erases block data |
| Memory read | Reads data from memory |
| User boot mat checksum | Performs checksum verification for the user boot mat |
| User mat checksum | Performs checksum verification for the user mat |
| User boot mat blank check | Checks whether the user boot mat is blank |
| User mat blank check | Checks whether the user mat is blank |
| Read lock bit status | Reads from the lock bit |
| Lock bit program | Writes to the lock bit |
| Lock bit enable | Enables the lock bit protection |
| Lock bit disable | Disables the lock bit protection |
| Embedded program status inquiry | Inquires regarding the state of the RX610 |

If the host has sent an undefined command, the RX610 returns a response indicating a command error. For the format of this response, see section 26.10.5, Inquiry/Selection Host Command Wait State.

To program the ROM, issue a programming selection command (user boot mat programming selection or user mat programming selection command) and then a 256-byte programming command from the host. Upon reception of a programming selection command, the RX610 enters the programming data wait state (see section 26.10.3, State Transitions in Boot Mode). In response to a 256-byte programming command sent from the host in this state, the RX610 starts programming the ROM. When the host sends a 256-byte programming command specifying FFFF FFFFh as the programming start address, the RX610 detects it as the end of programming and enters the programming/erasure host command wait state.

To erase the ROM, issue an erasure selection command and then a block erasure command from the host. Upon reception of an erasure selection command, the RX610 enters the erasure block selection wait state (see section 26.10.3, State Transitions in Boot Mode). In response to a block erasure command sent from the host in this state, the RX610 erases the specified block in the ROM. When the host sends a block erasure command specifying FFh as the block number, the RX610 detects it as the end of erasure and enters the programming/erasure host command wait state.



Downloaded from Arrow.com.

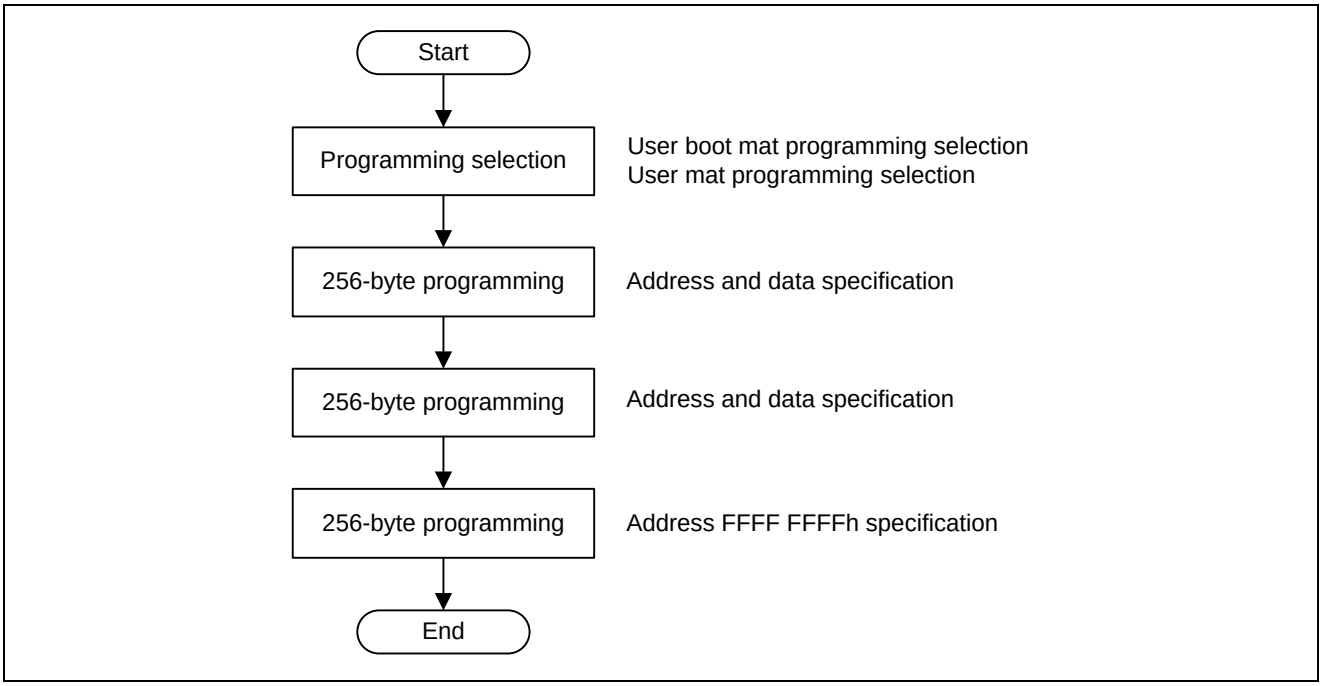



**Figure 26.31 Procedure for ROM Programming in Boot Mode**

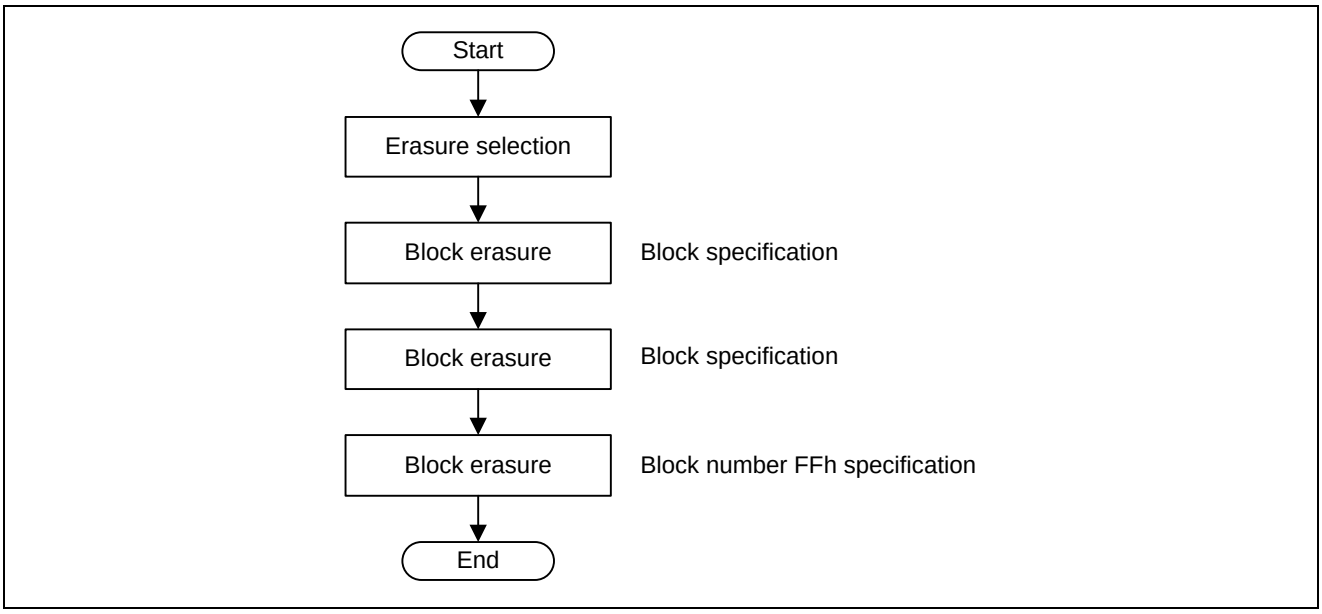



**Figure 26.32 Procedure for ROM Erasure in Boot Mode**

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to the RX610 and the "response" indicates a response sent from the RX610 to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by the RX610 becomes 00h.

### (1) User Boot Mat Programming Selection

In response to a user boot mat programming selection command sent from the host, the RX610 selects the program for user boot mat programming and waits for programming data.



Downloaded from Arrow.com.

| | |
|---|---|
| Command | 42h |
| Response | 06h |

### (2) User Mat Programming Selection

In response to a user mat programming selection command sent from the host, the RX610 selects the program for user mat programming and waits for programming data.

| | |
|---|---|
| Command | 43h |
| Response | 06h |

### (3) 256-Byte Programming

In response to a 256-byte programming command sent from the host, the RX610 programs the ROM. After completing ROM programming successfully, the RX610 returns a response (06h). If an error has occurred during ROM programming, the RX610 returns an error response (D0h).

| | | | | |
|---|---|---|---|---|
| Command | 50h | Programming address | | |
| | Data | Data | . . . | Data |
| | SUM | | | |
| Response | 06h | | | |
| Error response | D0h | Error | | |

[Legend]

Programming address (4 bytes): Target address of programming
To program the ROM, a 256-byte boundary address should be specified.
To terminate programming, FFFF FFFFh should be specified.

Data (256 bytes): Programming data
FFh should be specified for the bytes that do not need to be programmed.
When terminating programming, no data needs to be sent (only the programming address and SUM should be sent in that order).

SUM (1 byte): Checksum

Error (1 byte): Error code
11h: Checksum error
2Ah: Address error (the specified address is not in the target mat)
53h: Programming cannot be done due to a programming error



Downloaded from Arrow.com.

### (4) Erasure Selection

In response to an erasure selection command sent from the host, the RX610 selects the erasure program and waits for erasure block specification.

| | |
|---|---|
| Command | 48h |
| Response | 06h |

### (5) Block Erasure

In response to a block erasure command sent from the host, the RX610 erases the ROM. After completing ROM erasure successfully, the RX610 returns a response (06h). If an error has occurred during ROM erasure, the RX610 returns an error response (D8h).

| | | | | |
|---|---|---|---|---|
| Command | 58h | Size | Block | SUM |
| Response | 06h | | | |
| Error response | D8h | Error | | |

[Legend]

Size (1 byte): Number of bytes in the block specification field (fixed at 1)

Block (1 byte): Block number whose data is to be erased
To terminate erasure, FFh should be specified.

SUM (1 byte): Checksum

Error (1 byte): Error code
11h: Checksum error
29h: Block number error (an incorrect block number is specified)
51h: Erasure cannot be done due to an erasure error



Downloaded from Arrow.com.

### (6) Memory Read

In response to a memory read command sent from the host, the RX610 reads data from the ROM. After completing ROM reading successfully, the RX610 returns the data stored in the address specified by the memory read command. If the RX610 has failed to read the ROM, the RX610 returns an error response (D2h).

| | | | | |
|---|---|---|---|---|
| Command | 52h | Size | Area | Read start address |
| | Reading size | | | SUM |

| | | | | |
|---|---|---|---|---|
| Response | 52h | Reading size | | |
| | Data | Data | . . . | Data |
| | SUM | | | |

| | | |
|---|---|---|
| Error response | D2h | Error |

[Legend]

Size (1 byte): Total number of bytes in the area, read start address, and reading size fields

Area (1 byte): Target mat to be read
00h: User boot mat
01h: User mat

Read start address (4 bytes): Start address of the area to be read

Reading size (4 bytes): Size of data to be read (bytes)

SUM (1 byte): Checksum

Data (1 byte): Data read from the ROM

Error (1 byte): Error code
11h: Checksum error
2Ah: Address error
- The value specified for area selection is neither 00h nor 01h.
- The specified read start address is outside the selected mat.

2Bh: Data size error
- 00h is specified for the reading size.
- The reading size is larger than the mat.
- The end address calculated from the read start address and the reading size is outside the selected mat.



Downloaded from Arrow.com.

### (7) User Boot Mat Checksum

In response to a user boot mat checksum command sent from the host, the RX610 sums the user boot mat data in byte units and returns the result (checksum).

| | | | | |
|---|---|---|---|---|
| Command | 4Ah | | | |
| Response | 5Ah | Size | Mat checksum | SUM |

[Legend]

Size (1 byte): Number of bytes in the mat checksum field (fixed at 4)

Mat checksum (4 bytes): Checksum of the user boot mat data

SUM (1 byte): Checksum (for the response data)

### (8) User Mat Checksum

In response to a user mat checksum command sent from the host, the RX610 sums the user mat data in byte units and returns the result (checksum).

| | | | | |
|---|---|---|---|---|
| Command | 4Bh | | | |
| Response | 5Bh | Size | Mat checksum | SUM |

[Legend]

Size (1 byte): Number of bytes in the mat checksum field (fixed at 4)

Mat checksum (4 bytes): Checksum of the user mat data
The user mat also stores the key code for debugging function authentication. Note that the checksum includes this key code value.

SUM (1 byte): Checksum (for the response data)

### (9) User Boot Mat Blank Check

In response to a user boot mat blank check command sent from the host, the RX610 checks whether the user boot mat is completely erased. When the user boot mat is completely erased, the RX610 returns a response (06h). If the user boot mat has an unerased area, the RX610 returns an error response (sends CCh and 52h in that order).

| | | |
|---|---|---|
| Command | 4Ch | |
| Response | 06h | |
| Error response | CCh | 52h |



Downloaded from Arrow.com.

### (10) User Mat Blank Check

In response to a user mat blank check command sent from the host, the RX610 checks whether the user mat is completely erased. When the user mat is completely erased, the RX610 returns a response (06h). If the user mat has an unerased area, the RX610 returns an error response (sends CDh and 52h in that order).

| | | |
|---|---|---|
| Command | 4Dh | |
| Response | 06h | |
| Error response | CDh | 52h |

### (11) Read Lock Bit Status

In response to a read lock bit status command sent from the host, the RX610 reads data from the lock bit. After completing the lock bit reading successfully, the RX610 returns the data stored in the address specified by the read lock bit status command. If the RX610 has failed to read the lock bit, the RX610 returns an error response (F1h).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Command | 71h | Size | Area | Third highest order address | Second highest order address | Highest order address | SUM |
| Response | Status | | | | | | |
| Error response | F1h | Error | | | | | |

[Legend]

Size (1 byte): Total number of bytes in the area, third highest order address, second highest order address, and highest order address fields (fixed at 4 in the RX610)

Area (1 byte): Target mat to be read
01h: User mat

Third highest order address (1 byte): Third highest order address at the specified block's end address (8 to 15 bits)

Second highest order address (1 byte): Second highest order address at the specified block's end address (16 to 23 bits)

Highest order address (1 byte): Highest order address at the specified block's end address (24 to 31 bits)

SUM (1 byte): Checksum

Status (1 byte): Bit 6 locked at "0"
Bit 6 unlocked at "1"

Error (1 byte): Error code
11h: Checksum error
2Ah: Address error (the specified address is not in the target mat)



Downloaded from Arrow.com.

### (12) Lock Bit Program

In response to a lock bit program command sent from the host, the RX610 writes to a lock bit and locks the specified block. After completing the lock bit blocking successfully, the RX610 returns a response (06h). If the RX610 has failed to lock, the RX610 returns an error response (F7h).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Command | 77h | Size | Area | Third highest order address | Second highest order address | Highest order address | SUM |
| Response | 06h | | | | | | |
| Error response | F7h | Error | | | | | |

[Legend]

Size (1 byte): Total number of bytes in the area, third highest order address, second highest order address, and highest order address fields (fixed at 4 in the RX610)

Area (1 byte): Target mat to be locked
01h: User mat

Third highest order address (1 byte): Third highest order address at the specified block's end address (8 to 15 bits)

Second highest order address (1 byte): Second highest order address at the specified block's end address (16 to 23 bits)

Highest order address (1 byte): Highest order address at the specified block's end address (24 to 31 bits)

SUM (1 byte): Checksum

Error (1 byte): Error code
11h: Checksum error
2Ah: Address error (the specified address is not in the target mat)
53h: Locking cannot be done due to a programming error

### (13) Lock Bit Enable

In response to a lock bit enable command sent from the host, the RX610 enables a lock bit.

| | |
|---|---|
| Command | 7Ah |
| Response | 06h |



Downloaded from Arrow.com.

### (14) Lock Bit Disable

In response to a lock bit disable command sent from the host, the RX610 disables a lock bit.

| | |
|---|---|
| Command | 75h |
| Response | 06h |

### (15) Embedded Program Status Inquiry

For details, refer to section 26.10.5, Inquiry/Selection Host Command Wait State.



Downloaded from Arrow.com.

## 26.11 ID Code Protection on Connection of the On-Chip Debugger

This function is used to prohibit connection with the on-chip debugger. When connecting an on-chip debugger, the control code and ID code that have been written to the ROM are used to determine whether ID code protection on connection of the on-chip debugger is enabled or disabled and to judge ID code protection on connection of the on-chip debugger. When the ID code protection is enabled, the code sent from the on-chip debugger is compared with the control code and ID code in the ROM to determine whether they match. If they match, connection with the on-chip debugger is allowed. If they do not match, the on-chip debugger cannot be connected. However, if the control code is 52h and the ID code is 50h, 72h, 6Fh, 74h, 65h, 63h, 74h, FFh, ..., FFh (from the ID code 1 field), there is no determination of matching and the ID code is always considered to be non-matching. Furthermore, if all bytes of the control code and ID code have the value FFh, there is no determination of matching, the ID code is always considered to match, and connection of the on-chip debugger is allowed. See figure 26.25 for the configuration of ID codes in flash memory.

**Table 26.18 Specifications for ID Code Protection on Connection of the On-Chip Debugger**

| Control Code | ID Code | State of Protection | Operations at the Time of Connection with the On-Chip Debugger |
|---|---|---|---|
| FFh | FFh, …, FFh (all bytes FFh) | Protection disabled | The control code and ID code are not judged, the ID code always matches, and connection to the on-chip debugger is permitted. |
| 52h | 50h, 72h, 6Fh, 74h, 65h, 63h, 74h, FFh, ..., FFh | Protection enabled | The control code and ID code are not judged, the ID code is always non-matching, and connection to the on-chip debugger is prohibited. |
| Other than the above | Other than the above | Protection enabled | Matching ID code: Authentication of the on-chip debugger is ended and connection with the on-chip debugger is permitted.<br>Non-matching ID code: Further transition to the ID code protection waiting state |



Downloaded from Arrow.com.

## 26.12 ROM Code Protection

ROM code protection is a facility for prohibiting a PROM programmer from reading from or writing to flash memory. The ROM code in flash memory is a 32-bit code. Figure 26.33 shows the configuration of ROM codes. Set ROM codes as 32-bit units.

For release from ROM code protection, erase block EB00 (erasure block 00) in boot mode or by user programming.

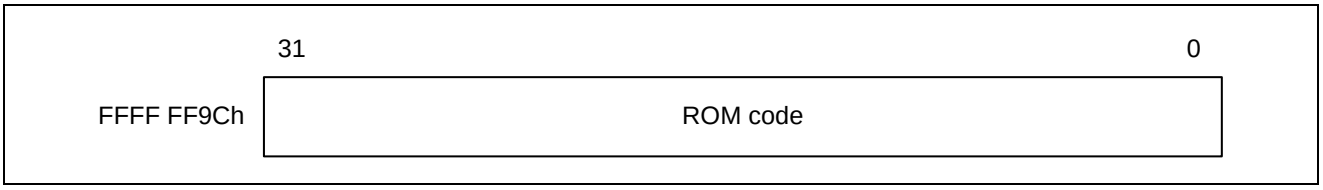



**Figure 26.33 Configuration of a ROM Code**

**Table 26.19 Specifications for ROM Code Protection**

| ROM Code | State of Protection | Operations at the Time of Connection with the PROM Programmer |
|---|---|---|
| 0000 0000h | Protection enabled (ROM code protection 1) | Access (both reading and writing) to the user mat and user boot mat is prohibited. |
| 0000 0001h | Protection enabled (ROM code protection 2) | Reading from the user mat and user boot mat is prohibited. |
| Other than the above | Protection disabled | Access (both reading and writing) to the user mat and user boot mat is permitted. |



Downloaded from Arrow.com.

## 26.13 Usage Notes

### (1) Areas where Programming or Erasure is Suspended

Data in areas where programming or erasure is suspended are undefined. To avoid malfunctions due to the reading of undefined data, prevent the reading of data and execution of code from areas where programming or erasure is currently suspended.

### (2) Suspending Programming or Erasure

If you use the programming/erasure suspension command to suspend the processing of programming or erasure, be sure to use the resumption command so that the processing is completed. Within 20µs (when PCLK = 50 MHz) after issuance of the resumption command, the programming/erasure suspension command should not be issued again.

### (3) Prohibition of Reprogramming

Two or more programming operations cannot be performed for the same address range. If an address range that has already been programmed is to be programmed again, be sure to erase the area in advance of the programming.

### (4) Reset during Programming or Erasure

Do not apply the active level to the RES# pin during programming or erasure because a reset may damage the flash memory. If the reset signal is erroneously input, only deassert the signal after assertion of the signal within the range of operating voltage specified in the Electrical Characteristics has continued for at least 100 µs.

If programming or erasure is proceeding and the FRESET bit in FRESETR is used to reset the FCU , make sure that the reset state is maintained over the period tRESW2 (see section 29, Electrical Characteristics). Do not attempt to read a target ROM for programming or erasure during the period of an FCU reset.

A WDT reset, however, can be applied even during programming or erasure regardless of the time to be secured that was described above.

### (5) Prohibition of Non-Maskable Interrupts during Programming or Erasure

A non-maskable interrupt (an interrupt on the NMI pin) during programming or erasure will lead to fetching of the vector from the ROM, and the data read out will be undefined. Ensure that non-maskable interrupts are not generated during programming or erasure of the ROM. (This restriction only applies to the ROM).

### (6) Interrupt Vector Assignment During Programming or Erasure

The generation of interrupts during programming or erasure may lead to the fetching of vectors from the ROM. To prevent access to the ROM area due to the generation of interrupts, set the interrupt table register (INTB) of the CPU so that the destination for the fetching of interrupt vector is an area outside the ROM.

### (7) Abnormal Termination during Programming or Erasure

If a failure in programming or erasure is caused by a voltage beyond the specified range of operating voltage, a reset, an FCU reset by the FRESETR.FRSET bit, entry to the command-locked state due to error detection, or either of the prohibited items under (8), the lock bit may be set to 0 (protected). In this case, set the FPROTR.FPROTCN bit to 1 to issue a block erase command and erase the lock bit. After this, proceed with the failed programming or erasure again.



Downloaded from Arrow.com.

### (8) Actions Prohibited during Programming and Erasure

The following prohibitions must be observed to prevent damage to the flash memory during programming or erasure.

- Do not allow the power-supply voltage of RX610 to go outside the specified range for operation.
- Do not change the value of the FWEPROR.FLWE[1:0] bits.
- Do not change the operating mode by changing the setting of the SYSCR0.ROME bit.
- Do not change the multiplication ratio of PCLK by using the SCKCR register.
- Do not change the frequency of PCLK by changing the setting of the PCKAR register.
- Do not make a transition to all-module-clock-stopped mode, software standby mode, or deep software standby mode.



Downloaded from Arrow.com.

# 27. Data Flash (Flash Memory for Data Storage)

The RX610 has a maximum 2-Mbyte flash memory for storing program code (ROM) and a 32-Kbyte flash memory for storing data (data flash).

This section covers the flash memory for data storage. For the ROM, see section 26, ROM (Flash Memory for Code Storage).

## 27.1 Overview

Table 27.1 lists the specifications of the data flash memory, and figure 27.1 is a block diagram of the ROM, data-flash memory, and related modules.

**Table 27.1 Specifications of Data Flash Memory**

| Item | | Specifications |
|---|---|---|
| Memory space | | 32 Kbytes |
| Reading via the peripheral bus | | A read operation takes three cycles of PCLK in words or bytes |
| Programming/erasing method | | • The chip incorporates a dedicated sequencer (FCU) for programming of the ROM and data flash.<br>• Programming and erasing the ROM and data flash are handled by issuing commands to the FCU. |
| BGO (background operation) | | • The CPU is able to execute program code from areas other than the ROM or data flash while the ROM is being programmed or erased.<br>• Execution of program code from the ROM is possible while the data flash memory is being programmed or erased. |
| Suspension and resumption | | • The CPU is able to execute program code from the ROM during suspension of programming or erasure.<br>• Programming and erasure of the ROM can be restarted (resumed) after suspension. |
| Units of programming and erasure | | • Unit of programming for the data mat: 8 or 128 bytes<br>• Unit of erasure for the data mat: 8 Kbytes (4 blocks) |
| Blank checking function | | • The blank checking command can be executed to check the erasure state of data flash.<br>• The size of the area to be blank-checked is 8 bytes or 8Kbytes. |
| On-board programming (three types) | Boot mode | • The data mat is programmable via the SCI.<br>• The bit rate for SCI transfer between the host and RX610 is automatically adjusted. |
| | User boot mode | Booting up from the user boot mat and programming of the data mat |
| | User program | Programming of the data mat under program control |
| Protection | Software-controlled protection | The FENTRYR.FENTRYD and FWEPROR.FLWE[1:0] bits, and the DFLRE and DFLWE registers, can be used to prevent unintentional programming. |
| | Error protection | Prevention of further programming or erasure after the detection of abnormal operations during programming or erasure |
| Times for programming and erasure, durability (number of times reprogramming is poss ble) | | See section 29, Electrical Characteristics. |



Downloaded from Arrow.com.

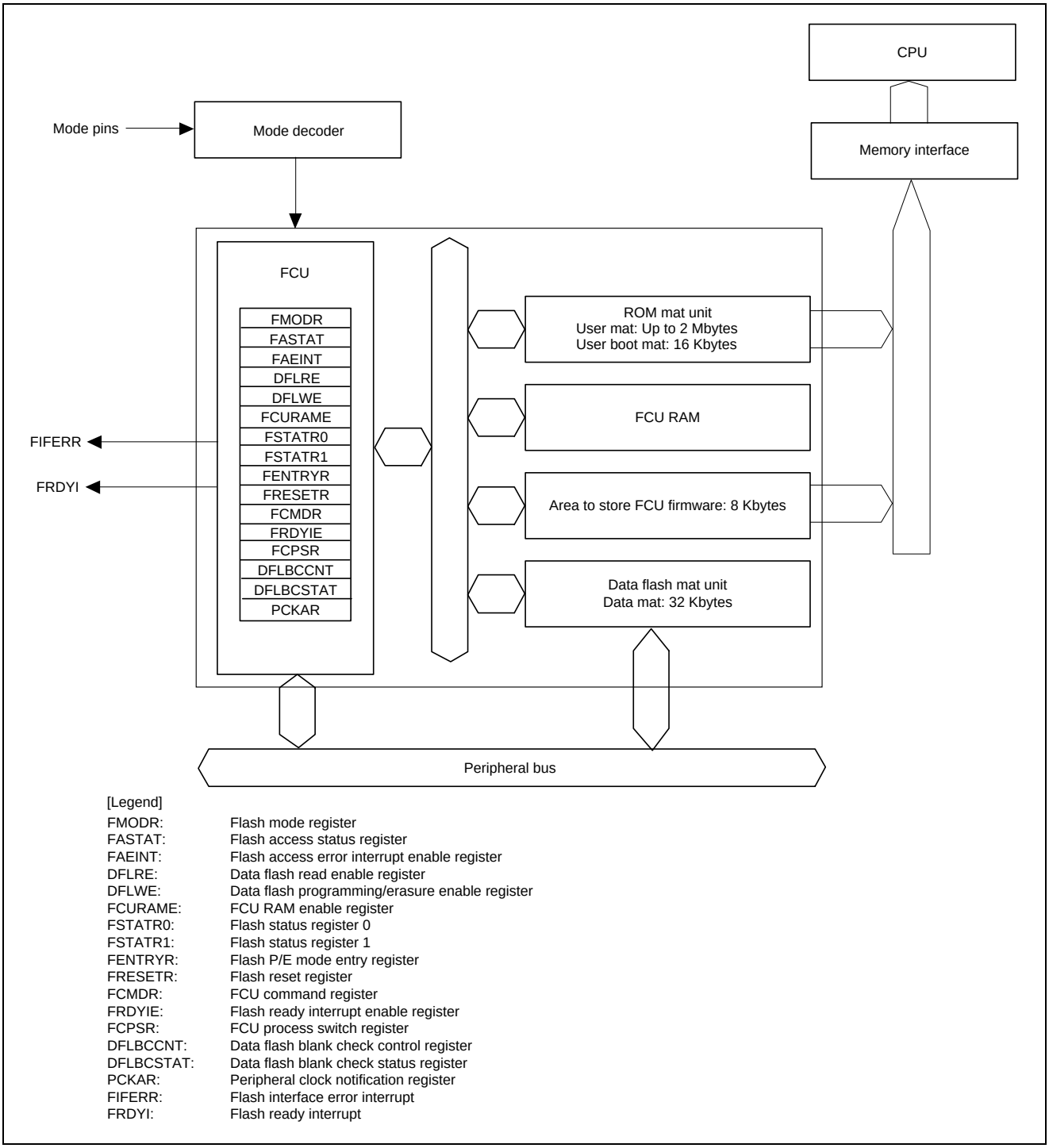



**Figure 27.1 Block Diagram of Data Flash Memory**

Input and output pins associated with the data flash are listed in table 27.2.

**Table 27.2 Input and Output Pins Associated with the Data Flash**

| Pin Name | I/O | Description |
|---|---|---|
| P05/RxD4 | Input | Used in boot mode to receive data via SCI4 (for host communications) |
| P04/TxD4 | Output | Used in boot mode to transmit data from SCI4 (for host communications) |



Downloaded from Arrow.com.

## 27.2 Register Descriptions

Table 27.3 lists the registers related to the data flash memory. Some registers also have bits related to the ROM, but this section deals only with the bits that are relevant to the data flash. For registers containing bits with common functions for the ROM and data flash (FRDYIE, FCURAME, FSTATR0, FSTATR1, FRESETR, FCMDR, FCPSR, PCKAR, and FWEPROR) and details on the functions of bits dedicated to the ROM, see section 26.2, Register Descriptions.

The registers related to the data flash are initialized by a reset.

**Table 27.3 Registers Related to the Data Flash**

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Flash mode register | FMODR | 00h | 007F C402h | 8 |
| Flash access status register | FASTAT | 00h | 007F C410h | 8 |
| Flash access error interrupt enable register | FAEINT | 9Bh | 007F C411h | 8 |
| Flash ready interrupt enable register | FRDYIE | 00h | 007F C412h | 8 |
| Data flash read enable register | DFLRE | 0000h | 007F C440h | 16 |
| Data flash programming/erasure enable register | DFLWE | 0000h | 007F C450h | 16 |
| FCU RAM enable register | FCURAME | 0000h | 007F C454h | 16 |
| Flash status register 0 | FSTATR0 | 80h | 007F FFB0h | 8 |
| Flash status register 1 | FSTATR1 | 00h | 007F FFB1h | 8 |
| Flash P/E mode entry register | FENTRYR | 0000h | 007F FFB2h | 16 |
| Flash reset register | FRESETR | 0000h | 007F FFB6h | 16 |
| FCU command register | FCMDR | FFFFh | 007F FFBAh | 16 |
| FCU processing switching register | FCPSR | 0000h | 007F FFC8h | 16 |
| Data flash blank check control register | DFLBCCNT | 0000h | 007F FFCAh | 16 |
| Data flash blank check status register | DFLBCSTAT | 0000h | 007F FFCEh | 16 |
| Peripheral clock notification register | PCKAR | 0000h | 007F FFE8h | 16 |
| Flash write erase protection register | FWEPROR | 02h | 0008 C289h | 8 |



Downloaded from Arrow.com.

### 27.2.1 Flash Mode Register (FMODR)

Address: 007F C402h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | FRDMD | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b4 | FRDMD | FCU Read Mode Select | 0: Memory Area Reading Method<br>The method of lock bit reading is set. Since there are no lock bits for the data flash, undefined data are read from the data flash area after the FCU has been placed in lock bit read mode.<br>1: Register Reading Method<br>This is the setting when the blank checking command is to be used. | R/W |
| b7 to b5 | — | Reserved | These bits are always read as 0.The write value should always be 0. | R/W |

FMODR is used to specify the method for the reading of lock bits. Set the FRDMD bit to 1 if blank checking is to be used.

In modes in which the on-chip ROM is disabled, the value read from FMODR is 00h and writing is disabled.

FMODR is initialized by a reset.

#### FRDMD Bit (FCU Read Mode Select)

Processing for a transition of the lock bit reading mode is used in processing for blank checking of the data flash.

The FRDMD bit is used to select the method of reading when lock bit values for the ROM are read out (see section 26, ROM (Flash Memory for Code Storage)).



Downloaded from Arrow.com.

### 27.2.2 Flash Access Status Register (FASTAT)

Address: 007F C410h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ROMAE | – | – | CMDLK | DFLAE | – | DFLRPE | DFLWPE |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DFLWPE | Data Flash Programming/Erasure Protection Violation | 0: No data flash programming/erasure command is issued which conflicts with the DFLWE settings<br>1: A data flash programming/erasure command is issued which conflicts with the DFLWE settings | R/(W)* |
| b1 | DFLRPE | Data Flash Read Protection Violation | 0: There is no such data flash read that conflicts with the DFLRE settings<br>1: There is such data flash read that conflicts with the DFLRE setting | R/(W)* |
| b2 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b3 | DFLAE | Data Flash Access Violation | 0: No data flash access violation<br>1: Data flash access violation | R/(W)* |
| b4 | CMDLK | FCU Command Lock | 0: FCU is not in the command-locked state<br>1: FCU is in the command-locked state | R |
| b6, b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | ROMAE | ROM Access Violation | See section 26, ROM (Flash Memory for Code Storage). | R/(W)* |

Note: * Only 0 can be written after reading 1 to clear the flag.

FASTAT is a register to check if the access to the ROM/data flash is allowed.

When on-chip ROM is disabled, the data read from FASTAT is 00h and writing is disabled. When one of the bits in FASTAT is set to 1, the FCU is placed in the command-locked state (see section 27.7.2, Error Protection). To clear the command-locked state, a status clear command must be issued to the FCU after setting FASTAT to 10h.

FASTAT is initialized by a reset.



Downloaded from Arrow.com.

### EEPWPE Bit (Data Flash Programming/Erasure Protection Violation)

This bit is used to indicate whether or not the programming/erasure protection set by DFLWE is violated.

[Setting condition]

- A programming/erasure command is issued for a data flash area for which programming or erasure is disabled by DFLWE.

[Clearing condition]

- When 0 is written after reading 1

### DFLRPE Bit (Data Flash Read Protection Violation)

This bit is used to indicate whether or not the reading protection set by DFLRE is violated.

[Setting condition]

- A read command is issued for a data flash area for which reading is disabled by DFLRE.

[Clearing condition]

- When 0 is written after reading 1

### DFLAE Bit (Data Flash Read Protection Violation)

This bit indicates whether a data flash access violation occurred.

When the DFLAE bit is set to 1, the ILGLERR bit in FSTATR0 is set to 1, placing the FCU in the command-locked state.

For FSTATR0, see section 26.2.5, Flash Status Register 0 (FSTATR0).

[Setting conditions]

- A read command is issued for a data flash area in data flash P/E normal mode and when the FENTRYD bit in FENTRYR is set to 1.
- A write command is issued for a data flash area when the FENTRYD bit is set to 0.
- A command is issued for a data flash area when the FENTRY1 or FENTRY0 bit in FENTRYR is set to 1.

[Clearing condition]

- When 0 is written after reading 1

### CMDLK Bit (FCU Command-Locked)

This command indicates that the FCU is in the command-locked state (see section 27.7.2, Error Protection).

[Setting condition]

- After the FCU detects an error and enters the command-locked state

[Clearing condition]

- After the FCU has processed a status clear command



Downloaded from Arrow.com.

### 27.2.3 Flash Access Error Interrupt Enable Register (FAEINT)

Address: 007F C411h

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | ROMAEIE | – | – | CMDLKIE | DFLAEIE | – | DFLRPEIE | DFLWPEIE |
| Value after reset: | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DFLWPEIE | Data Flash Programming/Erasure Protection Violation Interrupt Enable | 0: FIFERR interrupt requests disabled when the DFLWPE bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the DFLWPE bit in FASTAT is set to 1 | R/W |
| b1 | DFLRPEIE | Data Flash Read Protection Violation Interrupt Enable | 0: FIFERR interrupt requests disabled when the DFLRPE bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the DFLRPE bit in FASTAT is set to 1 | R/W |
| b2 | — | Reserved | This bit is always read as 0. The write value should always be 0. | R/W |
| b3 | DFLAEIE | Data Flash Read Access Violation Interrupt Enable | 0: FIFERR interrupt requests disabled when the DFLAE bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the DFLAE bit in FASTAT is set to 1 | R/W |
| b4 | CMDLKIE | FCU Command Lock Interrupt Enable | 0: FIFERR interrupt requests disabled when the CMDLK bit in FASTAT is set to 1<br>1: FIFERR interrupt requests enabled when the CMDLK bit in FASTAT is set to 1 | R/W |
| b6, b5 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | ROMAEIE | ROM Access Violation Interrupt Enable | See section 26, ROM (Flash Memory for Code Storage). | R/W |

FAEINT is a register to enable and disable a flash interface error interrupt (FIFERR).

When on-chip ROM is disabled, the data read from FAEINT is 00h and writing is disabled.

FAEINT is initialized by a reset.

#### DFLWPEIE Bit (Data Flash Programming/Erasure Protection Violation Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when a data flash programming/erasure protection violation occurs and the DFLWPE bit in FASTAT is set to 1.

#### DFLRPEIE Bit (Data Flash Read Protection Violation Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when a data flash read protection violation occurs and the DFLRPE bit in FASTAT is set to 1.



Downloaded from Arrow.com.

### DFLAEIE Bit (Data Flash Read Protection Violation Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when a data flash access violation occurs and the DFLAE bit in FASTAT is set to 1.

### CMDLKIE Bit (FCU Command Lock Interrupt Enable)

This bit is used to enable or disable FIFERR interrupt requests when a FCU command-locked state occurs and the CMDLK bit in FASTAT is set to 1.



Downloaded from Arrow.com.

## 27.2.4 Data Flash Read Enable Register (DFLRE)

Address: 007F C440h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | KEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | DBRE3 | DBRE2 | DBRE1 | DBRE0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DBRE0 | DB0 Block Read Enable | 0: Read disabled | R/W |
| b1 | DBRE1 | DB1 Block Read Enable | 1: Read enabled | R/W |
| b2 | DBRE2 | DB2 Block Read Enable | | R/W |
| b3 | DBRE3 | DB3 Block Read Enable | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | KEY[7:0] | Key Code | Enable or disable rewriting of the DBREj bit (j = 3 to 0). | R/(W)* |

Note: * Write data is not retained.

DFLRE is a register to enable or disable the DB0 to DB3 blocks (see figure 27.3) to be read.

Only specific values written to the upper byte in word access are valid. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from DFLRE is 0000h and writing is disabled.

DFLRE is initialized by a reset.

### DBREj Bit (DBj Block Read Enable) (j = 3 to 0)

This bit is used to enable or disable the DB3 to DB0 blocks of the data mat to be read.

The DBREj bit is used to control reading of the DBj blocks.

Writing to the DBREj is enabled only in word access when the KEY[7:0] bits are 2Dh.

### KEY[7:0] Bits (Key Code)

These bits are used to enable or disable rewriting of the DBREj bit.

Data written to the KEY[7:0] bits is not retained.



Downloaded from Arrow.com.

## 27.2.5 Data Flash Programming/Erasure Enable Register (DFLWE)

Address: 007F C450h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | KEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | DBWE3 | DBWE2 | DBWE1 | DBWE0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | DBWE0 | DB0 Block Programming/Erasure Enable | 0: Programming/erasure disabled | R/W |
| b1 | DBWE1 | DB1 Block Programming/Erasure Enable | 1: Programming/erasure enabled | R/W |
| b2 | DBWE2 | DB2 Block Programming/Erasure Enable | | R/W |
| b3 | DBWE3 | DB3 Block Programming/Erasure Enable | | R/W |
| b7 to b4 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b15 to b8 | KEY[7:0] | Key Code | Enable or disable rewriting of the DBWEj bit (j = 3 to 0). | R/(W)* |

Note: * Write data is not retained.

DFLWE is a register to enable or disable the DB0 to DB3 blocks (see figure 27.3) to be programmed or erased.

Only specific values written to the upper byte in word access are valid. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from DFLWE is 0000h and writing is disabled.

DFLWE is initialized by a reset.

### DBWEj Bit (DBj Block Programming/Erasure Enable) (j = 0 to 3)

This bit is used to enable or disable the DB3 to DB0 blocks of the data mat to be programmed or erased.

The DBWEj bit is used to control programming/erasure of the DBj blocks.

Programming of the DBWEj bit is enabled only in word access when the KEY[7:0] bits are 1Eh.

### KEY[7:0] Bits (Key Code)

These bits are used to enable or disable rewriting of the DBWEj bit.

Data written to the KEY[7:0] bits is not retained.



Downloaded from Arrow.com.

### 27.2.6 Flash P/E Mode Entry Register (FENTRYR)

Address: 007F FFB2h

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | FEKEY[7:0] | | | | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | FENTRYD | – | – | – | – | – | FENTRY1 | FENTRY0 |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | FENTRY0 | ROM P/E Mode Entry 0 | See section 26, ROM (Flash Memory for Code Storage). | R/W |
| b1 | FENTRY1 | ROM P/E Mode Entry 1 | See section 26, ROM (Flash Memory for Code Storage). | R/W |
| b6 to b2 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b7 | FENTRYD | Data Flash P/E Mode Entry | 0: Data flash is in read mode<br>1: Data flash is in P/E mode | R/W |
| b15 to b8 | FEKEY[7:0] | Key Code | Enable or disable rewriting of the FENTRYD, FENTRY1 and FENTRY0 bits. | R/(W)* |

Note: * Write data is not retained.

FENTRYR is a register to place the ROM/data flash in P/E mode.

To place ROM/data flash in P/E mode and accept commands from the FCU, one of the FENTRYD, FENTRY1 and FENTRY0 bits must be set to 1. If more than one bit is set to 1, the ILGLERR bit is set in FSTATR0 and the FCU enters the command-locked state.

Only specific values written to the upper byte in word access are valid. Any other writing causes the register to be initialized. Data written to the upper byte is not retained.

When on-chip ROM is disabled, the data read from FENTRYR is 0000h and writing is disabled.

FENTRYR is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

For FSTATR0, see section 26.2.5, Flash Status Register 0 (FSTATR0).

For FRESETR, see section 26.2.10, Flash Reset Register (FRESETR).



Downloaded from Arrow.com.

### FENTRYD Bit (Data Flash P/E Mode Entry)

The FENTRYD bit is used to place the data flash in P/E mode.

[Writing-enable conditions (when all of the following conditions are met)]

- On-chip ROM is enabled.
- The FRDY bit in FSTATR0 is set to 1
- AAh is written to the FEKEY[7:0] bits in word access.

[Setting condition]

- When the writing-enable conditions are met, FENTRYR is set to 0000h, and 1 is written to the FENTRYD bit

[Clearing conditions]

- Data is written in byte access.
- Data is written in word access when the FEKEY[7:0] bits are other than AAh.
- When the writing-enable conditions are met, 0 is written to the FENTRYD bit.
- When the writing-enable conditions are met and FENTRYR is other than 0000h, data is written to FENTRYR.

### FEKEY[7:0] Bits (Key Code)

These bits enable or disable rewriting of the FENTRYD, FENTRY1 and FENTRY0 bits.

Data written to the FEKEY[7:0] bits is not retained.



Downloaded from Arrow.com.

## 27.2.7 Data Flash Blank Check Control Register (DFLBCCNT)

Address: 007F FFCAh

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | BCADR[9:0] | | | | |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | BCADR[9:0] | | | | | – | – | BCSIZE |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | BCSIZE | Blank Check Size Setting | 0: The size of the area to be blank-checked is 8 bytes.<br>1: The size of the area to be blank-checked is 8 Kbytes. | R/W |
| b2, b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |
| b12 to b3 | BCADR[9:0] | Blank Check Address Setting | Set the address of the area to be checked | R/W |
| b15 to b13 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DFLBCCNT is a register for specifying the address and size of the area to be checked by a blank check command.

When on-chip ROM is disabled, the data read from DFLBCCNT is 0000h and writing is disabled.

DFLBCCNT is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

For FRESETR, see section 26.2.10, Flash Reset Register (FRESETR).

### BCSIZE Bit (Blank Check Size Setting)

The BCSIZE bit is used to set the size of the area to be checked by a blank check command.

### BCADR[9:0] Bits (Blank Check Address Setting)

When the size of the area to be checked by a blank check command is 8 bytes (the BCSIZE bit is set to 0), this bit is used to set the address of the area to be checked.

When the BCSIZE bit is set to 0, the setting of DFLBCCNT (the setting of the BCADR bit shifted three bits in the MSB direction) added with the erased block start address specified when issuing a blank check command is the start address of the area to be checked.



Downloaded from Arrow.com.

## 27.2.8 Data Flash Blank Check Status Register (DFLBCSTAT)

Address: 007F FFCEh

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | BCST |
| Value after reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b0 | BCST | Blank Check Status | 0: The area to be blank-checked is erased (blank)<br>1: 0 or 1 is written in the area to be blank-checked | R |
| b15 to b1 | — | Reserved | These bits are always read as 0. The write value should always be 0. | R/W |

DFLBCSTAT is a register which stores the results of a blank check command.

When on-chip ROM is disabled, the data read from DFLBCSTAT is 0000h and writing is disabled.

DFLBCSTAT is initialized by a reset, or when the FRESET bit in FRESETR is set to 1.

For FRESETR, see section 26.2.10, Flash Reset Register (FRESETR).

### BCST Bit (Blank Check Status)

This bit is used to indicate the results of blank checking.



Downloaded from Arrow.com.

## 27.3 Configuration of Memory Mat for the Data Flash Memory

The data flash memory of products in the RX610 Group is configured as a 32-Kbyte data mat. The address range occupied by this mat is shown in figure 27.2.

Note that for the data mat, the address range for reading is the same as the address range for programming and erasure.

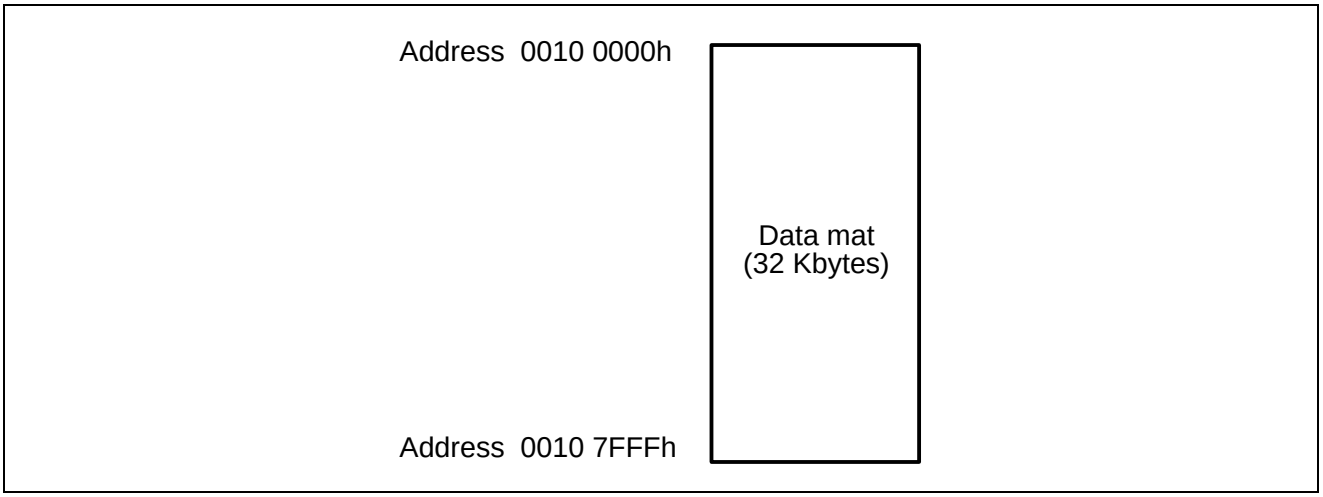



**Figure 27.2 Configuration of the Data Mat**

## 27.4 Block Configuration

The configuration of erasure blocks for the data mat is shown in figure 27.3. As units of erasure, the data mat is divided into 4 blocks of 8 Kbytes. Programming is in 8- or 128-byte units. Programming in eight-byte units proceeds for the eight bytes from an address for which the three lower-order bits are zero. Programming in 128-byte units proceeds for the 128 bytes from an address for which the lower-order byte is 00h or 80h.

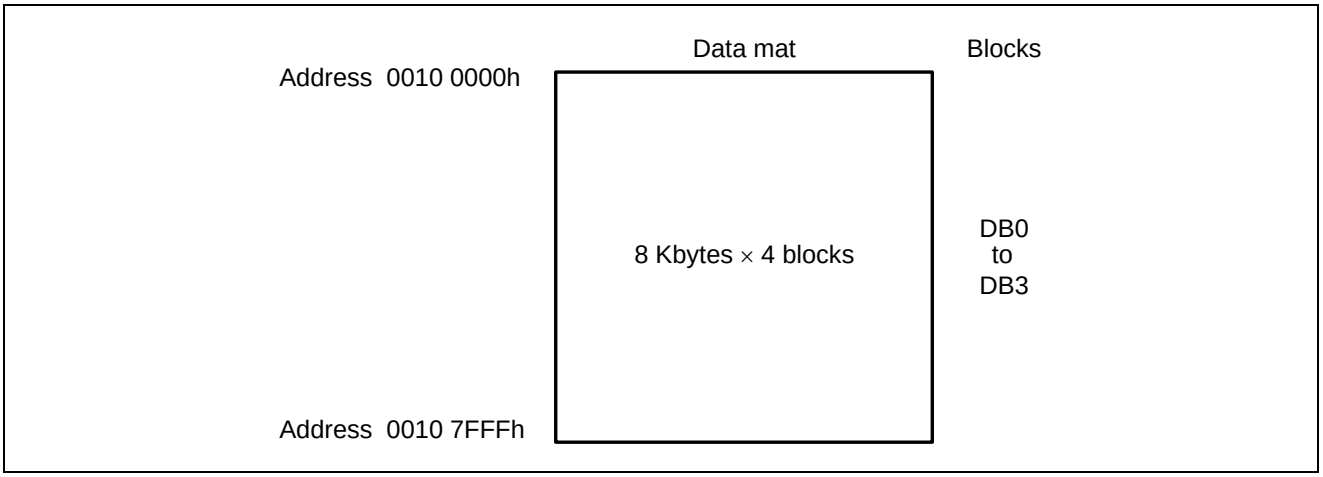



**Figure 27.3 Division of the Data Mat into Erasure Blocks**



Downloaded from Arrow.com.

## 27.5 Operating Modes Associated with the Data Flash

For the transitions between operating modes, see section 26.5, Operating Modes Associated with the ROM.

Reading, programming, and erasing of the data flash memory in an on-board device can proceed if the device is in boot, user-boot, or single-chip mode (with on-chip ROM enabled), or in on-chip-ROM-enabled expansion mode.

The differences between modes are indicated in table 27.4.

**Table 27.4 Differences between Modes**

| Item | Boot Mode | User Boot Mode | Single-Chip Mode (with On-chip ROM Enabled) or On-chip-ROM-Enabled Expansion Mode |
|---|---|---|---|
| Environment for programming and erasure | On-board programming | On-board programming | On-board programming |
| Programmable and erasable mat | Data mat | Data mat | Data mat |
| Division into erasure blocks | Poss ble*1 | Possible | Possible |
| Target mat for booting after a reset | Mat containing the embedded program*2 | User boot mat | User mat |

Notes: 1. All flash memory areas may be erased at the time of booting up. Specified blocks can subsequently be erased. For details, refer to section 26.10.4, ID Code Protection in Boot Mode.

2. Not available to users.

- In boot mode, a host is able to program or read out the data mat via an SCI.
- After booting up from the user boot mat in user boot mode, programming and erasure of the user mat or data mat via the desired interface is possible.
- In boot mode, on-chip RAM is employed for the embedded program for use in boot mode. For this reason, preserving the contents of on-chip RAM is not possible in this case.



Downloaded from Arrow.com.

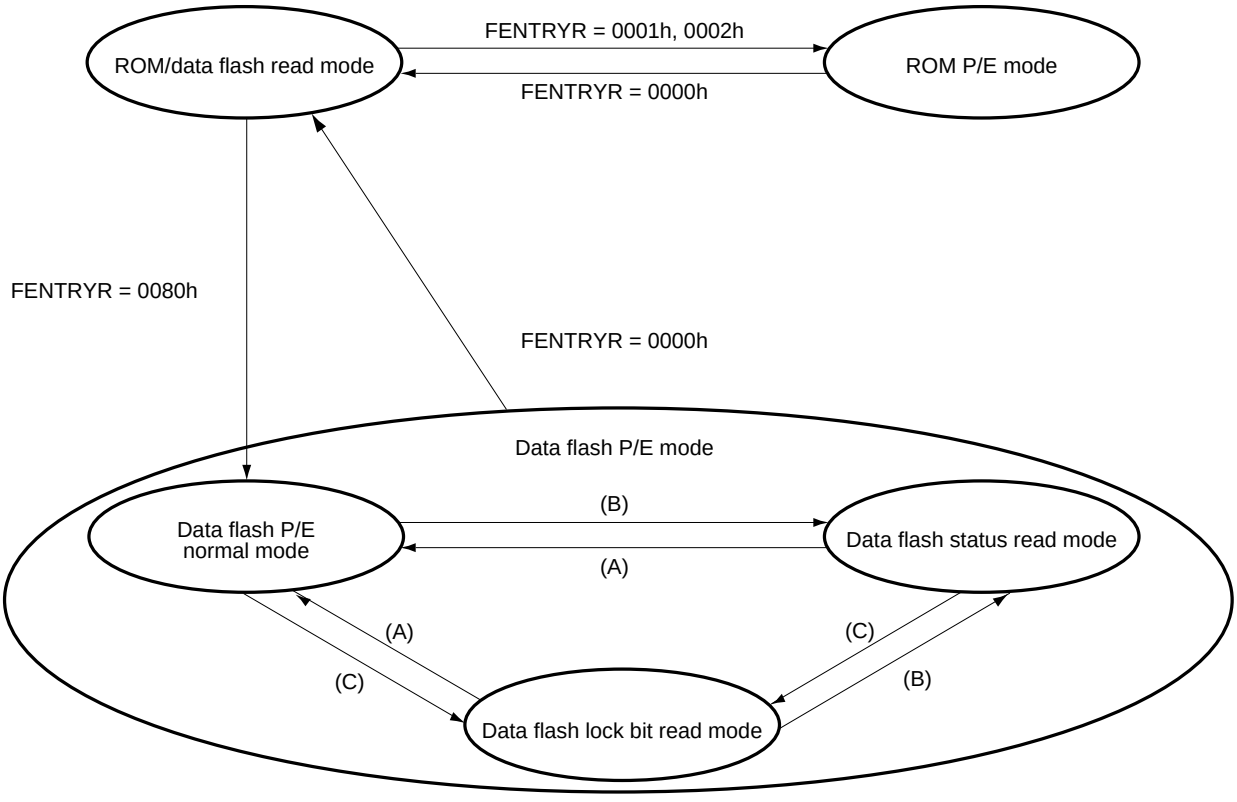
## 27.6 Programming and Erasing the Data Flash Memory

The data flash memory is programmed and erased by issuing commands to a dedicated sequencer (FCU) for programming and erasure. The FCU has five modes. For programming and erasure, the mode is changed and then commands for programming and erasure are issued.

The mode transitions required to program or erase the data flash and the system of commands are described below. The descriptions apply in common to boot, user-boot, and single-chip mode (with on-chip ROM enabled), and to on-chip-ROM-enabled expansion mode.

### 27.6.1 FCU Modes

The FCU has five modes or sets of modes. Transitions between modes are caused by writing to FENTRYR or issuing FCU commands. Figure 27.4 is a diagram of the FCU mode transitions.




[Legend]
(A): Execute a normal mode transition command.
(B): Execute a command that is neither a normal mode transition command nor a lock bit read mode transition command.
(C): Execute a lock bit read mode transition command.

**Figure 27.4 Mode Transitions of the FCU (Associated with the Data Flash)**



Downloaded from Arrow.com.

#### 27.6.1.1 ROM P/E Modes

The ROM P/E modes are for programming and erasure of the ROM.

For details on the ROM P/E modes, see section 26.6.1.2, ROM P/E Modes.

#### 27.6.1.2 ROM/Data Flash Read Mode

This mode is for reading the ROM or data-flash memory. The FCU does not accept commands.
The FCU enters this mode when the FENTRYD bit in FENTRYR is set to 0 and the FENTRY1 and FENTRY0 bits in FENTRYR are set to 0.

#### 27.6.1.3 Data Flash P/E Modes

These modes are for programming and erasure of the data-flash memory. Reading out the data flash is not possible.

Data flash P/E normal mode, data flash status read mode, and data flash lock-bit read mode are the three data flash P/E modes.

##### (1) Data Flash P/E Normal Mode

The transition to data flash P/E normal mode is the first transition in the process of programming or erasing the data flash. The FCU enters this mode when the FENTRYD bit in FENTRYR is set to 1 and the FENTRY1 and FENTRY0 bits in FENTRYR are set to 0 in ROM/data flash read mode, or when the normal mode transition command is received in data flash P/E modes. Table 27.7 lists the acceptable commands in this mode.

Read access to an address within the data flash area causes a data-flash-access violation, and the FCU enters the command-locked state. High-speed reading of the ROM is possible.

##### (2) Data Flash Status Read Mode

The data flash status read mode is for reading information on the state of the data flash.

The FCU enters this mode when a command other than the normal mode transition and lock bit read mode transition command is received in data flash P/E modes. Data flash status read mode encompasses the states where the FRDY bit in FSTATR0 is 0 and the command-locked state after an error has occurred. Table 27.7 lists the acceptable commands in this mode.

Read access to an address within the data flash area will actually read out the value of FSTATR0. High-speed reading of the ROM is possible.

##### (3) Data Flash Lock-Bit Read Mode

The data flash lock-bit read mode is for reading the values of the lock bits of the data flash. However, this is not possible because the data flash does not have lock bits.

The FCU enters this mode when a lock-bit read mode transition command is received in data flash P/E modes. Table 27.7 lists the acceptable commands in this mode.

Since the data flash does not have lock bits, data read out in read access to addresses within the data flash area are undefined. However, the access does not lead to a data-flash-access violation. High-speed reading of the ROM is possible.



Downloaded from Arrow.com.

## 27.6.2 FCU Commands

FCU commands consist of commands for mode transitions of the FCU and of commands for programming and erasure.

Table 27.5 lists the FCU commands for use with the data flash.

**Table 27.5 FCU Commands for Use with Data Flash Memory**

| Command | Description |
|---|---|
| Normal mode transition | Changes the mode to normal mode (see section 27.6.3, Connections between FCU Modes and Commands) |
| Status read mode transition | Changes the mode to status read mode (see section 27.6.3, Connections between FCU Modes and Commands) |
| Lock bit read mode transition (lock bit read 1) | Changes the mode to lock bit read mode (see section 27.6.3, Connections between FCU Modes and Commands) |
| Peripheral clock notification | Sets the frequency of the peripheral clock |
| Programming | Data flash programming (in 8-byte or 128-byte units) |
| Block erasure | Data flash erasure (in block units, with the lock bit being erased simultaneously) |
| P/E suspension | Suspends programming/erasure |
| P/E resumption | Resumes programming/erasure |
| Status register clearing | Clears the ILGLERR, ERSERR and PRGERR bits in FSTATR0 and releases the FCU from the command-locked state |
| Lock bit read 2/blank checking | Checks whether the specified block of data-flash memory has been erased (is blank) |

Commands other than the blank-checking command are also for use with the ROM.

The blank-checking command for the data-flash memory is also used as the lock bit read 2 command for the ROM. That is, when the same command is issued for the ROM, a lock bit of the ROM is read out.

Commands for the FCU are issued by write access to addresses within the data flash area.

Table 27.6 shows the formats of the programming commands and the blank checking command. Write access as listed in table 27.6 and in accord with certain conditions causes the FCU to execute processing for the corresponding command. For details on the conditions for acceptance of the individual FCU commands, see section 27.6.3, Connections between FCU Modes and Commands. For how to use the FCU commands, see section 27.6.4, FCU Command Usage.

**Table 27.6 FCU Command Formats**

| Command | Number of bus cycles | First Cycle | | Second Cycle | | Third Cycle | | 4th to (N+2)th Cycles | | (N+3)th Cycles | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Address | Data | Address | Data | Address | Data | Address | Data | Address | Data |
| Programming (8-byte programming; n = 4) | 7 | EA | E8h | EA | 04h | WA | WDn | EA | WDn | EA | D0h |
| Programming (128-byte programming; n = 64) | 67 | EA | E8h | EA | 40h | WA | WDn | EA | WDn | EA | D0h |
| Blank checking | 2 | EA | 71h | BA | D0h | — | — | — | — | — | — |

[Legend] Address column EA: Address within the data flash area: Any address in the range from 0010 0000h to 0010 7FFFh
WA: Start address of 8 bytes or 128 bytes
BA: Address in an erasure block of the data flash
Any address within the target erasure block
Data columns WDn: nth word of data for programming (n = 1 to N)



Downloaded from Arrow.com.

### 27.6.3 Connections between FCU Modes and Commands

The sets of FCU commands that can be accepted in each of the FCU modes are fixed. Furthermore, which commands are acceptable in a given FCU mode varies according to the state of the FCU.

Issuing of an FCU command must follow checking of the FCU's state after transitions of the FCU mode.

Commands that are acceptable in the various FCU modes and states are listed in table 27.7. Issuing a command that is not currently acceptable leads to the FCU being placed in the command-locked state (see section 27.7.2, Error Protection).

Issuing of an FCU command must follow checking of the values of the FRDY, ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and of the FCUERR bit in FSTATR1 after transitions of the FCU mode. Furthermore, the CMDLK bit in FASTAT can be checked to see if an error has occurred. The value of the CMDLK bit in FASTAT is the logical OR of the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and the FCUERR bit in FSTATR1.

Table 27.7 Acceptable Commands and the State and Mode (Data Flash P/E Mode) of the FCU

| | P/E Normal Mode | | | Status read mode | | | | | | Lock-bit read mode | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Programming suspended | Erasure suspended | Other state | Programming or erasure | Processing to suspend programming or erasure | Blank checking | Programming suspended | Erasure suspended | Command-locked state | Other state | Programming suspended | Erasure suspended | Other state |
| FSTATR0.FRDY bit | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0/1 | 1 | 1 | 1 | 1 |
| FSTATR0.SUSRDY bit | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| FSTATR0.ERSSPD bit | 0 | 1 | 0 | 0 | 0/1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| FSTATR0.PRGSPD bit | 1 | 0 | 0 | 0 | 0/1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| FASTAT.CMDLK bit | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Normal mode transition | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Status read transition | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Lock-bit read transition (lock bit read 1) | A | A | A | X | X | X | A | A | X | A | A | A | A |
| Peripheral clock notification | X | X | A | X | X | X | X | X | X | A | X | X | A |
| Programming | X | * | A | X | X | X | X | * | X | A | X | * | A |
| Block erasure | X | X | A | X | X | X | X | X | X | A | X | X | A |
| P/E suspension | X | X | X | A | X | X | X | X | X | X | X | X | X |
| P/E resumption | A | A | X | X | X | X | A | A | X | X | A | A | X |
| Status register clearing | A | A | A | X | X | X | A | A | A | A | A | A | A |
| Blank checking | A | A | A | X | X | X | A | A | X | A | A | A | A |

[Legend]

A: Acceptable

*: Only programming is acceptable for blocks other than the block where erasure was suspended

X: Not acceptable



Downloaded from Arrow.com.

## 27.6.4 FCU Command Usage

This section shows how to program and erase the data flash memory by using programming and block erasure commands, respectively, and how to check the state of erasure of the data flash by using the blank check command. For the method for transferring the firmware to the FCU RAM and the ways to use other FCU commands, see section 26.6.4, FCU Command Usage.

### (1) Using the Peripheral Clock Notification Command

This command handles notification of the frequency of the peripheral clock. For details, see section 26.6.4, FCU Command Usage. Set the FENTRYD bit in FENTRYR to 1 and make settings to indicate an address within the data flash area.

### (2) Programming

To program the data flash, use one of the programming commands.

Use byte access to write E8h to an address within the data flash area in the first cycle of the programming command, and the number of words (N)* to be programmed in the second cycle. Access the peripheral bus in words from the third cycle to cycle N + 2 of the command. In the third cycle, write the first word of data for programming to the address where the target area for programming starts. This address must be on an 8-byte boundary for 8-byte programming or on a 128-byte boundary address for 128-byte programming. After writing words to addresses in the data flash area N times, write byte D0h to an address within the data flash area in cycle N + 3; the FCU will then start actual programming of the data flash. Read the FRDY bit in FSTATR0 to confirm the completion of data flash programming.

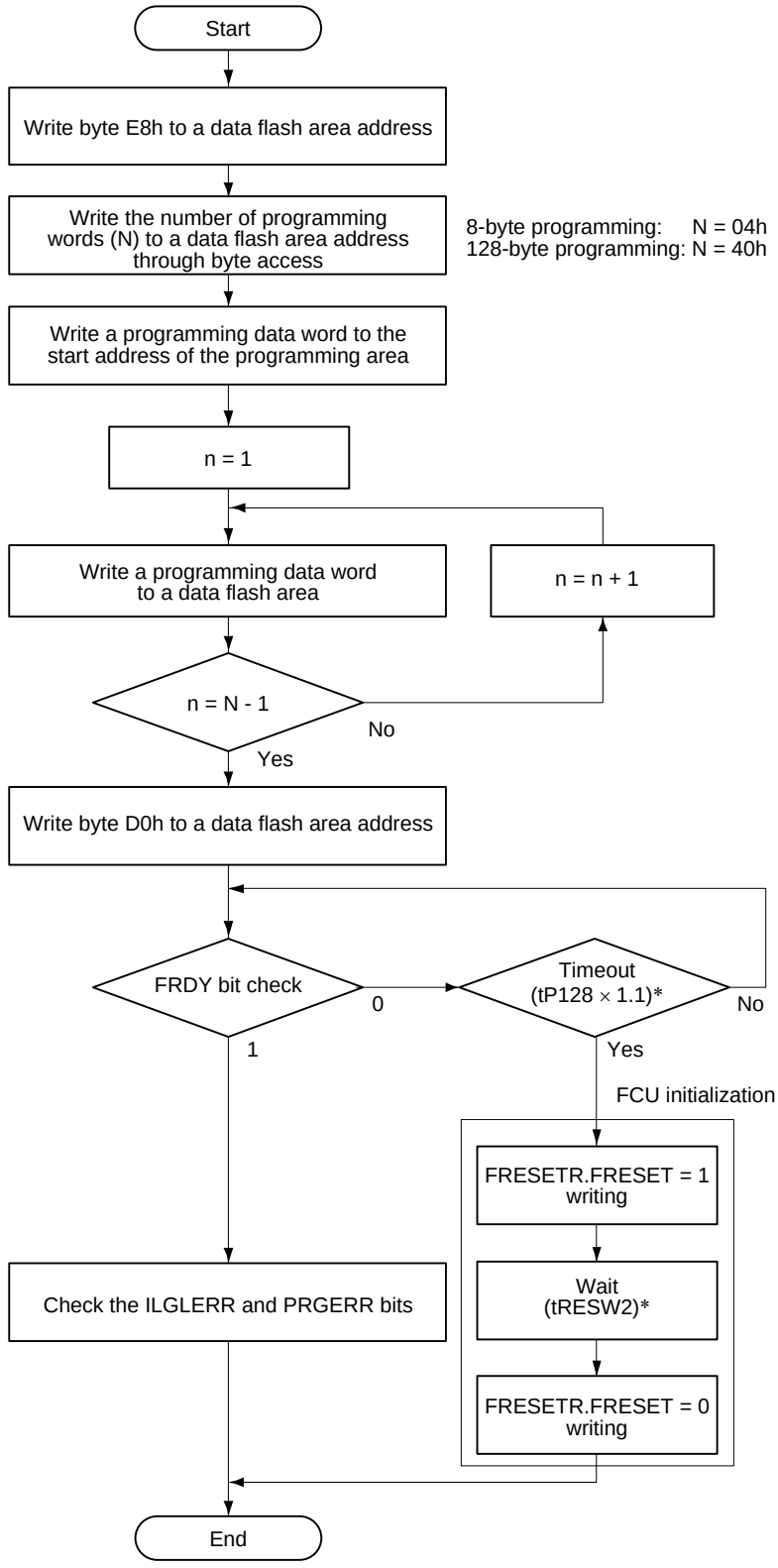
If the area accessed in the third cycle to cycle N + 2 includes addresses that do not require programming, write FFFFh as the programming data for those addresses. To ignore the programming and erasure protection provided by the DFLWE settings, set the program/erase enable bit for the target block to 1 before programming starts.

Figure 27.5 shows the procedure for data flash programming.

Note: * N = 04h for 8-byte programming or N = 40h for 128-byte programming.



Downloaded from Arrow.com.




Note: * tP128: Time required for programming 128-byte data (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 27.5 Procedure for Data Flash Programming**



Downloaded from Arrow.com.

### (3) Erasure

To erase the data flash, use the block erasure command. The data flash is erased in the same way as the ROM (see section 26, ROM (Flash Memory for Code Storage)).

Note that the data flash has a programming and erasure protection function that is controlled by DFLWE. Erasure can only be performed with protection provided by the DFLWE setting disabled, so set the programming/erasure enable bit for the target erasure block to 1 before issuing the erasure command.

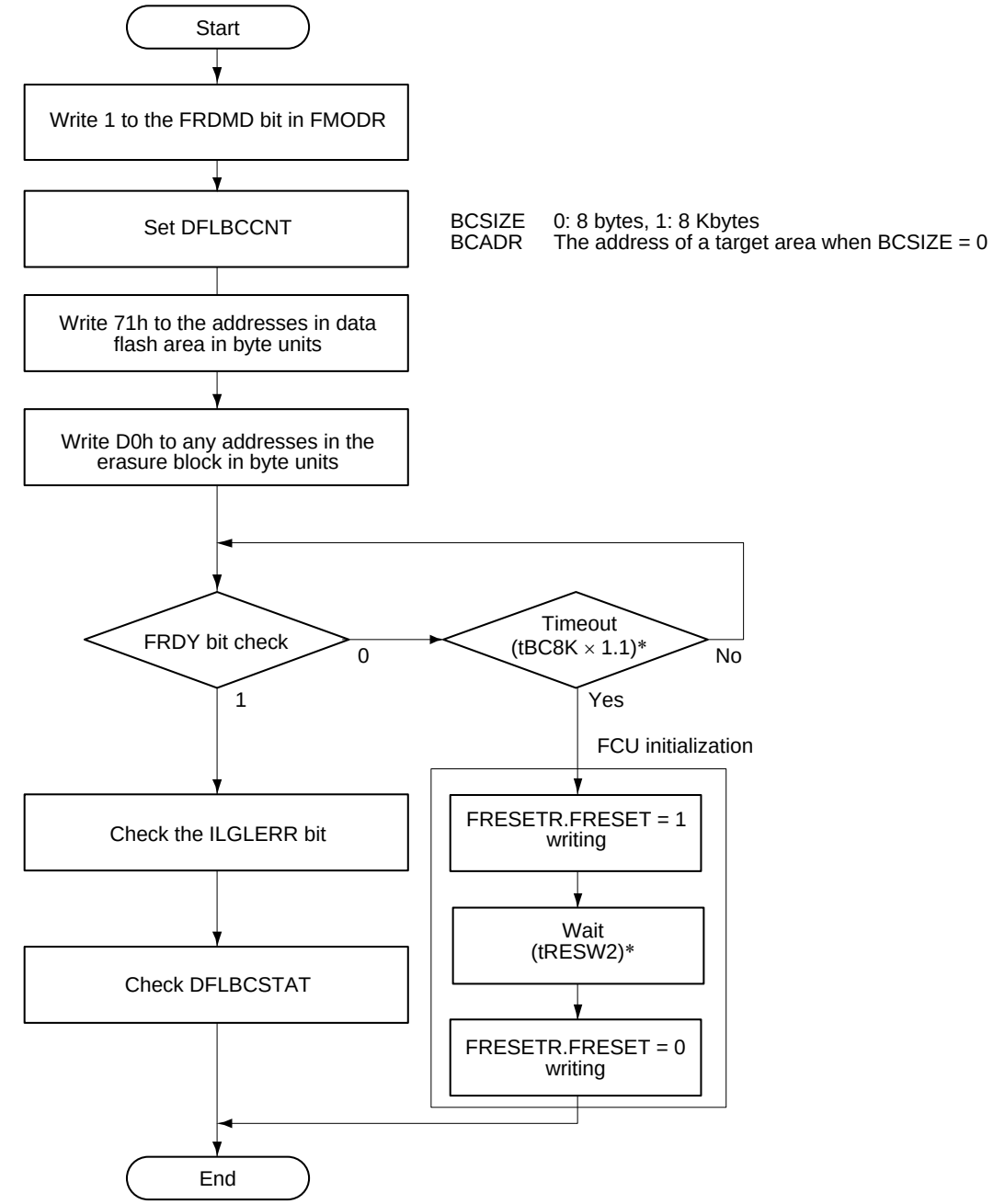
### (4) Blank Checking

Since using the CPU to read erased areas of the data flash produces undefined values, the blank checking command should be used to check whether the data flash has actually been erased. To make the blank checking command available for use, start by setting the FRDMD bit in FMODR to 1 to enable the command, and then specify the size and start address of the target area in DFLBCCNT. If the BCSIZE bit of DFLBCCNT is set to 1, checking will be performed for the entire erasure block (8 Kbytes) specified in the second cycle of the command. If the BCSIZE bit is set to 0, checking will be performed on the 8-byte range starting from the address obtained by adding the start address of the erased area as specified in the second cycle of the command and the value held by DFLBCCNT. In the first cycle of the command sequence, the value 71h is written as a byte unit to an address in the data flash. In the second cycle, when the value D0h is written to an address within the target area, the FCU starts blank checking of the data flash. Test the FRDY bit in FSTATR0 to check whether or not the check is complete. On completion of blank checking, check the BCST bit of DFLBCSTAT to see whether the target area has been erased or is filled with 0s and/or 1s via.

Figure 27.6 shows the procedure for blank checking of the data flash.



Downloaded from Arrow.com.




Note: tBC8K: Time required for blank check of 8-Kbyte data (see section 29, Electrical Characteristics)
tRESW2: Reset pulse width during programming/erasure (see section 29, Electrical Characteristics)

**Figure 27.6 Procedure for Blank Checking of the Data Flash**



Downloaded from Arrow.com.

## 27.7 Protection

There are two types of data flash programming/erasure protection: software protection and error protection.

### 27.7.1 Software Protection

In the software protection function, control register settings are used to disable data flash programming and erasure. If an attempt is made to issue a programming or erasure command for the data flash and the command violates current software protection, the FCU detects the error and enters the command-locked state.

#### (1) Protection through FWEPROR

If the FLWE[1:0] bits in FWEPROR are not set to 01b, programming cannot be performed in any mode.

#### (2) Protection through FENTRYR

When the FENTRYD bit in FENTRYR is 0, the ROM/data flash read mode is selected. Since the FCU does not accept commands in ROM/data flash read mode, data flash programming and erasure are disabled. If an attempt is made to issue an FCU command for the data flash in ROM/data flash read mode, the FCU detects an illegal command error and enters the command-locked state (see section 27.7.2, Error Protection).

#### (3) Protection through DFLWE

When the DBWEj (j = 3 to 0) bit in DFLWE is 0, programming and erasure of block DBj in the data mat is disabled. If an attempt is made to program or erase block DBj while the DBWEj bit is 0, the FCU detects a programming/erasure protection error and enters the command-locked state (see section 27.7.2, Error Protection).

#### (4) Protection through DFLRE

When the DBREj (j = 3 to 0) bit in DFLRE is 0, reading of block DBj in the data mat is disabled. If an attempt is made to read block DBj while the DBREj bit is 0, the FCU detects a read protection error and enters the command-locked state (see section 27.7.2, Error Protection).



Downloaded from Arrow.com.

### 27.7.2 Error Protection

Error protection is the detection of errors in the issuing of FCU commands and of prohibited access, and response in the form of notification of the FCU malfunction and prohibition of the reception of further commands by the FCU (the FCU enters the command-locked state). When the FCU enters the command-locked state (FASTAT.CMDLK bit is 1), one or several of the status bits (FSTATR0.ILGLERR, ERSERR, and PRGERR bits, FSTATR1.FCUERR bit, and FASTST.DFLAE, DFLRPE, and DFLWPE bits) are set to 1 and programming and erasure of the data-flash are prohibited. To release the FCU from the command-locked state, a status register clearing command must be issued with FASTAT set to 10h.

While the CMDLKIE bit in FAEINT is 1, a flash interface error (FIFERR) interrupt will be generated if the FCU enters the command-locked state (the CMDLK bit in FASTAT becomes 1). While a data flash-related interrupt enable bit (DFLAEIE, DFLRPEIE, or DFLWPEIE) in FAEINT is 1, an FIFERR interrupt will also be generated if the corresponding status bit (DFLAE, DFLRPE, or DFLWPE) in FASTAT becomes 1.

Table 27.8 shows the error protection types for the data flash and the values of the status bits (the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and the DFLAE, DFLRPE, and DFLWPE bits in FASTAT) after the detection of each type of error. For the error protection types used in common by the ROM and data flash (FENTRYR setting error, most illegal command errors, erasing errors, programming errors, and FCU errors, see section 26.8.2, Error Protection.

If the FCU enters the command-locked state due to a command other than a suspension command issued during programming or erasure processing, the FCU continues programming or erasing the data flash. In this state, the P/E suspension command cannot suspend programming or erasure. If a command is issued in the command-locked state, the ILGLERR bit becomes 1.

**Table 27.8 Error Protection Types (for Data Flash Only)**

| Error | Description | ILGLERR | ERSERR | PRGERR | DFLAE | DFLRPE | DFLWPE | CMDLK |
|---|---|---|---|---|---|---|---|---|
| Illegal command error | The value specified in the second cycle of a programming command was neither 04h nor 40h. | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| | A lock bit programming command was issued for an area in the data flash while the FENTRYD bit of FENTRYR register was set to 1. | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Data flash access error | A read access command was issued for the data flash area while FENTRYD = 1 in FENTRYR in data flash P/E normal mode. | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| | A write access command was issued for the data flash area while FENTRYD = 0. | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| | An access command was issued for the data flash area while the FENTRY1 or FENTRY0 bit in FENTRYR was 1. | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Data flash read protect error | A read access command was issued for the data flash area while it was protected against reading by the DFLRE setting. | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Data flash programming protect error | A program/block erase command was issued for the data flash area while it was protected against programming and erasure by the DFLWE setting. | 1 | 0 | 0 | 0 | 0 | 1 | 1 |



Downloaded from Arrow.com.

## 27.8 Boot Mode

To program or erase the data mat in boot mode, send control commands and programming data from the host. For the system configuration and settings in boot mode, see section 26.10, Boot Mode. This section describes only the commands dedicated for the data flash.

### 27.8.1 Inquiry/Selection Host Commands

Table 27.9 shows the inquiry/selection host commands dedicated to the data flash. The data mat inquiry and data mat information inquiry commands are used in the step of "Inquiry regarding mat programming information" in the flowchart shown in figure 26.29 (Example of Procedure to Use Inquiry/Selection Host Commands for User Mat/User Boot Mat) in section 26.10.5, Inquiry/Selection Host Command Wait State.

**Table 27.9 Inquiry/Selection Host Commands (only for Data Flash)**

| Host Command Name | Function |
|---|---|
| Data mat inquiry | Inquires regarding the availability of data mat |
| Data mat information inquiry | Inquires regarding the number of data mats and the start and end addresses |

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to the RX610 and the "response" indicates a response sent from the RX610 to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by the RX610 becomes 00h.

#### (1) Data Mat Inquiry

In response to a data mat inquiry command sent from the host, the RX610 returns the information concerning the availability of data mats.

| | | | | |
|---|---|---|---|---|
| Command | 2Ah | | | |
| Response | 3Ah | Size | Mat availability | SUM |

[Legend]

Size (1 byte): Number of characters in the mat availability field (fixed at 1)

Mat availability (1 byte): Availability of data mats (fixed at 21h)

21h: Data mat is available

SUM (1 byte): Checksum



Downloaded from Arrow.com.

### (2) Data Mat Information Inquiry

In response to a data mat information inquiry command sent from the host, the RX610 returns the number of data mat areas and their addresses.

| | | | |
|---|---|---|---|
| Command | 2Bh | | |
| Response | 3Bh | Size | Area count |
| | Area start address | | |
| | Area end address | | |
| | Area start address | | |
| | Area end address | | |
| | : | | |
| | Area start address | | |
| | Area end address | | |
| | SUM | | |

[Legend]

| | |
|---|---|
| Size (1 byte): | Total number of bytes in the area count, area start address, and area end address fields |
| Area count (1 byte): | Number of data mat areas (consecutive areas are counted as one area) |
| Area start address (4 bytes): | Start address of a data mat area |
| Area end address (4 bytes): | End address of a data mat area |
| SUM (1 byte): | Checksum |

The information concerning the block configuration in the data mat is included in the response to the erasure block information inquiry command (see section 26.10.5, Inquiry/Selection Host Command Wait State).



Downloaded from Arrow.com.

## 27.8.2 Programming/Erasing Host Commands

Table 27.10 shows the programming/erasing host commands dedicated to the data flash. Data flash-dedicated host commands are provided only for checksum and blank check of the data flash; the programming, erasing, and reading commands are used in common for the ROM and data flash.

To program the data mat, issue from the host a user mat programming selection command and then a 256-byte programming command specifying a data mat address as the programming address. To erase the data mat, issue an erasure selection command and then a block erasure command specifying an erasure block in the data mat. The information concerning the erasure block in the data mat is included in the response to the erasure block information inquiry command. To read data from the data mat, select the user mat through a memory read command specifying a data mat address as the read address.

For the user mat programming selection, user boot mat programming selection, 256-byte programming, erasure selection, block erasure, and memory read commands, refer to section 26.10.7, Programming/Erasing Host Command Wait State. For the erasure block information inquiry command, refer to section 26.10.5, Inquiry/Selection Host Command Wait State.

**Table 27.10 Programming/Erasure Host Commands (only for Data Flash)**

| Host Command Name | Function |
|---|---|
| Data mat checksum | Performs checksum verification for the data mat |
| Data mat blank check | Checks whether the data mat is blank |

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to the RX610 and the "response" indicates a response sent from the RX610 to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by the RX610 becomes 00h.

### (1) Data Mat Checksum

In response to a data mat checksum command sent from the host, the RX610 sums the data mat data in byte units and returns the result (checksum).

| | | | | |
|---|---|---|---|---|
| Command | 61h | | | |
| Response | 71h | Size | Mat checksum | SUM |

[Legend]

Size (1 byte): Number of bytes in the mat checksum field (fixed at 4)

Mat checksum (4 bytes): Checksum of the data mat data

SUM (4 bytes): Checksum (for the response data)



Downloaded from Arrow.com.

### (2) Data Mat Blank Check

In response to a data mat blank check command sent from the host, the RX610 checks whether the data mat is completely erased. When the data mat is completely erased, the RX610 returns a response (06h). If the data mat has an unerased area, the RX610 returns an error response (sends E2h and 52h in that order).

| | | |
|---|---|---|
| Command | 62h | |
| Response | 06h | |
| Error response | E2h | 52h |

## 27.9 Usage Notes

### (1) Protection of the Data Mat Immediately after a Reset

As the initial values of DFLRE and DFLWE are 0000h, programming, erasure, and reading of the data mat are disabled immediately after a reset. To read data from the data mat, set DFLRE appropriately before accessing the data mat. To program or erase the data mat, set DFLWE appropriately before issuing an FCU command for programming or erasure. If an attempt is made to read, program, or erase the data mat without setting the registers, the FCU detects the error and enters the command-locked state.

### (2) Other Points to Note

The other points to note are the same as for the ROM. See section 26.13, Usage Notes. The data flash memory also has a blank checking operation, so interpret references to programming or erasure as meaning programming, erasure, or blank checking.



Downloaded from Arrow.com.

# 28. Boundary Scan

The RX610 Group has boundary scan function, and this function is supported only in the 176-pin LFBGA version.

The boundary scan is a serial I/O interface based on the JTAG (Joint Test Action Group, IEEE Std.1149.1 and IEEE Standard Test Access Port and Boundary-Scan Architecture).

## 28.1 Features

Table 28.1 lists the specifications of boundary scan.

Figure 28.1 shows the block diagram of the boundary scan function.

**Table 28.1 Specifications of Boundary Scan**

| Item | Description |
|---|---|
| Boundary scan enabled/disabled | Boundary can is enabled when the EMLE pin is driven low and the BSCANP pin is driven high. |
| Dedicated boundary scan pins | Pins P02, P03, P04, P05, and WDTOVF# are dedicated for JTAG, when boundary scan function is enabled. |
| Six test modes | • BYPASS mode<br>• EXTEST mode<br>• SAMPLE/PRELOAD mode<br>• CLAMP mode<br>• HIGHZ mode<br>• IDCODE mode |

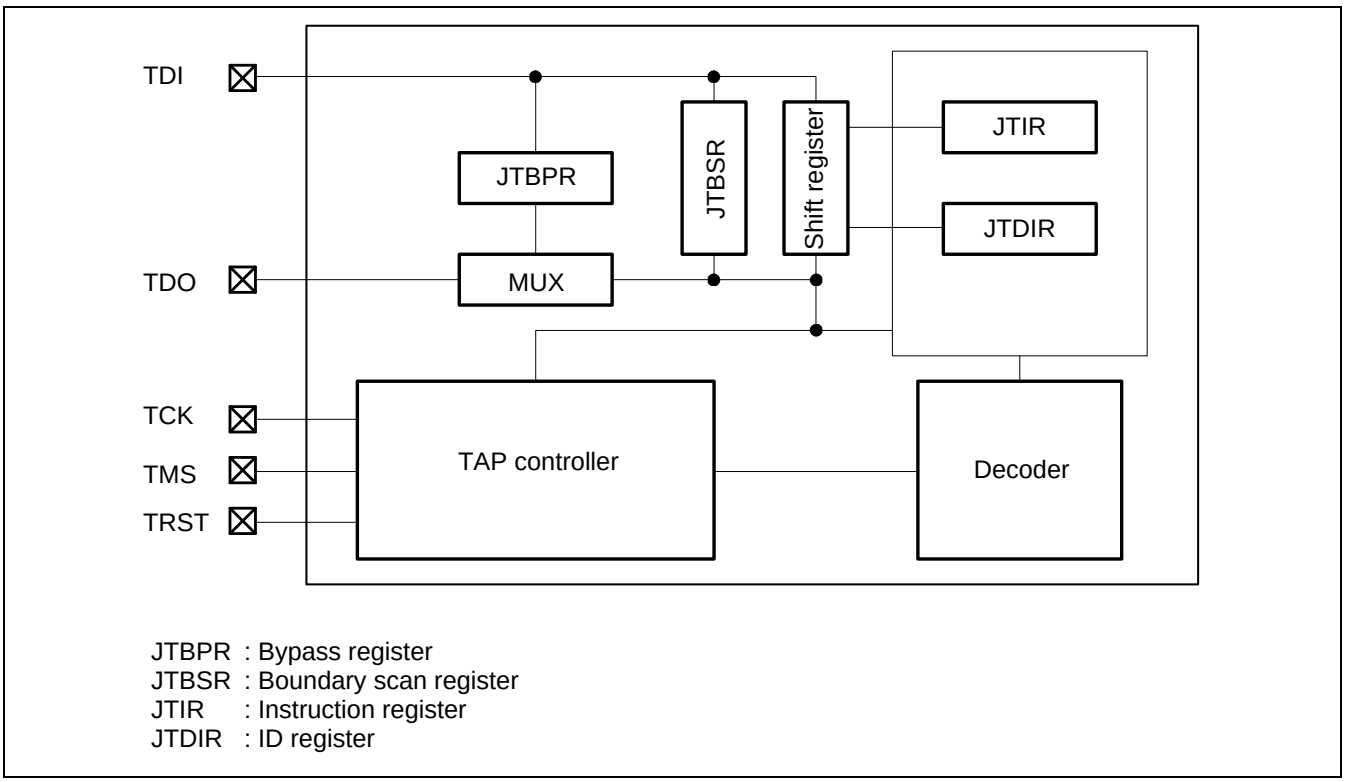



**Figure 28.1 Block Diagram of Boundary Scan Function**



Downloaded from Arrow.com.

Table 28.2 shows the I/O pins used in the boundary scan function.

**Table 28.2 Pin Configuration**

| Pin Name | I/O | Description |
|---|---|---|
| TCK | Input | Test clock input pin<br>Clock signal for boundary scan. Input the clock the duty cycle of which is 50 percent when boundary scan function is used. |
| TMS | Input | Test mode select pin |
| TDI | Input | Test data input pin |
| TDO | Output | Test data output pin |
| TRST | Input | Test reset input pin |

## 28.2 Register Descriptions

Table 28.3 lists the boundary scan registers.

| Register Name | Symbol | Value after Reset | Address | Access Size |
|---|---|---|---|---|
| Instruction register | JTIR | 4h | — | 4 |
| Bypass register | JTBPR | Undefinded | — | 1 |
| Boundary scan register | JTBSR | Undefinded | — | — |
| IDCODE register | JTIDR | 0809 9447h | — | 32 |

Instructions can be input to the instruction register (JTIR) via the test data input pin (TDI) by serial transfer.

The bypass register (JTBPR), which is a 1-bit register, is connected between the TDI and TDO pins in BYPASS mode.

The boundary scan register (JTBSR), which is a JTBSR-bit register (see table 28.6), is connected between the TDI and TDO pins when test data are being shifted in.

None of the registers is accessible from the CPU.

Table 28.4 shows the availability of serial transfer for the registers.

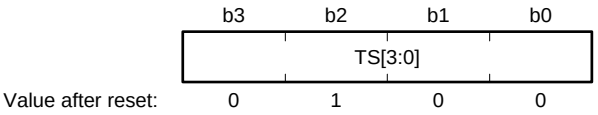
**Table 28.4 Serial Transfers for Registers**

| Register Abbreviation | Serial Input | Serial Output |
|---|---|---|
| JTIR | Available | Available |
| JTBPR | Available | Available |
| JTBSR | Available | Available |
| JTID | Available | Available |



Downloaded from Arrow.com.

### 28.2.1 Instruction Register (JTIR)




| Bit | Symbol | Bit Name | Description | R/W |
|---|---|---|---|---|
| b3 to b0 | TS[3:0] | Test Bit Set | The command configuration is as shown in table 28.5. | — |

**Table 28.5 Command Configuration**

| TS3 | TS2 | TS1 | TS0 | Instruction |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | EXTEST |
| 0 | 0 | 0 | 1 | SAMPLE/PRELOAD |
| 0 | 1 | 0 | 0 | IDCODE (initial value) |
| 0 | 1 | 1 | 0 | CLAMP |
| 0 | 1 | 1 | 1 | HIGHZ |
| 1 | 1 | 1 | 1 | BYPASS |
| Other than above | | | | Reserved |

JTIR is a 4-bit register.

JTAG instructions can be transferred to JTIR by serial input from the TDI pin.

JTIR is initialized when the TRST signal is low level, when the TAP controller is in the Test-Logic-Reset state.

### 28.2.2 Bypass Register (JTBPR)

JTBPR is a 1-bit register and is connected between the TDI and TDO pins when JTIR is set to BYPASS mode.

JTBPR cannot be read from or written to by the CPU.

### 28.2.3 Boundary Scan Register (JTBSR)

JTBSR is a shift register to control the external input and output pins of this LSI and is distributed across the pads.

The EXTEST, SAMPLE/PRELOAD, CLAMP, and HIGHZ instructions are issued to apply JTBSR in boundary-scan testing conformant to the JTAG standard.

Table 28.6 shows the correspondence between the JTBSR bits and the pins of this LSI.

The value after reset is undefined.



Downloaded from Arrow.com.

### Table 28.6 Relationship between Pins and JTBSR Bits

| | | From TDI | |
|---|---|---|---|
| **176-Pin LFBGA** | **Pin Name** | **Input/Output** | **Bit Name** |
| B1 | P67 | Input | 411 |
| | | Output enable | 410 |
| | | Output | 409 |
| C2 | P66 | Input | 408 |
| | | Output enable | 407 |
| | | Output | 406 |
| D2 | P01 | Input | 405 |
| | | Output enable | 404 |
| | | Output | 403 |
| E4 | P00 | Input | 402 |
| | | Output enable | 401 |
| | | Output | 400 |
| D1 | P65 | Input | 399 |
| | | Output enable | 398 |
| | | Output | 397 |
| F4 | MDE | Input | 396 |
| F1 | MD1 | Input | 395 |
| F2 | MD0 | Input | 394 |
| G4 | P86 | Input | 393 |
| | | Output enable | 392 |
| | | Output | 391 |
| G3 | P85 | Input | 390 |
| | | Output enable | 389 |
| | | Output | 388 |
| H2 | NMI | Input | 387 |
| J4 | P34 | Input | 386 |
| | | Output enable | 385 |
| | | Output | 384 |
| J3 | PF6 | Input | 383 |
| | | Output enable | 382 |
| | | Output | 381 |
| J1 | PF5 | Input | 380 |
| | | Output enable | 379 |
| | | Output | 378 |
| J2 | PF4 | Input | 377 |
| | | Output enable | 376 |
| | | Output | 375 |
| K4 | P33 | Input | 374 |
| | | Output enable | 373 |
| | | Output | 372 |
| K3 | P32 | Input | 371 |
| | | Output enable | 370 |
| | | Output | 369 |
| K1 | P31 | Input | 368 |
| | | Output enable | 367 |
| | | Output | 366 |
| K2 | P30 | Input | 365 |
| | | Output enable | 364 |
| | | Output | 363 |
| L3 | PF3 | Input | 362 |
| | | Output enable | 361 |
| | | Output | 360 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| L1 | PF2 | Input | 359 |
| | | Output enable | 358 |
| | | Output | 357 |
| L2 | PF1 | Input | 356 |
| | | Output enable | 355 |
| | | Output | 354 |
| L4 | PF0 | Input | 353 |
| | | Output enable | 352 |
| | | Output | 351 |
| M1 | P27 | Input | 350 |
| | | Output enable | 349 |
| | | Output | 348 |
| M2 | P26 | Input | 347 |
| | | Output enable | 346 |
| | | Output | 345 |
| N1 | P25 | Input | 344 |
| | | Output enable | 343 |
| | | Output | 342 |
| N2 | P24 | Input | 341 |
| | | Output enable | 340 |
| | | Output | 339 |
| P1 | P23 | Input | 338 |
| | | Output enable | 337 |
| | | Output | 336 |
| P2 | P22 | Input | 335 |
| | | Output enable | 334 |
| | | Output | 333 |
| R1 | P21 | Input | 332 |
| | | Output enable | 331 |
| | | Output | 330 |
| N3 | P20 | Input | 329 |
| | | Output enable | 328 |
| | | Output | 327 |
| R2 | P17 | Input | 326 |
| | | Output enable | 325 |
| | | Output | 324 |
| N4 | P16 | Input | 323 |
| | | Output enable | 322 |
| | | Output | 321 |
| P4 | P15 | Input | 320 |
| | | Output enable | 319 |
| | | Output | 318 |
| M5 | P14 | Input | 317 |
| | | Output enable | 316 |
| | | Output | 315 |
| R4 | P13 | Input | 314 |
| | | Output enable | 313 |
| | | Output | 312 |
| N5 | P12 | Input | 311 |
| | | Output enable | 310 |
| | | Output | 309 |
| P5 | P11 | Input | 308 |
| | | Output enable | 307 |
| | | Output | 306 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| R5 | P10 | Input | 305 |
| | | Output enable | 304 |
| | | Output | 303 |
| M6 | P37 | Input | 302 |
| | | Output enable | 301 |
| | | Output | 300 |
| N6 | P36 | Input | 299 |
| | | Output enable | 298 |
| | | Output | 297 |
| R6 | P35 | Input | 296 |
| | | Output enable | 295 |
| | | Output | 294 |
| P6 | P84 | Input | 293 |
| | | Output enable | 292 |
| | | Output | 291 |
| M7 | P57 | Input | 290 |
| | | Output enable | 289 |
| | | Output | 288 |
| N7 | P56 | Input | 287 |
| | | Output enable | 286 |
| | | Output | 285 |
| R7 | P55 | Input | 284 |
| | | Output enable | 283 |
| | | Output | 282 |
| P7 | P54 | Input | 281 |
| | | Output enable | 280 |
| | | Output | 279 |
| M8 | P83 | Input | 278 |
| | | Output enable | 277 |
| | | Output | 276 |
| R8 | P82 | Input | 275 |
| | | Output enable | 274 |
| | | Output | 273 |
| M9 | P81 | Input | 272 |
| | | Output enable | 271 |
| | | Output | 270 |
| N9 | P80 | Input | 269 |
| | | Output enable | 268 |
| | | Output | 267 |
| R9 | P53 | Input | 266 |
| | | Output enable | 265 |
| | | Output | 264 |
| P9 | P52 | Input | 263 |
| | | Output enable | 262 |
| | | Output | 261 |
| M10 | P51 | Input | 260 |
| | | Output enable | 259 |
| | | Output | 258 |
| N10 | P50 | Input | 257 |
| | | Output enable | 256 |
| | | Output | 255 |
| R10 | PH7 | Input | 254 |
| | | Output enable | 253 |
| | | Output | 252 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| P10 | PH6 | Input | 251 |
| | | Output enable | 250 |
| | | Output | 249 |
| R11 | PH5 | Input | 248 |
| | | Output enable | 247 |
| | | Output | 246 |
| M11 | PH4 | Input | 245 |
| | | Output enable | 244 |
| | | Output | 243 |
| R12 | PH3 | Input | 242 |
| | | Output enable | 241 |
| | | Output | 240 |
| P12 | P77 | Input | 239 |
| | | Output enable | 238 |
| | | Output | 237 |
| N12 | P76 | Input | 236 |
| | | Output enable | 235 |
| | | Output | 234 |
| R13 | P75 | Input | 233 |
| | | Output enable | 232 |
| | | Output | 231 |
| M12 | PC7 | Input | 230 |
| | | Output enable | 229 |
| | | Output | 228 |
| P13 | PC6 | Input | 227 |
| | | Output enable | 226 |
| | | Output | 225 |
| R14 | PC5 | Input | 224 |
| | | Output enable | 223 |
| | | Output | 222 |
| P14 | PC4 | Input | 221 |
| | | Output enable | 220 |
| | | Output | 219 |
| R15 | PC3 | Input | 218 |
| | | Output enable | 217 |
| | | Output | 216 |
| N13 | PH2 | Input | 215 |
| | | Output enable | 214 |
| | | Output | 213 |
| N14 | PC2 | Input | 212 |
| | | Output enable | 211 |
| | | Output | 210 |
| N15 | PC1 | Input | 209 |
| | | Output enable | 208 |
| | | Output | 207 |
| M14 | PC0 | Input | 206 |
| | | Output enable | 205 |
| | | Output | 204 |
| L12 | PB7 | Input | 203 |
| | | Output enable | 202 |
| | | Output | 201 |
| M15 | PB6 | Input | 200 |
| | | Output enable | 199 |
| | | Output | 198 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| L13 | PB5 | Input | 197 |
| | | Output enable | 196 |
| | | Output | 195 |
| L14 | PB4 | Input | 194 |
| | | Output enable | 193 |
| | | Output | 192 |
| L15 | PB3 | Input | 191 |
| | | Output enable | 190 |
| | | Output | 189 |
| K12 | PB2 | Input | 188 |
| | | Output enable | 187 |
| | | Output | 186 |
| K13 | PB1 | Input | 185 |
| | | Output enable | 184 |
| | | Output | 183 |
| K15 | P74 | Input | 182 |
| | | Output enable | 181 |
| | | Output | 180 |
| K14 | P73 | Input | 179 |
| | | Output enable | 178 |
| | | Output | 177 |
| J12 | P72 | Input | 176 |
| | | Output enable | 175 |
| | | Output | 174 |
| J13 | P71 | Input | 173 |
| | | Output enable | 172 |
| | | Output | 171 |
| J15 | P70 | Input | 170 |
| | | Output enable | 169 |
| | | Output | 168 |
| H12 | PB0 | Input | 167 |
| | | Output enable | 166 |
| | | Output | 165 |
| H15 | PH1 | Input | 164 |
| | | Output enable | 163 |
| | | Output | 162 |
| H14 | PH0 | Input | 161 |
| | | Output enable | 160 |
| | | Output | 159 |
| G12 | PA7 | Input | 158 |
| | | Output enable | 157 |
| | | Output | 156 |
| G13 | PA6 | Input | 155 |
| | | Output enable | 154 |
| | | Output | 153 |
| G15 | PA5 | Input | 152 |
| | | Output enable | 151 |
| | | Output | 150 |
| G14 | PA4 | Input | 149 |
| | | Output enable | 148 |
| | | Output | 147 |
| F12 | PA3 | Input | 146 |
| | | Output enable | 145 |
| | | Output | 144 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| F13 | PA2 | Input | 143 |
| | | Output enable | 142 |
| | | Output | 141 |
| F15 | PA1 | Input | 140 |
| | | Output enable | 139 |
| | | Output | 138 |
| F14 | PA0 | Input | 137 |
| | | Output enable | 136 |
| | | Output | 135 |
| E13 | PG7 | Input | 134 |
| | | Output enable | 133 |
| | | Output | 132 |
| E14 | PG6 | Input | 131 |
| | | Output enable | 130 |
| | | Output | 129 |
| D15 | PG5 | Input | 128 |
| | | Output enable | 127 |
| | | Output | 126 |
| D14 | PE7 | Input | 125 |
| | | Output enable | 124 |
| | | Output | 123 |
| D13 | PE6 | Input | 122 |
| | | Output enable | 121 |
| | | Output | 120 |
| C15 | PE5 | Input | 119 |
| | | Output enable | 118 |
| | | Output | 117 |
| D12 | PE4 | Input | 116 |
| | | Output enable | 115 |
| | | Output | 114 |
| C14 | PE3 | Input | 113 |
| | | Output enable | 112 |
| | | Output | 111 |
| B15 | PE2 | Input | 110 |
| | | Output enable | 109 |
| | | Output | 108 |
| B14 | PE1 | Input | 107 |
| | | Output enable | 106 |
| | | Output | 105 |
| A15 | PE0 | Input | 104 |
| | | Output enable | 103 |
| | | Output | 102 |
| C13 | PD7 | Input | 101 |
| | | Output enable | 100 |
| | | Output | 99 |
| A14 | PD6 | Input | 98 |
| | | Output enable | 97 |
| | | Output | 96 |
| B13 | PD5 | Input | 95 |
| | | Output enable | 94 |
| | | Output | 93 |
| A13 | PD4 | Input | 92 |
| | | Output enable | 91 |
| | | Output | 90 |



Downloaded from Arrow.com.

**From TDI**

| 176-Pin LFBGA | Pin Name | Input/Output | Bit Name |
|---|---|---|---|
| D11 | P64 | Input | 89 |
| | | Output enable | 88 |
| | | Output | 87 |
| A12 | P63 | Input | 86 |
| | | Output enable | 85 |
| | | Output | 84 |
| C11 | P62 | Input | 83 |
| | | Output enable | 82 |
| | | Output | 81 |
| B11 | P61 | Input | 80 |
| | | Output enable | 79 |
| | | Output | 78 |
| A11 | P60 | Input | 77 |
| | | Output enable | 76 |
| | | Output | 75 |
| D10 | PD3 | Input | 74 |
| | | Output enable | 73 |
| | | Output | 72 |
| C10 | PD2 | Input | 71 |
| | | Output enable | 70 |
| | | Output | 69 |
| A10 | PD1 | Input | 68 |
| | | Output enable | 67 |
| | | Output | 66 |
| B10 | PD0 | Input | 65 |
| | | Output enable | 64 |
| | | Output | 63 |
| D9 | PG4 | Input | 62 |
| | | Output enable | 61 |
| | | Output | 60 |
| C9 | PG3 | Input | 59 |
| | | Output enable | 58 |
| | | Output | 57 |
| A9 | PG2 | Input | 56 |
| | | Output enable | 55 |
| | | Output | 54 |
| B9 | PG1 | Input | 53 |
| | | Output enable | 52 |
| | | Output | 51 |
| D8 | PG0 | Input | 50 |
| | | Output enable | 49 |
| | | Output | 48 |
| A8 | P97 | Input | 47 |
| | | Output enable | 46 |
| | | Output | 45 |
| B8 | P96 | Input | 44 |
| | | Output enable | 43 |
| | | Output | 42 |
| D7 | P95 | Input | 41 |
| | | Output enable | 40 |
| | | Output | 39 |
| C7 | P94 | Input | 38 |
| | | Output enable | 37 |
| | | Output | 36 |



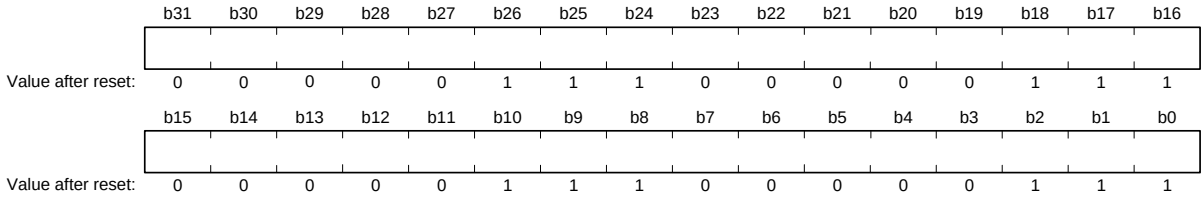
Downloaded from Arrow.com.

| From TDI | | | |
|---|---|---|---|
| **176-Pin LFBGA** | **Pin Name** | **Input/Output** | **Bit Name** |
| A7 | P93 | Input | 35 |
| | | Output enable | 34 |
| | | Output | 33 |
| B7 | P92 | Input | 32 |
| | | Output enable | 31 |
| | | Output | 30 |
| D6 | P91 | Input | 29 |
| | | Output enable | 28 |
| | | Output | 27 |
| A6 | P90 | Input | 26 |
| | | Output enable | 25 |
| | | Output | 24 |
| C5 | P47 | Input | 23 |
| | | Output enable | 22 |
| | | Output | 21 |
| A5 | P46 | Input | 20 |
| | | Output enable | 19 |
| | | Output | 18 |
| B5 | P45 | Input | 17 |
| | | Output enable | 16 |
| | | Output | 15 |
| D5 | P44 | Input | 14 |
| | | Output enable | 13 |
| | | Output | 12 |
| A4 | P43 | Input | 11 |
| | | Output enable | 10 |
| | | Output | 9 |
| B4 | P42 | Input | 8 |
| | | Output enable | 7 |
| | | Output | 6 |
| C4 | P41 | Input | 5 |
| | | Output enable | 4 |
| | | Output | 3 |
| D4 | P40 | Input | 2 |
| | | Output enable | 1 |
| | | Output | 0 |



Downloaded from Arrow.com.

### 28.2.4 IDCODE Register (JTID)




| Bit | Description | R/W |
|---|---|---|
| b31 to b0 | JTID is a register with the fixed value that indicates the device IDCODE. | — |

JTID is a 32-bit register.

JTID data is output from the TDO pin when the IDCODE instruction has been executed.



Downloaded from Arrow.com.

## 28.3 Operations

The boundary scan functionality is valid when the EMLE pin is driven low and the BSCANP pin is driven high.

### 28.3.1 TAP Controller

Figure 28.2 shows the state transition diagram of the TAP controller.

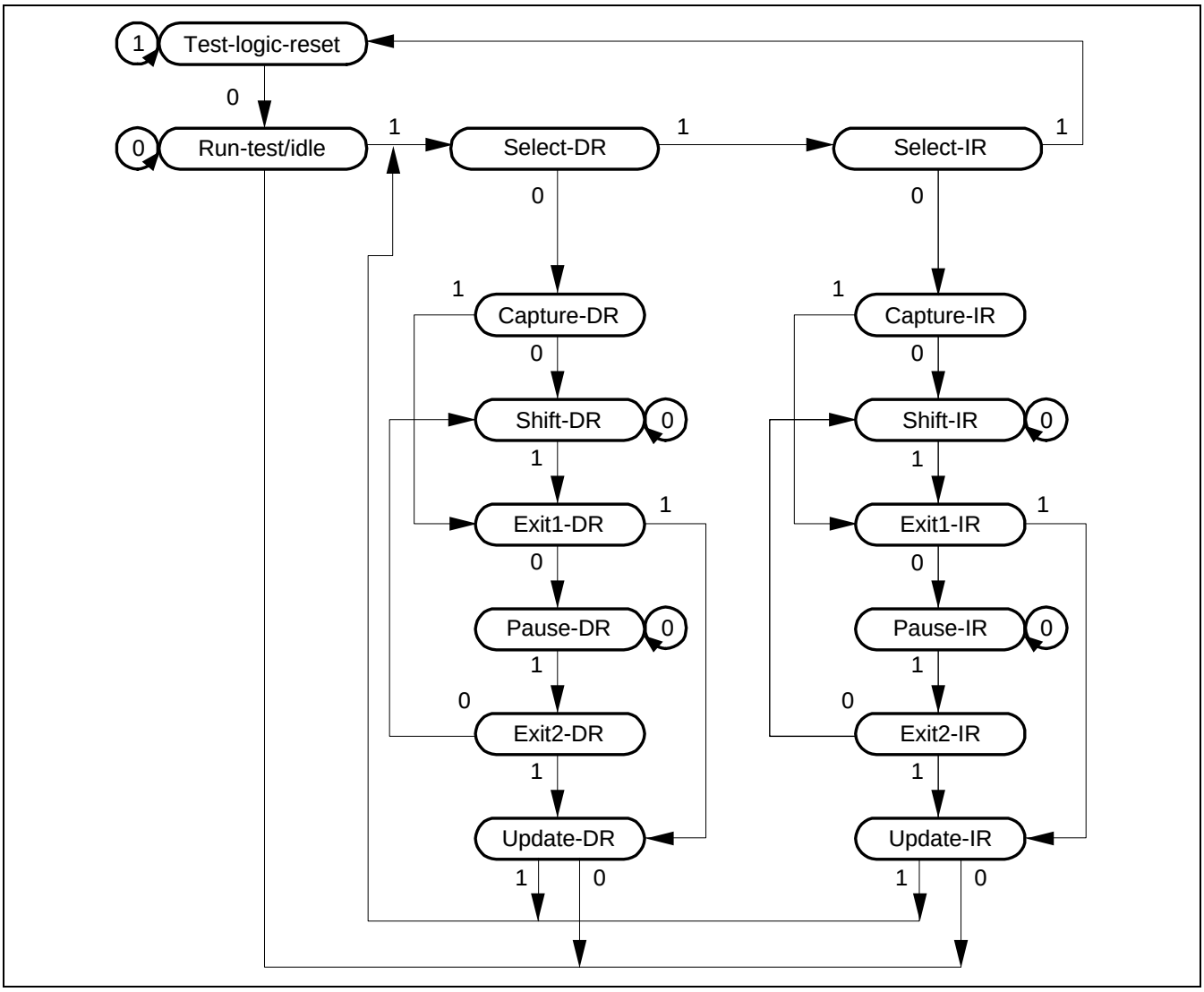



**Figure 28.2 State Transition of TAP Controller**



Downloaded from Arrow.com.

### 28.3.2 List of Commands

#### (1) BYPASS [Instruction Code: 1111b]

The BYPASS instruction is an instruction that drives the bypass register (JTBPR). This instruction shortens the shift path, facilitating the transfer of serial data to other LSIs on a printed-circuit board at higher speeds. While this instruction is being executed, the test circuit has no effect on the system circuits.

The bypass register (JTBPR) is connected between the TDI and TDO pins. Bypass operation is initiated from shift-DR operation. The TDO is low in the first clock cycle in the shift-DR state; in the subsequent clock cycles, the TDI signal is output on the TDO pin.

#### (2) EXTEST [Instruction Code: 0000b]

The EXTEST instruction is used to test external circuits when this LSI is installed on the printed circuit board. If this instruction is executed, output pins are used to output test data (specified by the SAMPLE/PRELOAD instruction) from the boundary scan register to the print circuit board, and input pins are used to input test result.

#### (3) SAMPLE/PRELOAD [Instruction Code: 0001b]

The SAMPLE/PRELOAD instruction is used to input data from the LSI internal circuits to the boundary scan register, output data from scan path, and reload the data to the scan path. While this instruction is executed, input signals are directly input to the LSI and output signals are also directly output to the external circuits. The LSI system circuit is not affected by this instruction.

In SAMPLE operation, the boundary scan register latches the snap shot of data transferred from input pins to internal circuit or data transferred from internal circuit to output pins. The latched data is read from the scan path. The scan register latches the snap data at the rising edge of the TCK in Capture-DR state. The scan register latches snap shot without affecting the LSI normal operation.

In PRELOAD operation, initial value is written from the scan path to the parallel output latch of the boundary scan register prior to the EXTEST instruction execution. If the EXTEST is executed without executing this PRELOAD operation, undefined values are output from the beginning to the end (transfer to the output latch) of the EXTEST sequence. (In EXTEST instruction, output parallel latches are always output to the output pins.)

#### (4) IDCODE [Instruction Code: 0100b]

When the IDCODE instruction is selected, IDCODE register value is output to the TDO in Shift-DR state of the TAP controller. In this case, IDCODE register value is output from the LSB. During this instruction execution, test circuit does not affect the system circuit. INSTR is initialized by the IDCODE instruction in Test-Logic-Reset state of the TAP controller.



Downloaded from Arrow.com.

### (5) CLAMP [Instruction Code: 0110b]

When the CLAMP instruction is selected, output pins output the boundary scan register value which was specified by the SAMPLE/PRELOAD instruction in advance. While the CLAMP instruction is selected, the status of boundary scan register is maintained regardless of the TAP controller state.

BYPASS is connected between TDI and TDO, the same operation as BYPASS instruction can be achieved.

### (6) HIGHZ [Instruction Code: 0111b]

When the HIGHZ instruction is selected, all output pins enter high-impedance state. While the HIGHZ instruction is selected, the status of boundary scan register is maintained regardless of the state of the TAP controller.

BYPASS is connected between TDI and TDO pins, leading to the same operation as when the BYPASS instruction has been selected.



Downloaded from Arrow.com.

## 28.4 Usage Notes

1. In serial transfer, data are input or output in LSB order (see figure 28.3).

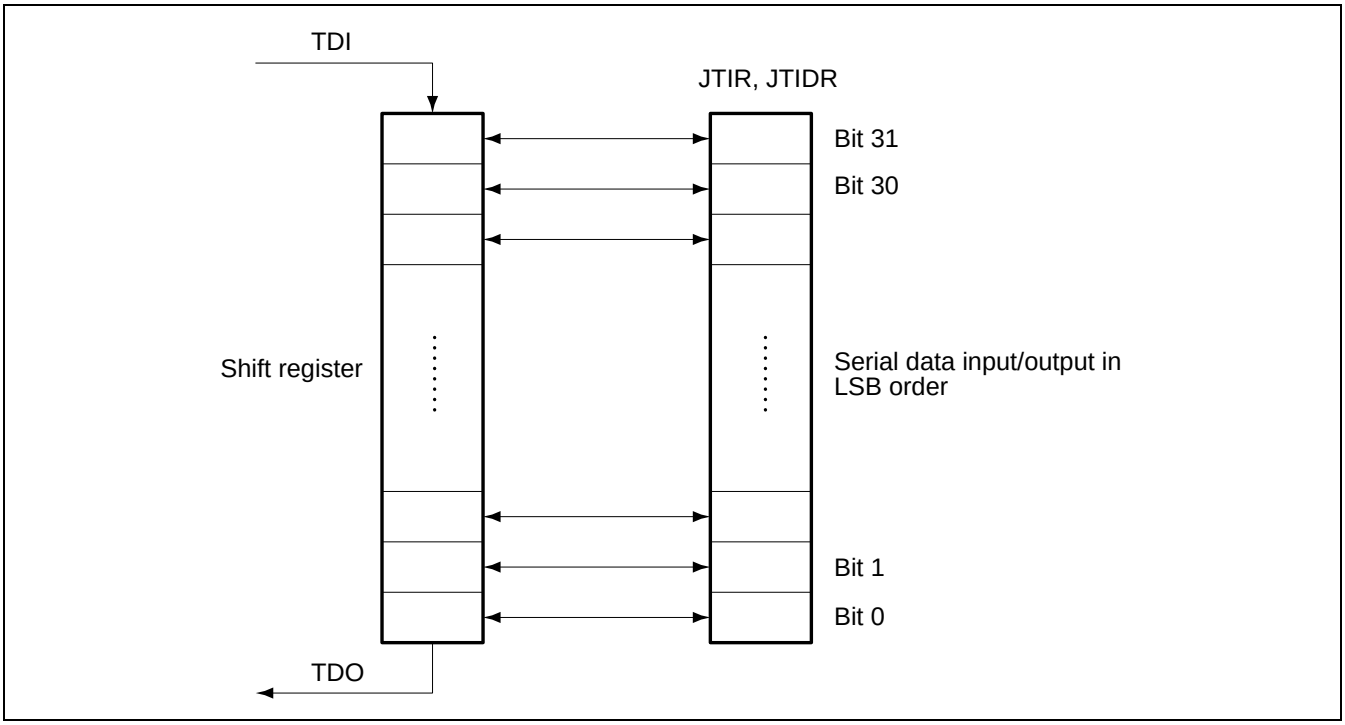



**Figure 28.3 Serial Data Input/Output**

2. Pins of the boundary scan (TCK, TDI, TMS, and TRST#) have to be pulled up by pull-up resistors.
3. Power supply pins (VCC, VCL, VSS, AVCC, AVSS, VREFH, VREF, PLLVCC, and PLLVSS) cannot be boundary-scanned.
4. Clock pins (EXTAL and XTAL) cannot be boundary-scanned.
5. Reset signal (RES#) cannot be boundary-scanned.
6. Boundary scan pins (TCK, TMS, TRST#, TDI, and TDO) cannot be boundary-scanned.
7. The boundary scan function is not available when this LSI are in the following states.
   (1) Reset state
   (2) Hardware standby mode, software standby mode, and deep software standby mode
8. While the pin with the open-drain function is enabled, set the output scan register to 1 and the output enable register to 1 by the boundary scan function. In this case, executing any of EXTEST, CLAMP, and SAMPLE/PRELOAD instructions makes the pin high-output instead of high-impedance.
9. Figure 28.4 (1) shows the pin configuration of pins P14 to P17. When the boundary scan function is used with pins P14 to P17 to be used as RIIC pins (SDA0, SDA1, SCL0, SCL1), the conflict with the open-drain output or sneak current might be generated.
10. Figure 28.4 (2) shows the pin configuration of pins P40 to P47, and P90 to P97. When the boundary scan function is used with pins P40 to P47, and P90 to P97 to be used as AD input pins (AN0 to AN7, and AN8 to AN15), the conflict with the AD input or sneak current might be generated.



Downloaded from Arrow.com.

11. Figure 28.4 (3) shows the pin configuration of pins P66 and P67. When the boundary scan function is used with pins P66 and P67 to be used as DA output pins (DA0 and DA1), the conflict with the DA output or sneak current might be generated.

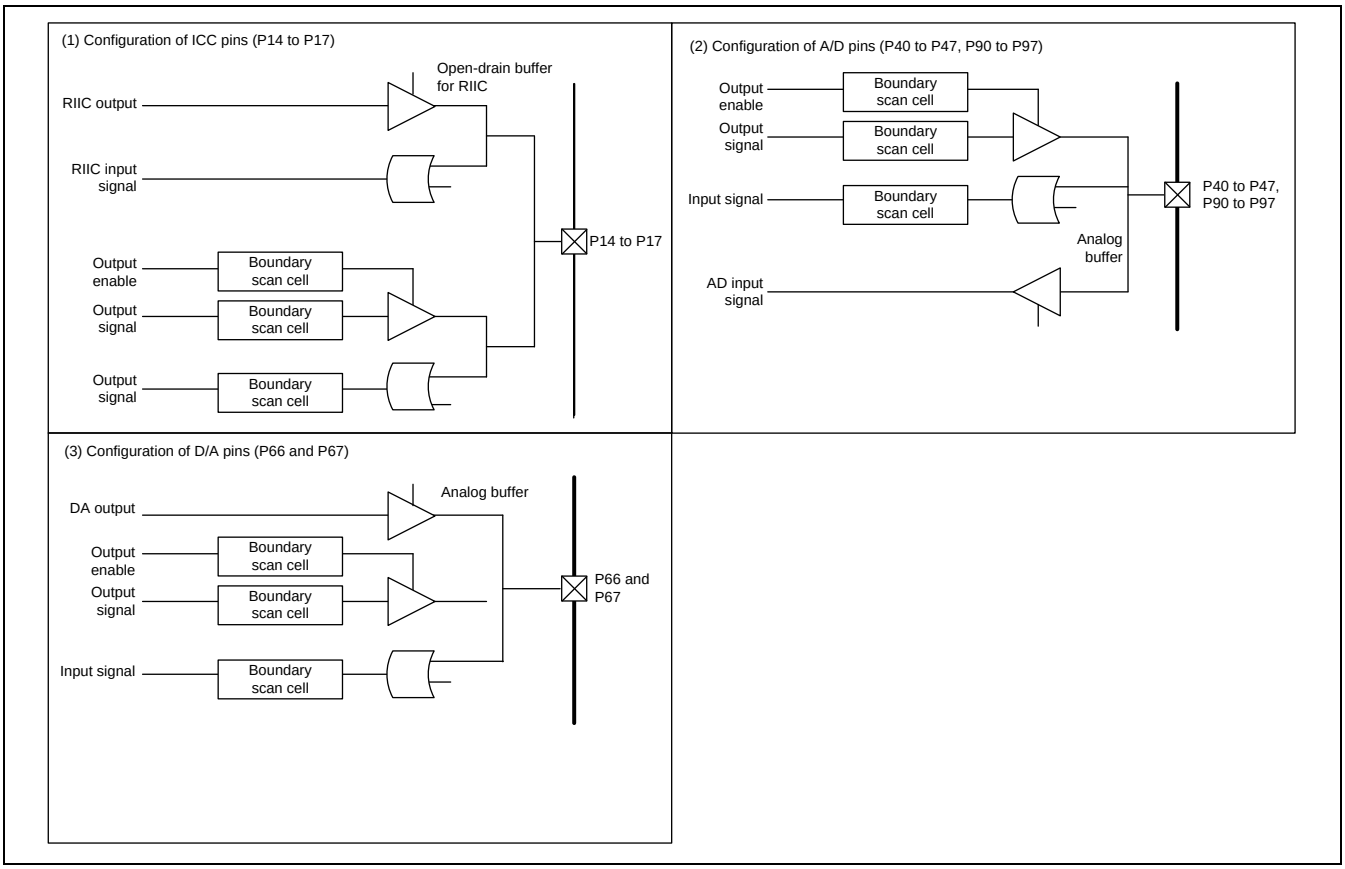



**Figure 28.4 Pin Configuration**



Downloaded from Arrow.com.

# 29. Electrical Characteristics

## 29.1 Absolute Maximum Ratings

**Table 29.1 Absolute Maximum Ratings**

| Item | Symbol | Value | Unit |
|---|---|---|---|
| Power supply voltage | $V_{CC}$, $PLLV_{CC}$ | -0.3 to +4.6 | V |
| Input voltage (except for ports 0, 14 to 17) | $V_{in}$ | -0.3 to $V_{CC}$ +0.3 | V |
| Input voltage (ports 0, 14 to 17*1) | $V_{in}$ | -0.3 to +6.5 | V |
| Reference power supply voltage | $V_{REFH}$ | -0.3 to $V_{CC}$ +0.3 | V |
| Analog power supply voltage | $AV_{CC}$*2 | -0.3 to +4.6 | V |
| Analog input voltage | $V_{AN}$ | -0.3 to $V_{CC}$ +0.3 | V |
| Operating temperature | $T_{opr}$ | Regular specifications: -20 to +85 | C |
| | | Wide-range specifications: -40 to +85 | |
| Storage temperature | $T_{stg}$ | -55 to +125 | C |

Caution: Permanent damage to the LSI may result if absolute maximum ratings are exceeded.

Notes: 1. Ports 0, and 14 to 17 are 5 V tolerant.

2. Connect $AV_{CC}$ to $V_{CC.}$ When neither the A/D converter nor the D/A converter is in use, do not leave the $AV_{SS}$, $V_{REFH}$, and $V_{REFL}$ pins open. Connect the $AV_{CC}$ and $V_{REFH}$ pins to $V_{CC}$, and the $AV_{SS}$ and $V_{REFL}$ pins to $V_{SS}$, respectively.



Downloaded from Arrow.com.

## 29.2 DC Characteristics

**Table 29.2 DC Characteristics**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Schmitt trigger input voltage | IRQ input pin*1<br>TPU input pin*1<br>TMR input pin*1<br>SCI input pin*1<br>ADTRG# input pin*1<br>RES#, NMI | $V_{IH}$ | $V_{CC}$ x 0.8 | — | $V_{CC}$ + 0.3 | V | |
| | | $V_{IL}$ | -0.3 | — | $V_{CC}$ x 0.2 | | |
| | | $\Delta V_T$ | $V_{CC}$ x 0.06 | — | — | | |
| | RIIC input pin | $V_{IH}$ | $V_{CC}$ x 0.7 | — | 5.8 | | |
| | | $V_{IL}$ | -0.3 | — | $V_{CC}$ x 0.3 | | |
| | | $\Delta V_T$ | $V_{CC}$ x 0.05 | — | — | | |
| | Ports 0, 14 to 17*2 | $V_{IH}$ | $V_{CC}$ x 0.8 | — | 5.8 | | |
| | | $V_{IL}$ | -0.3 | — | $V_{CC}$ x 0.2 | | |
| | Ports 10 to 13,<br>ports 2 to E (144-pin LQFP)<br>ports 2 to H (176-pin LFBGA)<br>Other input pins | $V_{IH}$ | $V_{CC}$ x 0.8 | — | $V_{CC}$ + 0.3 | | |
| | | $V_{IL}$ | -0.3 | — | $V_{CC}$ x 0.2 | | |
| Input high voltage (except Schmitt trigger input pin) | MD pin, EMLE | $V_{IH}$ | $V_{CC}$ x 0.9 | — | $V_{CC}$ + 0.3 | V | |
| | EXTAL | | $V_{CC}$ x 0.8 | — | $V_{CC}$ + 0.3 | | |
| | D0 to D15 | | $V_{CC}$ x 0.7 | — | $V_{CC}$ + 0.3 | | |
| Input low voltage (except Schmitt trigger input pin) | MD pin, EMLE | $V_{IL}$ | -0.3 | — | $V_{CC}$ x 0.1 | V | |
| | EXTAL | | -0.3 | — | $V_{CC}$ x 0.2 | | |
| | D0 to D15 | | -0.3 | — | $V_{CC}$ x 0.3 | | |
| Output high voltage | All output pins | $V_{OH}$ | $V_{CC}$-0.5 | — | — | V | $I_{OH}$ = -1 mA |
| Output low voltage | All output pins (except for RIIC pins) | $V_{OL}$ | — | — | 0.5 | V | $I_{OL}$ = 1.0 mA |
| | RIIC pins | | — | — | 0.4 | | $I_{OL}$ = 3.0 mA |
| | | | — | — | 0.6 | | $I_{OL}$ = 6.0 mA |
| | RIIC pins (only P14 and P15 in channel 1) | | — | — | 0.4 | | $I_{OL}$ = 15 mA (ICFER.FMPE = 1) |
| | | | — | 0.4 | — | | $I_{OL}$ = 20 mA (ICFER.FMPE = 1) |
| Input leakage current | RES#, MD pin, EMLE, NMI | $\lvert I_{in} \rvert$ | — | — | 1.0 | µA | $V_{in}$ = 0 V, $V_{CC}$ |
| Three-state leakage current (off state) | Ports 10 to 13,<br>ports 2 to E (144-pin LQFP)<br>ports 2 to H (176-pin LFBGA) | $\lvert I_{TSI} \rvert$ | — | — | 1.0 | µA | $V_{in}$ = 0 V, $V_{CC}$ |
| | Port 0, ports 14 to 17 | | — | — | 5.0 | | |



Downloaded from Arrow.com.

| Item | | | | Symbol | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|---|---|
| Input pull-up resistor current | Ports A to E | | | $-I_p$ | 10 | — | 300 | µA | $V_{CC}$ = 3.0 to 3.6 V, $V_{in}$ = 0 V |
| Input capacitance | All input pins (except port 0, ports 14 to 17) | | | $C_{in}$ | — | — | 15 | pF | $V_{in}$ = 0 V, f = 1 MHz, $T_a$= 25 C |
| | Port 0, ports 14 to 17 | | | | — | — | 30 | | |
| Supply current*3 | In operation | Max.*4 | | $I_{CC}$*5 | — | — | 100 | mA | ICLK = 100 MHz PCLK = 50 MHz BCLK = 25 MHz |
| | | Normal*6 | | | — | 35 | — | | |
| | | Increased by BGO operation*7 | | | — | 15 | — | | |
| | Sleep | | | | — | 18 | 52 | | |
| | All-module-clock-stop mode*8 | | | | — | 14 | 28 | | |
| | Standby mode | Software standby mode | | | — | 0.08 | 3.0 | | |
| | | Deep software standby mode | RAM retained | | — | 15 | 200 | µA | |
| | | | RAM power supply halted | | — | 0.9 | 26 | | |
| Analog power supply current | During A/D conversion (per unit) | | | $AI_{CC}$ | — | 0.8 | 1.2 | mA | |
| | During D/A conversion (per unit) | | | | — | 0.3 | 1.0 | µA | |
| | Idle (all units) | | | | — | 0.3 | 1.0 | | |
| Reference power supply current | During A/D conversion (per unit) | | | | — | 0.06 | 0.1 | mA | |
| | During D/A conversion (per unit) | | | | — | 0.4 | 0.6 | | |
| | Idle (all units) | | | | — | 0.3 | 1.0 | µA | |
| RAM standby voltage | | | | $V_{RAM}$ | 2.5 | — | — | V | |
| $V_{CC}$ start voltage*9 | | | | $V_{CCSTART}$ | — | — | 0.8 | V | |
| $V_{CC}$ rising gradient*9 | | | | $SV_{CC}$ | — | — | 20 | ms/V | |

Notes: 1. This does not include the pins, which are multiplexed as ports 0, and 14 to 17 for 5 V tolerant.

2. This includes the multiplexed pins, but RIIC input pins for ports 14 to 17 are excluded.

3. Supply current values are with all output pins unloaded, all input pins for $V_{IH}$ = $V_{CC}$ and $V_{IL}$= 0 V, and all input pull-up resistors in the off state.

4. Measured with clocks supplied to the peripheral functions. This does not include the BGO operation.

5. $I_{CC}$ depends on f (ICLK) as follows. (ICLK : PCLK : BCLK = 8 : 4: 2)

   $I_{CC}$ max. = 0.89 x f + 11 (max.)
   $I_{CC}$ typ. = 0.30 x f + 5 (normal operation)
   $I_{CC}$ max. = 0.41 x f + 11 (sleep mode)

6. Measured with clocks not supplied to the peripheral functions. This does not include the BGO operation.

7. Incremented if data is written to or erased from the ROM or data flash for data storage during the program execution.

8. The values are for reference.

9. This can be applied when the RES# pin is held low at power-on.



Downloaded from Arrow.com.

**Table 29.3 Permissible Output Currents**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Permissible output low current (average value per pin) | All output pins except for RIIC pins | $I_{OL}$ | — | — | 2.0 | mA |
| | RIIC pins (ICFER.FMPE = 0) | $I_{OL}$ | — | — | 6.0 | mA |
| | RIIC pins (ICFER.FMPE = 1) | $I_{OL}$ | — | — | 20.0 | mA |
| Permissible output low current (max. value per pin) | All output pins except for RIIC pins | $I_{OL}$ | — | — | 4.0 | mA |
| | RIIC pins (ICFER.FMPE = 0) | $I_{OL}$ | — | — | 6.0 | mA |
| | RIIC pins (ICFER.FMPE = 1) | $I_{OL}$ | — | — | 20.0 | mA |
| Permissible output low current (total) | Total of all output pins | $\Sigma I_{OL}$ | — | — | 80 | mA |
| Permissible output high current (average value per pin) | All output pins | $-I_{OH}$ | — | — | 2.0 | mA |
| Permissible output high current (max. value per pin) | All output pins | $-I_{OH}$ | — | — | 4.0 | mA |
| Permissible output high current (total) | Total of all output pins | $\Sigma-I_{OH}$ | — | — | 80 | mA |

Caution: To protect the LSI's reliability, do not exceed the output current values in table 29.3.



Downloaded from Arrow.com.

## 29.3 AC Characteristics

**Table 29.4 Operation Frequency Value**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

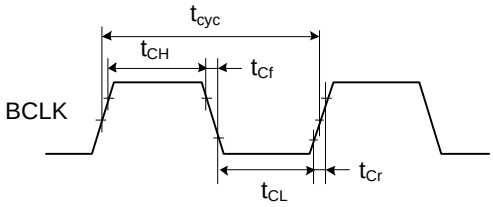
| Item | | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Operation frequency | System clock (ICLK) | f | 8 | — | 100 | MHz |
| | Peripheral module clock (PCLK) | | 8 | — | 50 | |
| | External bus clock (BCLK) | | 8 | — | 25 | |

### 29.3.1 Clock Timing

**Table 29.5 Clock Timing**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
ICLK = 8 to 100 MHz, BCLK = 8 to 25 MHz, PCLK = 8 to 50 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | Symbol | Min. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|
| Clock cycle time | $t_{cyc}$ | 40 | 125 | ns | Figure 29.1 |
| Clock high pulse width | $t_{CH}$ | 15 | — | ns | |
| Clock low pulse width | $t_{CL}$ | 15 | — | ns | |
| Clock rising time | $t_{Cr}$ | — | 5 | ns | |
| Clock falling time | $t_{Cf}$ | — | 5 | ns | |
| Oscillation settling time after reset (crystal) | $t_{OSC1}$ | 10 | — | ms | Figure 29.4 |
| Oscillation settling time after leaving software standby mode (crystal) | $t_{OSC2}$ | 10 | — | ms | Figure 29.2 |
| Oscillation settling time after leaving deep software standby mode (crystal) | $t_{OSC3}$ | 10 | — | ms | Figure 29.3 |
| External clock output delay settling time | $t_{DEXT}$ | 1 | — | ms | Figure 29.4 |
| External clock input low pulse width | $t_{EXL}$ | 30.71 | — | ns | Figure 29.5 |
| External clock input high pulse width | $t_{EXH}$ | 30.71 | — | ns | |
| External clock rising time | $t_{EXr}$ | — | 5 | ns | |
| External clock falling time | $t_{EXf}$ | — | 5 | ns | |




Test conditions $V_{OH}$ = $V_{CC}$ × 0.7, $V_{OL}$ = $V_{CC}$ × 0.3, $I_{OH}$ = - 1.0mA, $I_{OL}$ = 1.0mA, C = 30pF

**Figure 29.1 External Bus Clock Timing**



Downloaded from Arrow.com.

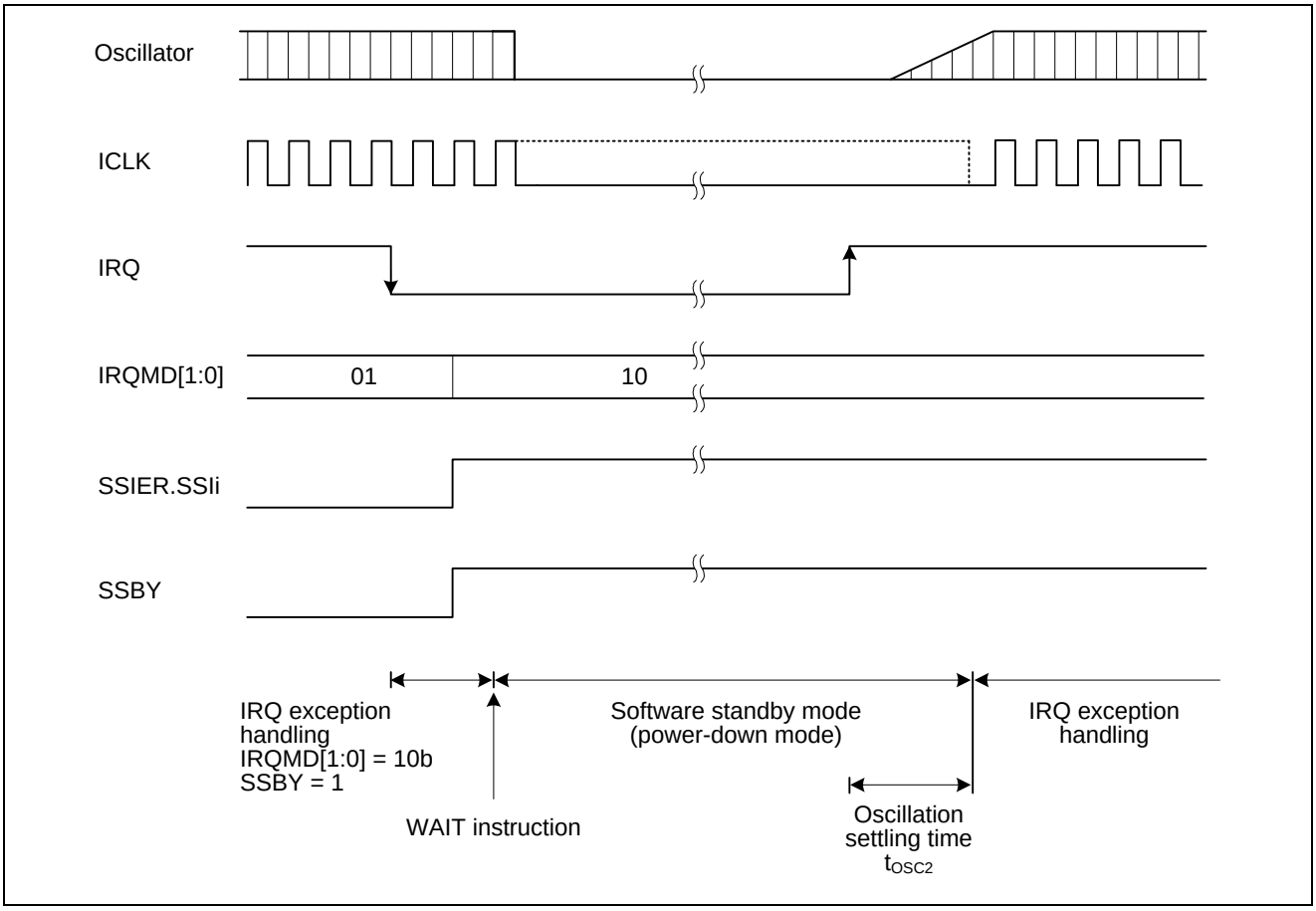



**Figure 29.2 Oscillation Settling Timing after Software Standby Mode**



Downloaded from Arrow.com.

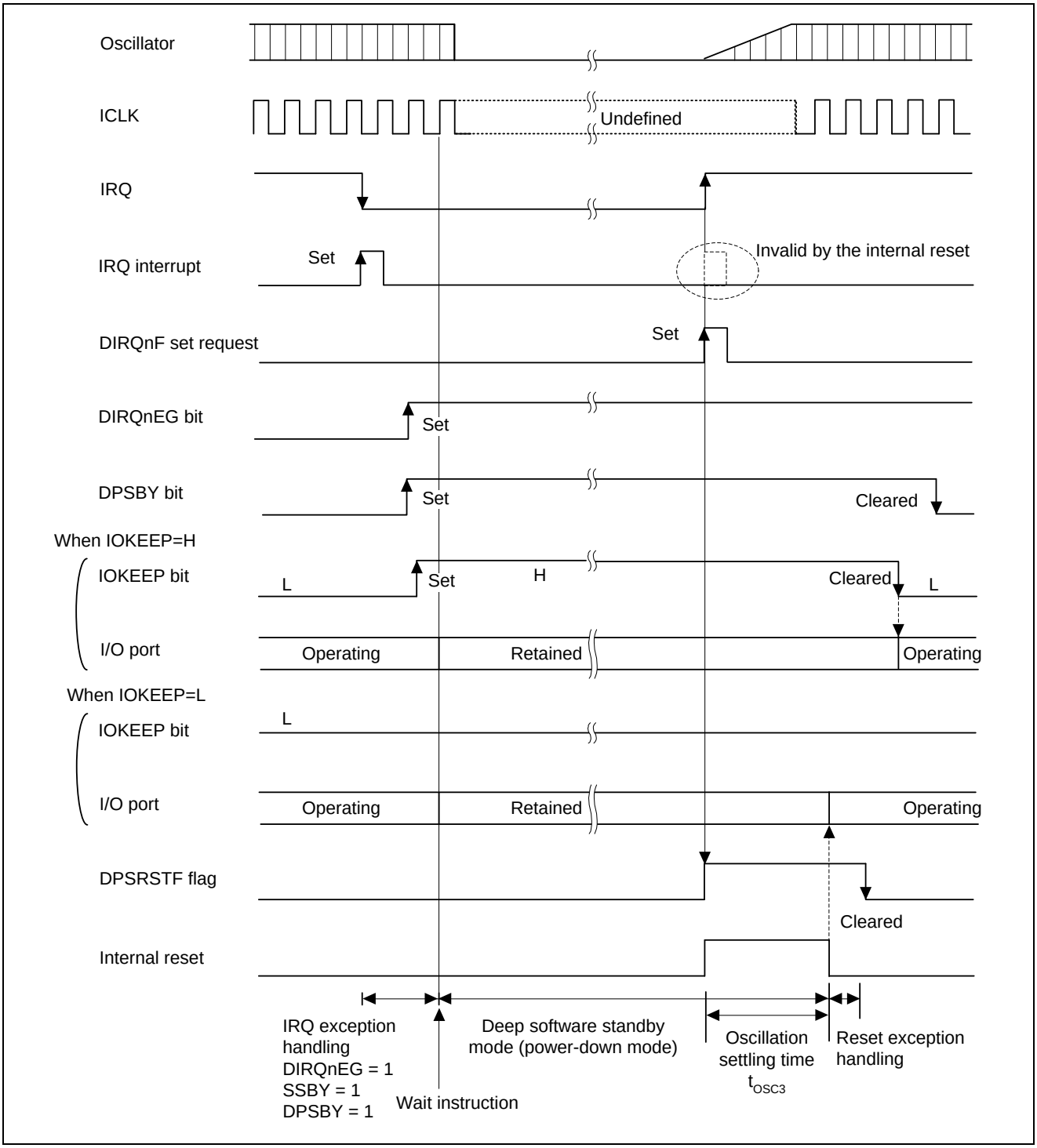



**Figure 29.3 Oscillation Settling Timing after Deep Software Standby Mode**



Downloaded from Arrow.com.

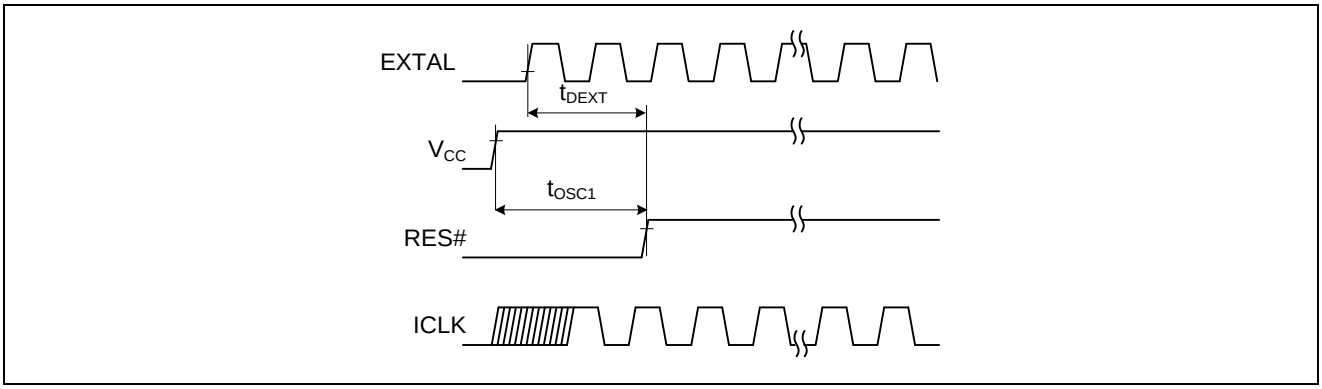



**Figure 29.4 Oscillation Settling Timing**

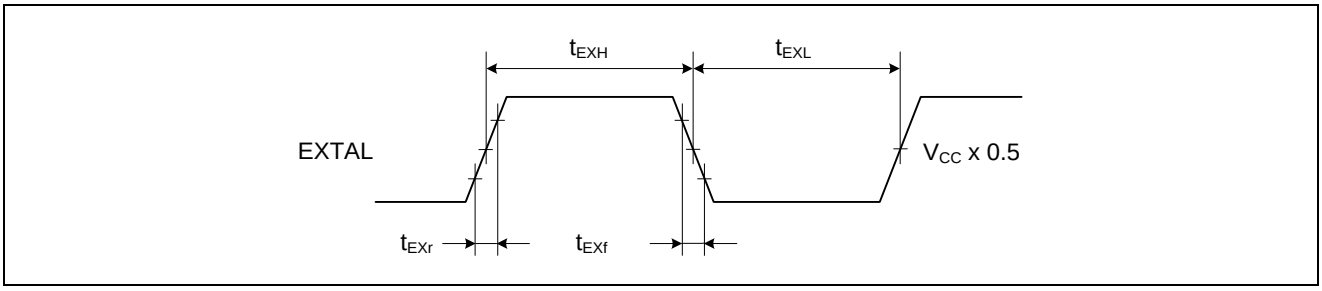



**Figure 29.5 External Input Clock Timing**



Downloaded from Arrow.com.

## 29.3.2 Control Signal Timing

**Table 29.6 Control Signal Timing**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
ICLK = 8 to 100 MHz, BCLK = 8 to 25 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | Symbol | Min. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|
| RES# pulse width (except for ROM, data flash programming/erasure) | $t_{RESW}$*1 | 20 | — | $t_{cyc}$ | Figure 29.6 |
| | | 1.5 | — | µs | |
| Internal reset time (during ROM, data flash programming/erasure) | $t_{RESW2}$ *2 | 35 | — | µs | |
| NMI pulse width | $t_{NMIW}$ | 200 | — | ns | Figure 29.7 |
| IRQ pulse width | $t_{IRQW}$ | 200 | — | ns | Figure 29.8 |

Notes: 1. Both the time and the number of cycles should satisfy the specifications.
2. This is to specify the FCU reset and the WDT reset.

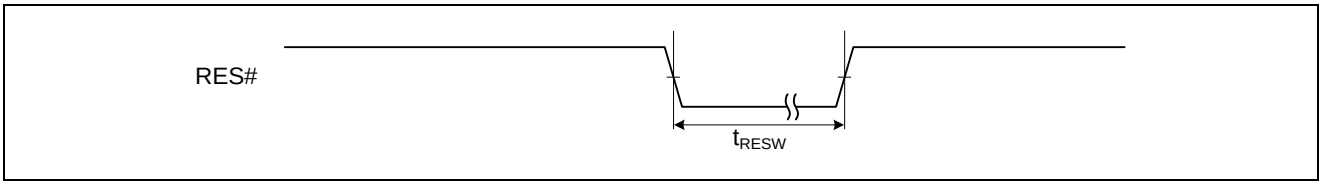



**Figure 29.6 Reset Input Timing**

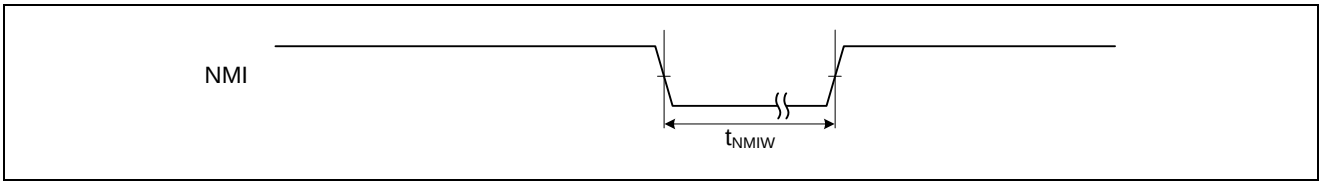



**Figure 29.7 NMI Interrupt Input Timing**

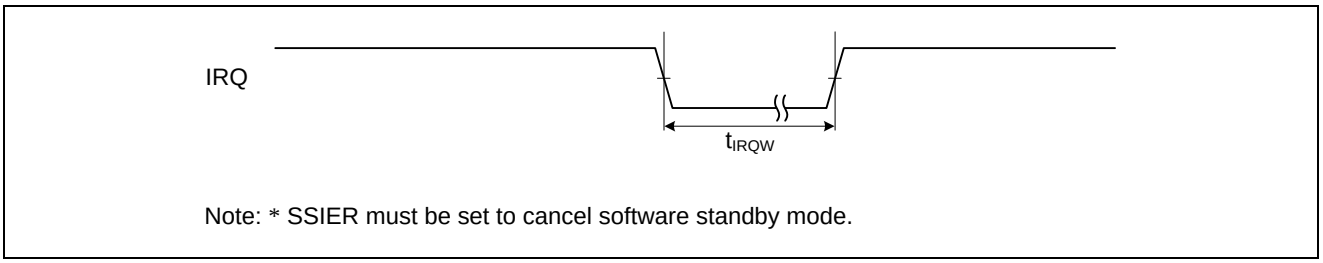



**Figure 29.8 IRQ Interrupt Input Timing**



Downloaded from Arrow.com.

## 29.3.3 Bus Timing

**Table 29.7 Bus Timing**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V, BCLK = 8 to 25 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)
Output load conditions: $V_{OH}$ = $V_{CC}$ x 0.5, $V_{OL}$ = $V_{CC}$ x 0.5, $I_{OH}$ = -1.0 mA, $I_{OL}$ = 1.0 mA, C = 30 pF

| Item | Symbol | Min. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|
| Address delay time | $t_{AD}$ | — | 30 | ns | Figures 29.9 to 29.12 |
| Byte control delay time | $t_{BCD}$ | — | 30 | ns | |
| CS# delay time | $t_{CSD}$ | — | 30 | ns | |
| RD# delay time | $t_{RSD}$ | — | 20 | ns | |
| RD# setup time | $t_{RSS}$ | 0.5 × (1/BCLK) - 20 | — | ns | |
| Read data setup time | $t_{RDS}$ | 15 | — | ns | |
| Read data hold time | $t_{RDH}$ | 0 | — | ns | |
| WR# delay time | $t_{WRD}$ | — | 20 | ns | |
| WR# setup time | $t_{WRS}$ | 0.5 × (1/BCLK) - 20 | — | ns | |
| Write data delay time | $t_{WDD}$ | — | 35 | ns | |
| Write data hold time | $t_{WDH}$ | 0 | — | ns | |
| WAIT# setup time | $t_{WTS}$ | 15 | — | ns | Figure 29.13 |
| WAIT# hold time | $t_{WTH}$ | 0 | — | ns | |



Downloaded from Arrow.com.

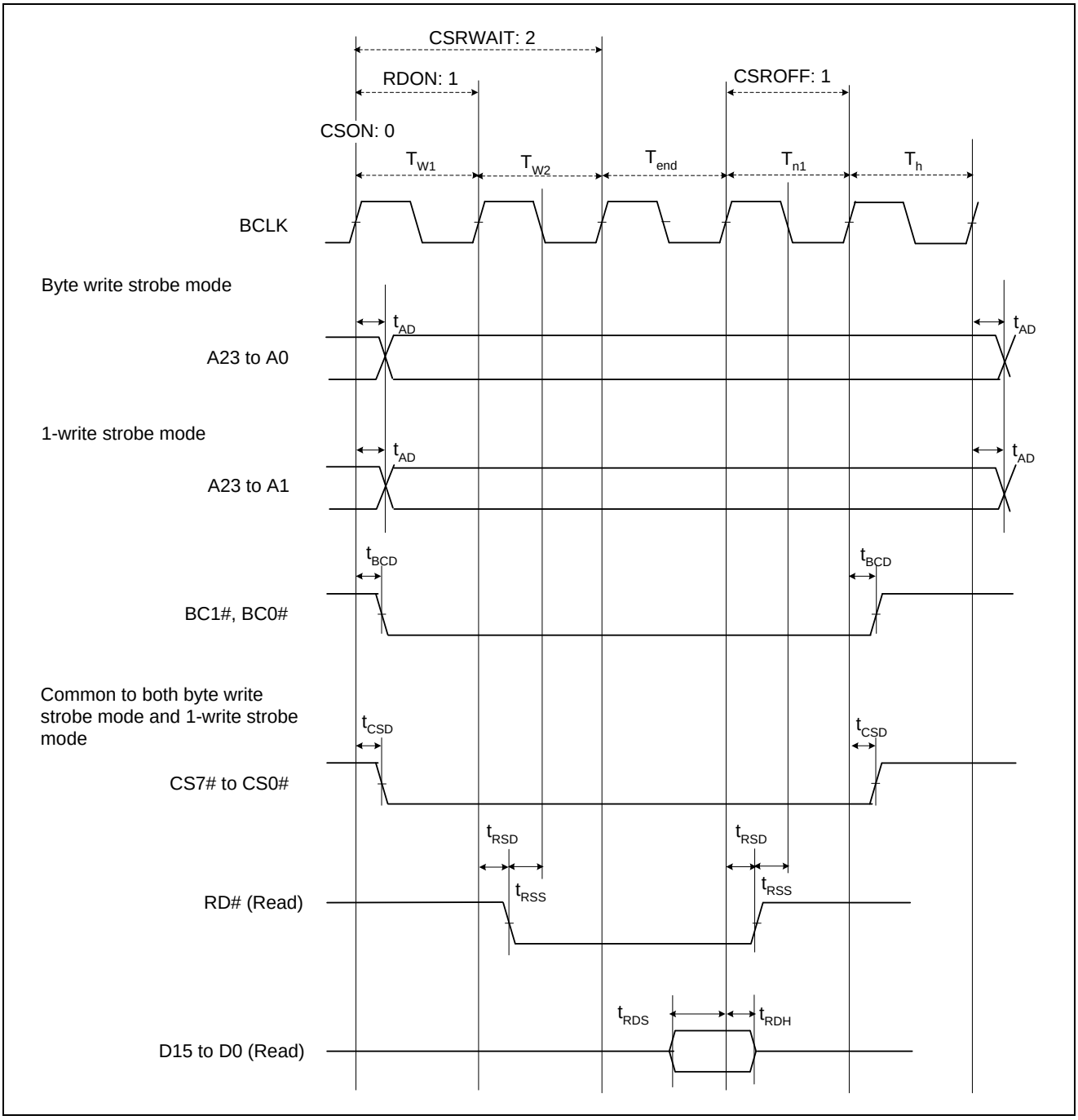
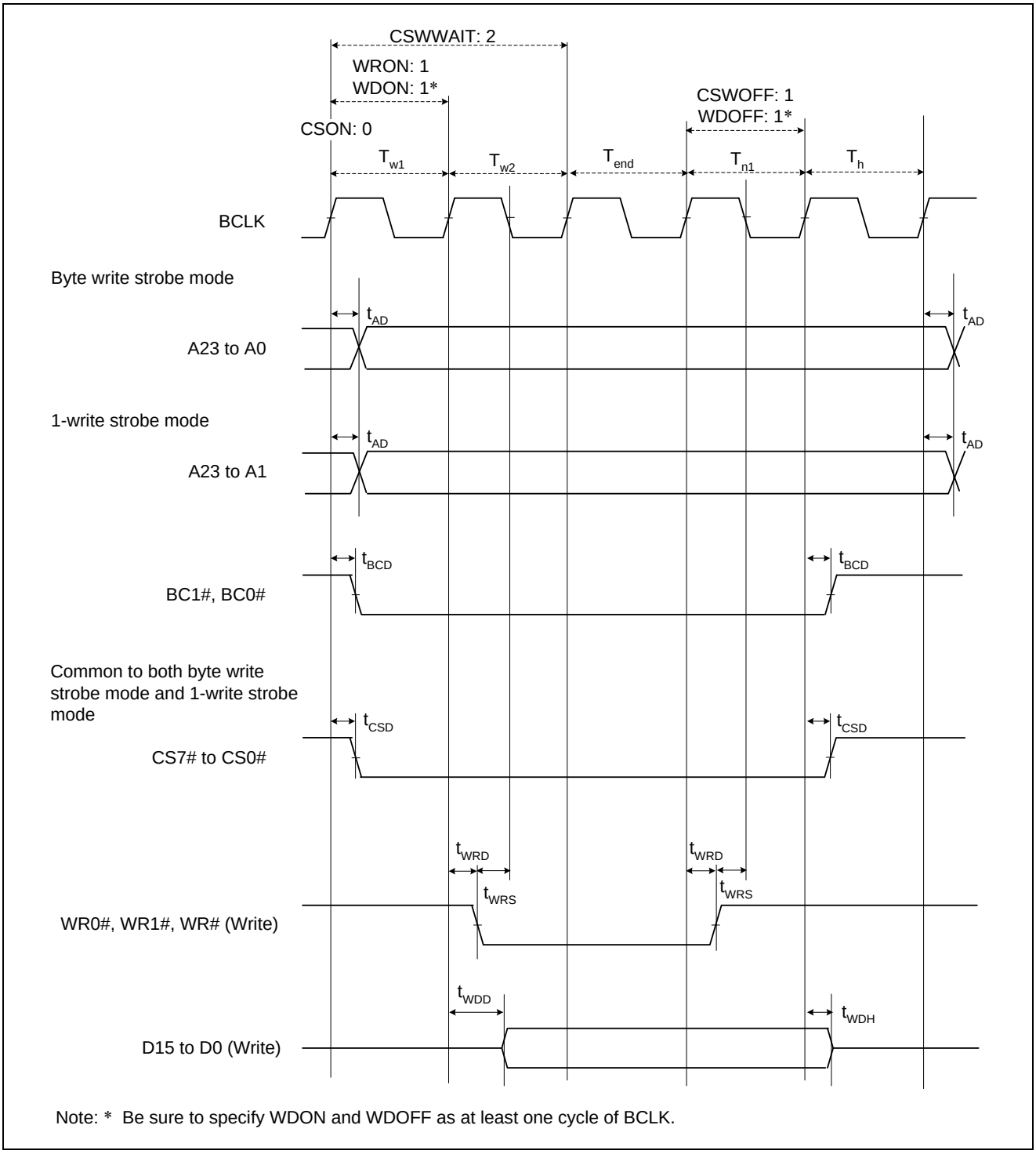



**Figure 29.9 External Bus Timing/Normal Read Cycle (Bus Clock Synchronized)**



Downloaded from Arrow.com.




Note: * Be sure to specify WDON and WDOFF as at least one cycle of BCLK.

**Figure 29.10 External Bus Timing/Normal Write Cycle (Bus Clock Synchronized)**



Downloaded from Arrow.com.

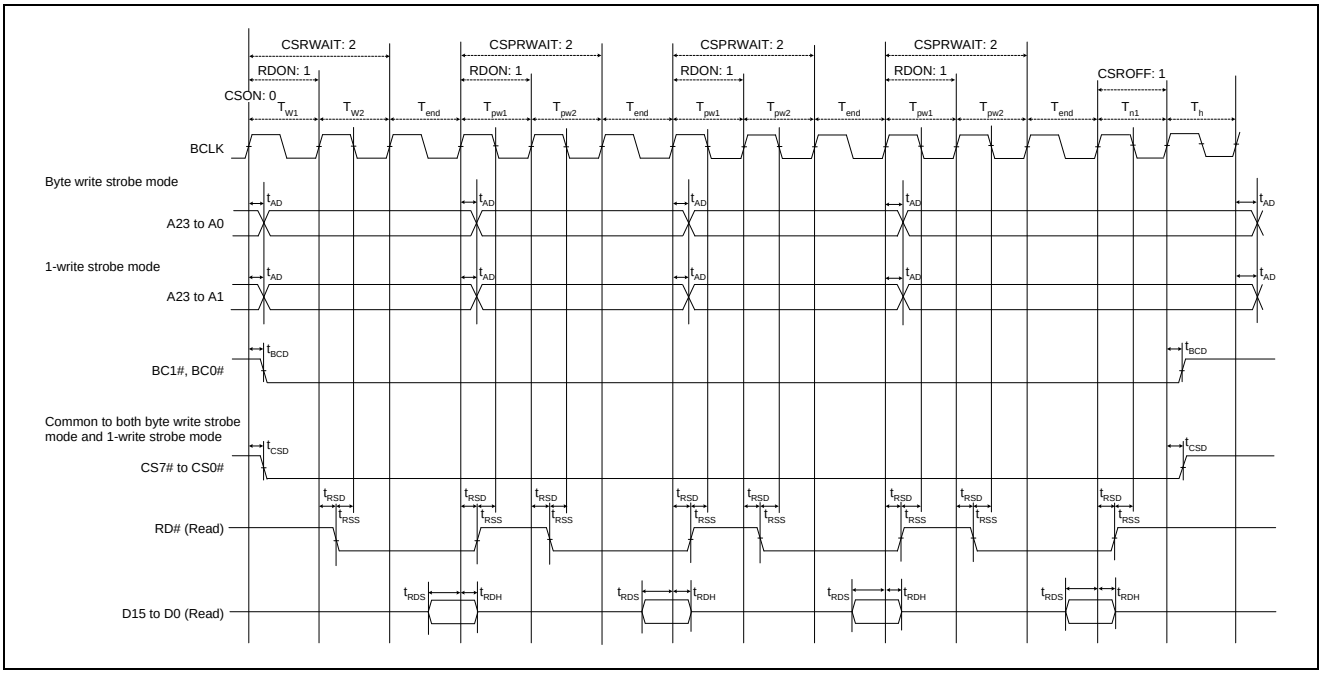



**Figure 29.11 External Bus Timing/Page Read Cycle (Bus Clock Synchronized)**

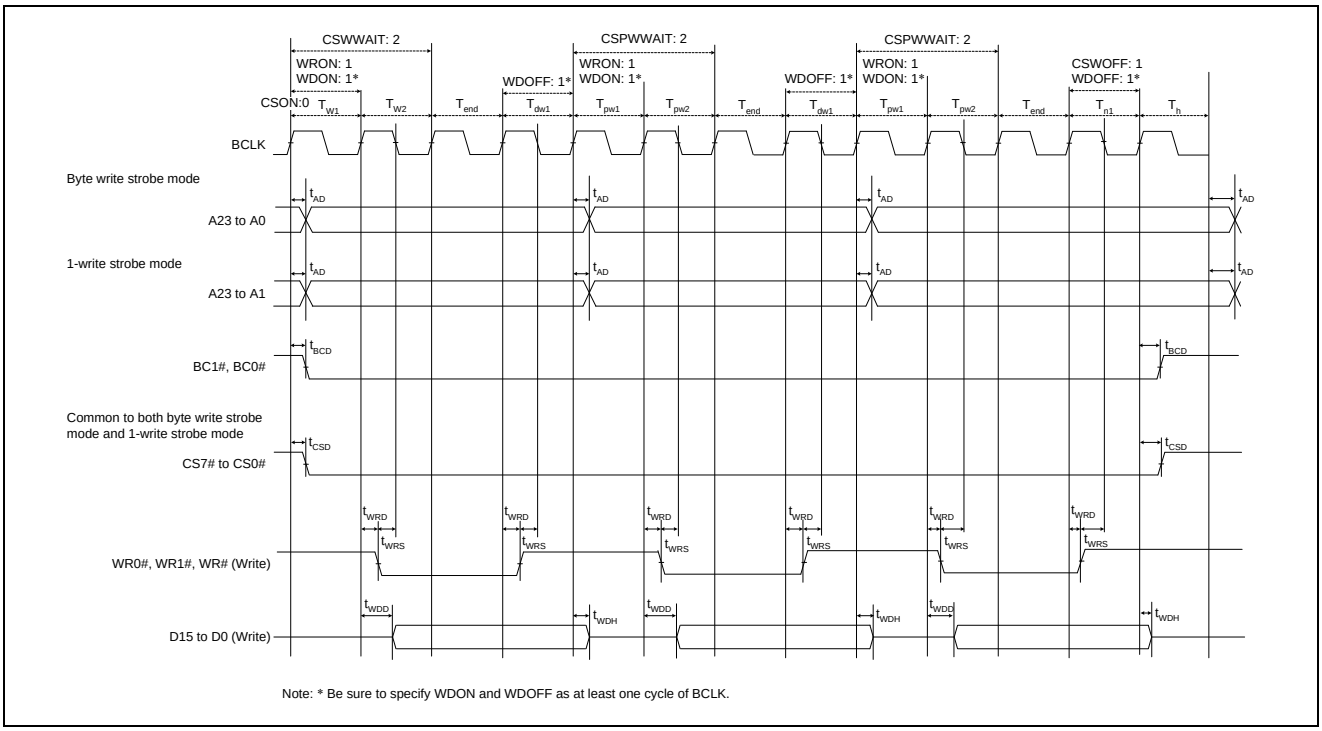



**Figure 29.12 External Bus Timing/Page Write Cycle (Bus Clock Synchronized)**



Downloaded from Arrow.com.

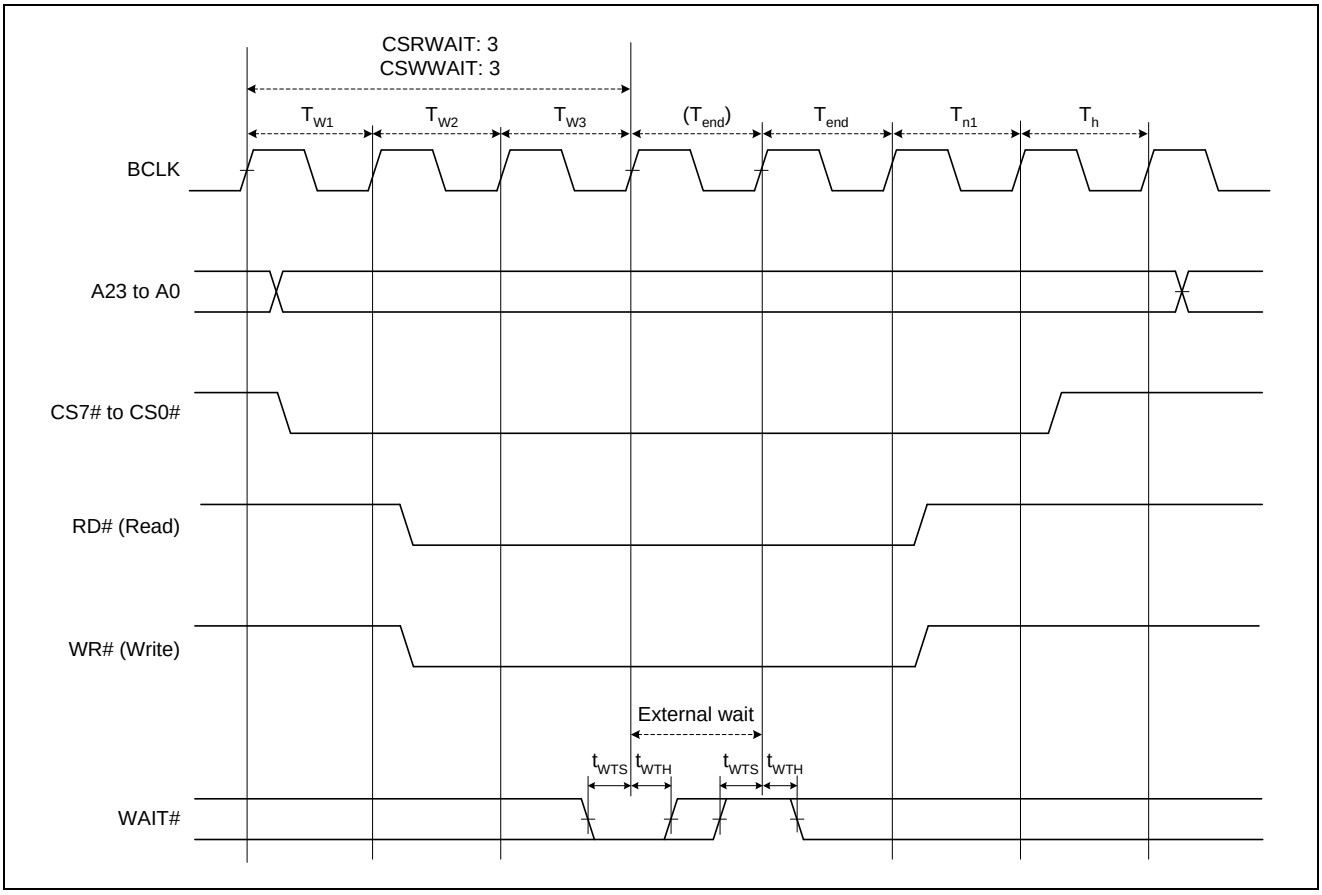



**Figure 29.13 External Bus Timing/External Wait Control**



Downloaded from Arrow.com.

## 29.3.4 Timing of On-Chip Peripheral Modules

**Table 29.8 Timing of On-Chip Peripheral Modules (1)**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V, PCLK = 8 to 50 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)
Output load conditions: $V_{OH}$ = $V_{CC}$ x 0.5, $V_{OL}$ = $V_{CC}$ x 0.5, $I_{OH}$ = -1.0 mA, $I_{OL}$ = 1.0 mA, C = 30 pF

| Item | | | Symbol | Min. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| I/O ports | Output data delay time | | $t_{PWD}$ | — | 40 | ns | Figure 29.14 |
| | Input data setup time | | $t_{PRS}$ | 25 | — | ns | |
| | Input data hold time | | $t_{PRH}$ | 25 | — | ns | |
| TPU | Timer output delay time | | $t_{TOCD}$ | — | 40 | ns | Figure 29.15 |
| | Timer input setup time | | $t_{TICS}$ | 25 | — | ns | |
| | Timer clock input setup time | | $t_{TCKS}$ | 25 | — | ns | Figure 29.16 |
| | Timer clock pulse width | Single-edge setting | $t_{TCKWH}$ | 1.5 x (1/PCLK) | — | $t_{cyc}$ | |
| | | Both-edge setting | $t_{TCKWL}$ | 2.5 x (1/PCLK) | — | $t_{cyc}$ | |
| PPG | Pulse output delay time | | $t_{POD}$ | — | 40 | ns | Figure 29.17 |
| 8-bit timer | Timer output delay time | | $t_{TMOD}$ | — | 40 | ns | Figure 29.18 |
| | Timer reset input setup time | | $t_{TMRS}$ | 25 | — | ns | Figure 29.19 |
| | Timer clock input setup time | | $t_{TMCS}$ | 25 | — | ns | Figure 29.20 |
| | Timer clock pulse width | Single-edge setting | $t_{TMCWH}$ | 1.5 x (1/PCLK) | — | $t_{cyc}$ | |
| | | Both-edge setting | $t_{TMCWL}$ | 2.5 x (1/PCLK) | — | $t_{cyc}$ | |
| WDT | Overflow output delay time | | $t_{WOVD}$ | — | 40 | ns | Figure 29.21 |
| SCI | Input clock cycle | Asynchronous | $t_{Scyc}$ | 4 x (1/PCLK) | — | $t_{cyc}$ | Figure 29.22 |
| | | Clock synchronous | | 6 x (1/PCLK) | — | | |
| | Input clock pulse width | | $t_{SCKW}$ | 0.4 x $t_{Scyc}$ | 0.6 x $t_{Scyc}$ | $t_{Scyc}$ | |
| | Input clock rise time | | $t_{SCKr}$ | — | 20 | ns | |
| | Input clock fall time | | $t_{SCKf}$ | — | 20 | ns | |
| | Output clock cycle | Asynchronous | $t_{Scyc}$ | 4 x (1/PCLK) | — | $t_{cyc}$ | |
| | | Clock synchronous | | 6 x (1/PCLK) | — | | |
| | Output clock pulse width | | $t_{SCKW}$ | 0.4 x $t_{Scyc}$ | 0.6 x $t_{Scyc}$ | $t_{Scyc}$ | |
| | Output clock rise time | | $t_{SCKr}$ | — | 20 | ns | |
| | Output clock fall time | | $t_{SCKf}$ | — | 20 | ns | |
| | Transmit data delay time | | $t_{TXD}$ | — | 40 | ns | Figure 29.23 |
| | Receive data setup time (clock synchronous) | | $t_{RXS}$ | 40 | — | ns | |
| | Receive data hold time (clock synchronous) | | $t_{RXH}$ | 40 | — | ns | |
| A/D converter | Trigger input setup time | | $t_{TRGS}$ | 25 | — | ns | Figure 29.24 |



Downloaded from Arrow.com.

**Table 29.8 Timing of On-Chip Peripheral Modules (2)**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V, PCLK = 8 to 50 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. *1*2 | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| RIIC (Standard-mode) ICFER.FMPE = 0 | SCL input cycle time | $t_{SCL}$ | 8(10) x (1/PCLK) + 1300 | — | ns | Figure 29.25 |
| | SCL input high pulse width | $t_{SCLH}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | SCL input low pulse width | $t_{SCLL}$ | 5x (1/PCLK) + 1000 | — | ns | |
| | SCL, SDA input rising time | $t_{Sr}$ | — | 1000 | ns | |
| | SCL, SDA input falling time | $t_{Sf}$ | — | 300 | ns | |
| | SCL, SDA input spike pulse removal time | $t_{SP}$ | 0 | 4 x (1/PCLK) | ns | |
| | SDA input bus free time | $t_{BUF}$ | 5x (1/PCLK) + 1000 | — | ns | |
| | Start condition input hold time | $t_{STAH}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | Re-start condition input setup time | $t_{STAS}$ | 5x (1/PCLK) + 1000 | — | ns | |
| | Stop condition input setup time | $t_{STOS}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | Data input setup time | $t_{SDAS}$ | 250 | — | ns | |
| | Data input hold time | $t_{SDAH}$ | 0 | — | ns | |
| | SCL, SDA capacitive load | $C_b$ | — | 400 | pF | |
| RIIC (Fast-mode) ICFER.FMPE = 0 | SCL input cycle time | $t_{SCL}$ | 8(10) x (1/PCLK) + 600 | — | ns | |
| | SCL input high pulse width | $t_{SCLH}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | SCL input low pulse width | $t_{SCLL}$ | 5 x (1/PCLK) + 300 | — | ns | |
| | SCL, SDA input rising time | $t_{Sr}$ | 20 + 0.1$C_b$ | 300 | ns | |
| | SCL, SDA input falling time | $t_{Sf}$ | 20 + 0.1$C_b$ | 300 | ns | |
| | SCL, SDA input spike pulse removal time | $t_{SP}$ | 0 | 4 x (1/PCLK) | ns | |
| | SDA input bus free time | $t_{BUF}$ | 5 x (1/PCLK) + 300 | — | ns | |
| | Start condition input hold time | $t_{STAH}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | Re-start condition input setup time | $t_{STAS}$ | 5 x (1/PCLK) + 300 | — | ns | |
| | Stop condition input setup time | $t_{STOS}$ | 3(5) x (1/PCLK) + 300 | — | ns | |
| | Data input setup time | $t_{SDAS}$ | 100 | — | ns | |
| | Data input hold time | $t_{SDAH}$ | 0 | — | ns | |
| | SCL, SDA capacitive load | $C_b$ | — | 400 | pF | |



Downloaded from Arrow.com.

**Table 29.8 Timing of On-Chip Peripheral Modules (3)**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V,
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. *1*2 | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| RIIC (Fast-mode+) ICFER.FMPE = 1 | SCL input cycle time | $t_{SCL}$ | 8(10) x (1/PCLK) + 240 | — | ns | Figure 29.25 |
| | SCL input high pulse width | $t_{SCLH}$ | 3(5) x (1/PCLK) + 120 | — | ns | |
| | SCL input low pulse width | $t_{SCLL}$ | 5 x (1/PCLK) + 120 | — | ns | |
| | SCL, SDA input rising time | $t_{Sr}$ | — | 120 | ns | |
| | SCL, SDA input falling time | $t_{Sf}$ | — | 120 | ns | |
| | SCL, SDA input spike pulse removal time | $t_{SP}$ | 0 | 4 x (1/PCLK) | ns | |
| | SDA input bus free time | $t_{BUF}$ | 5 x (1/PCLK) + 120 | — | ns | |
| | Start condition input hold time | $t_{STAH}$ | 3(5) x (1/PCLK) + 120 | — | ns | |
| | Re-start condition input setup time | $t_{STAS}$ | 5 x (1/PCLK) + 120 | — | ns | |
| | Stop condition input setup time | $t_{STOS}$ | 3(5) x (1/PCLK) + 120 | — | ns | |
| | Data input setup time | $t_{SDAS}$ | 50 | — | ns | |
| | Data input hold time | $t_{SDAH}$ | 0 | — | ns | |
| | SCL, SDA capacitive load | $C_b$ | — | 550 | pF | |
| Boundary scan (176-pin LFBGA) | TCK clock cycle time | tTCKcyc | 100 | — | ns | Figure 29.26 |
| | TCK clock high level pulse width | tTCKH | 45 | — | ns | |
| | TCK clock low level pulse width | tTCKL | 45 | — | ns | |
| | TCK clock rising time | tTCKr | — | 5 | ns | |
| | TCK clock falling time | tTCKf | — | 5 | ns | |
| | TRST# pulse width | tTRSTW | 20 | — | Tcyc | Figure 29.27 |
| | TMS setup time | tTMSS | 20 | — | ns | Figure 29.28 |
| | TMS hold time | tTMSH | 20 | — | ns | |
| | TDI setup time | tTDIS | 20 | — | ns | |
| | TDI hold time | tTDIH | 20 | — | ns | |
| | TDO data delay time | tTDOD | — | 40 | ns | |

Notes:1. The value in parentheses is used when ICMR3.NF[1:0] are set to 11b while a digital filter is enabled with ICFER.NFE = 1.

2. Cb indicates the total capacity of the bus line.



Downloaded from Arrow.com.

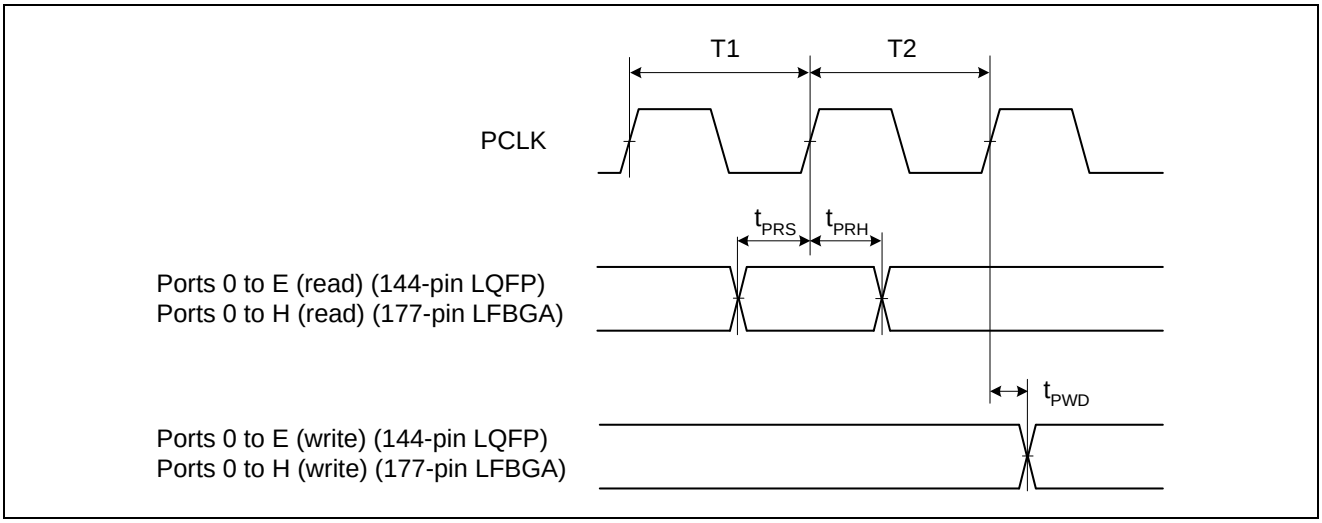



**Figure 29.14 I/O Port Input/Output Timing**

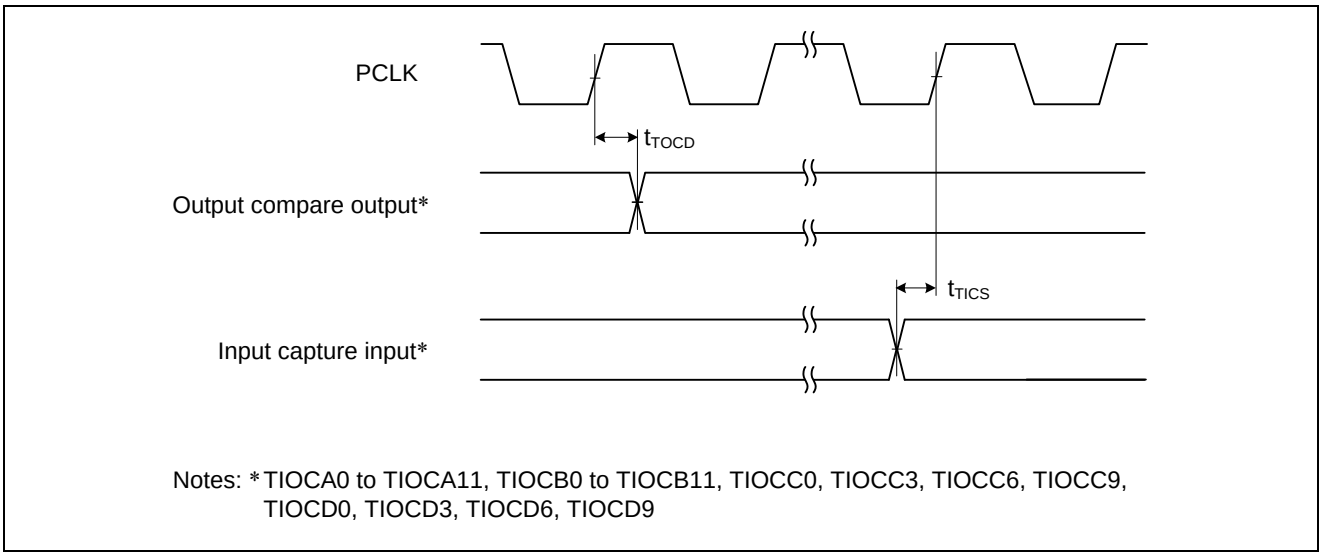



**Figure 29.15 TPU Input/Output Timing**

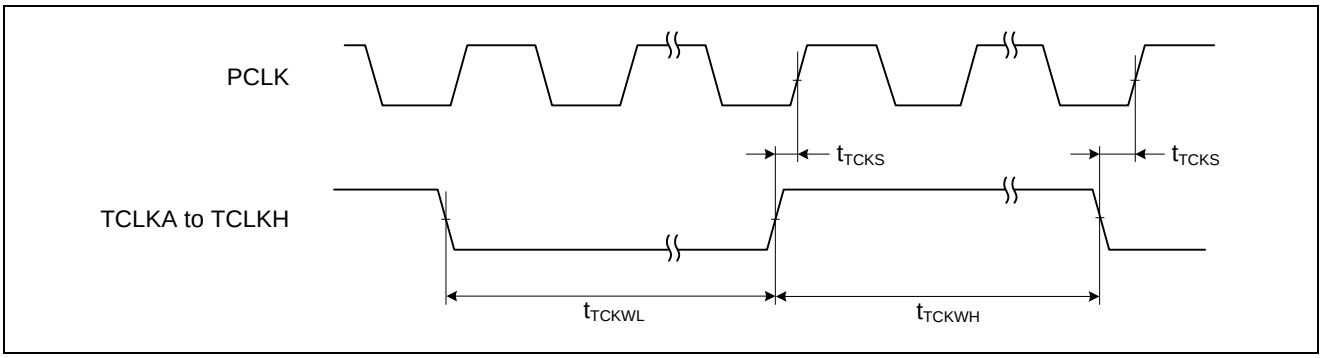



**Figure 29.16 TPU Clock Input Timing**



Downloaded from Arrow.com.

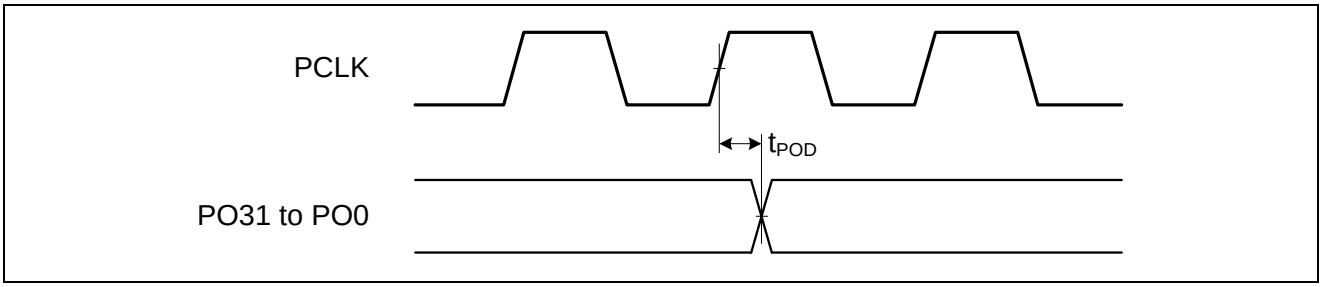



**Figure 29.17 PPG Output Timing**

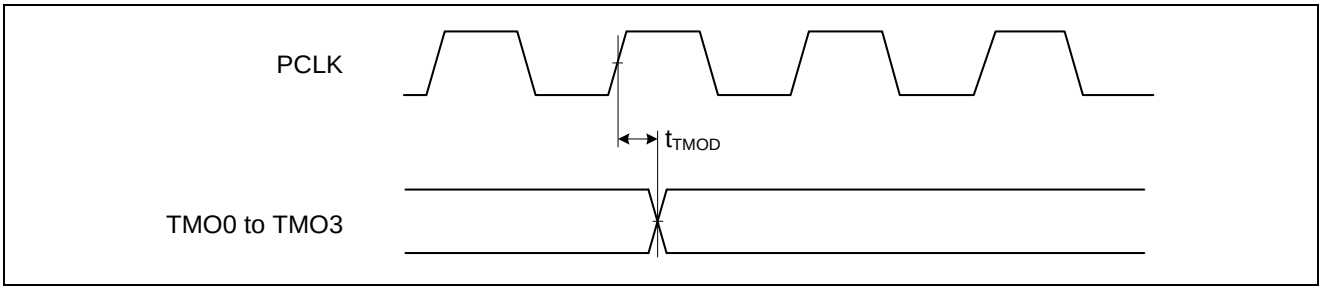



**Figure 29.18 8-Bit Timer Output Timing**

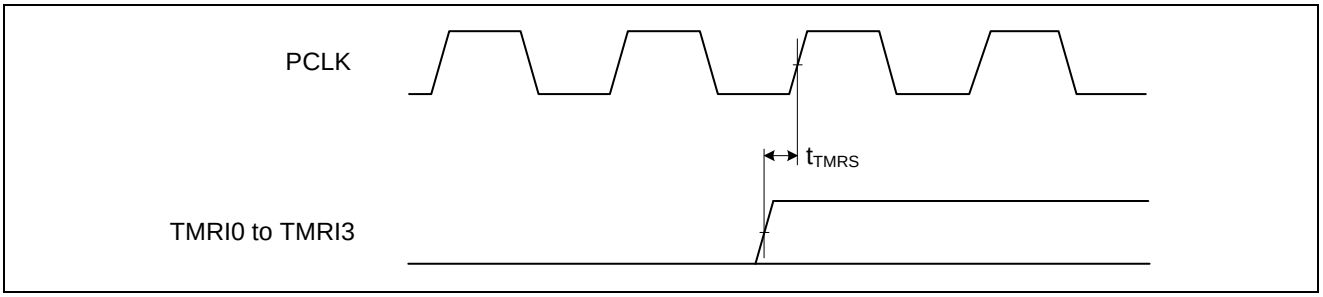



**Figure 29.19 8-Bit Timer Reset Input Timing**

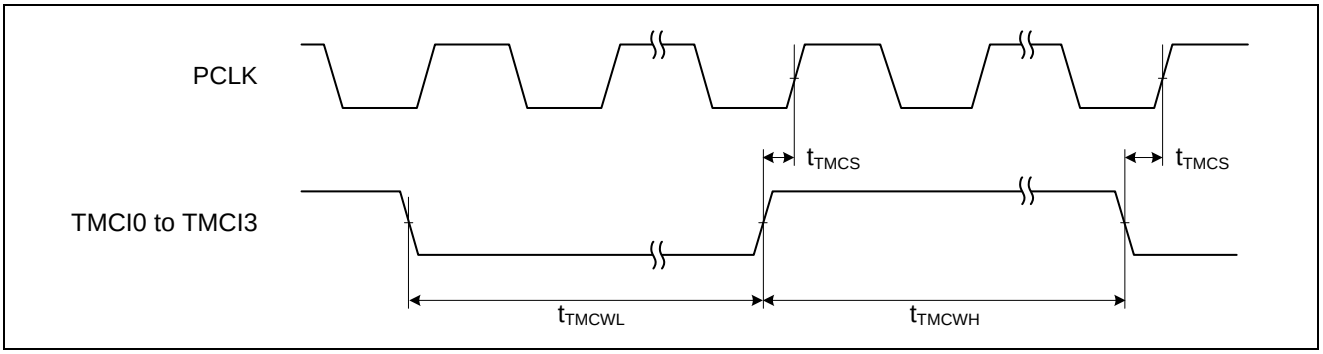



**Figure 29.20 8-Bit Timer Clock Input Timing**



Downloaded from Arrow.com.

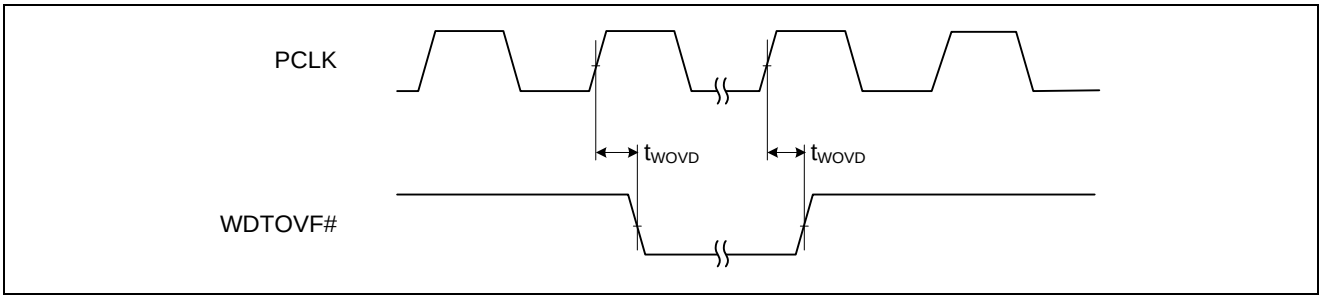



**Figure 29.21 WDT Output Timing**

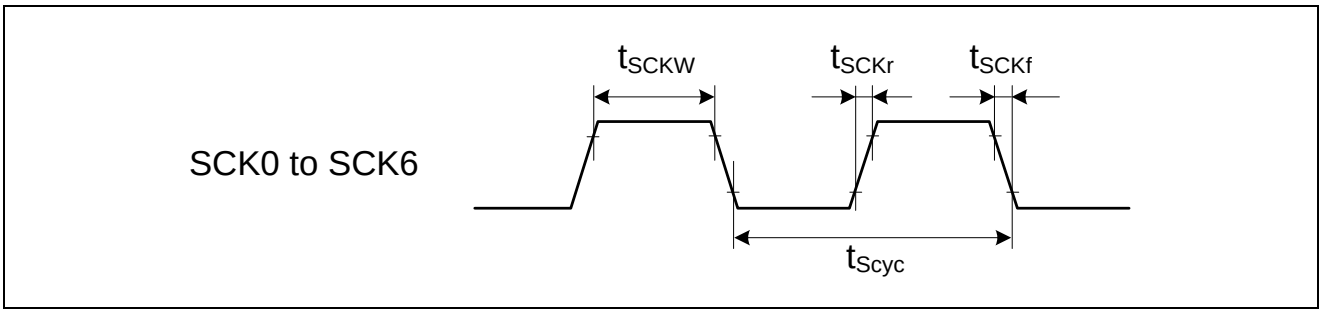



**Figure 29.22 SCK Clock Input Timing**

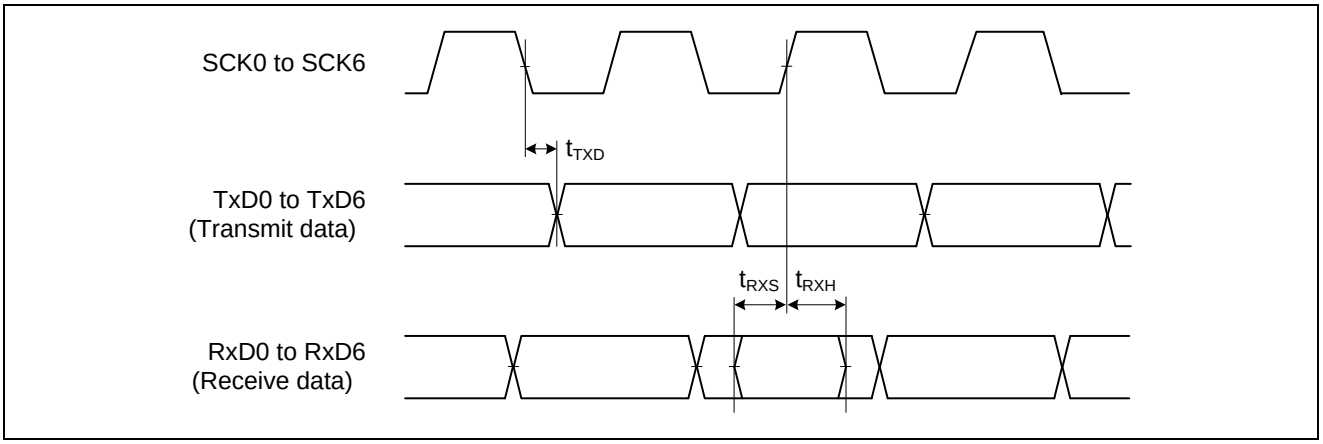



**Figure 29.23 SCI Input/Output Timing: Clock Synchronous Mode**

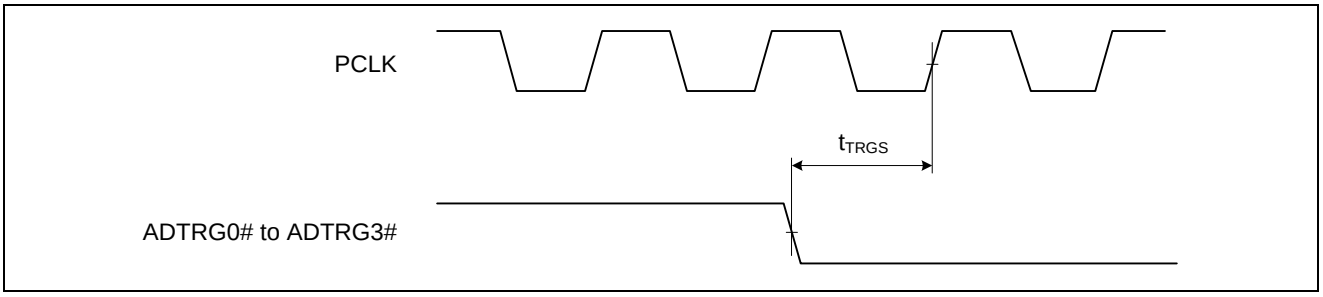



**Figure 29.24 A/D Converter External Trigger Input Timing**



Downloaded from Arrow.com.

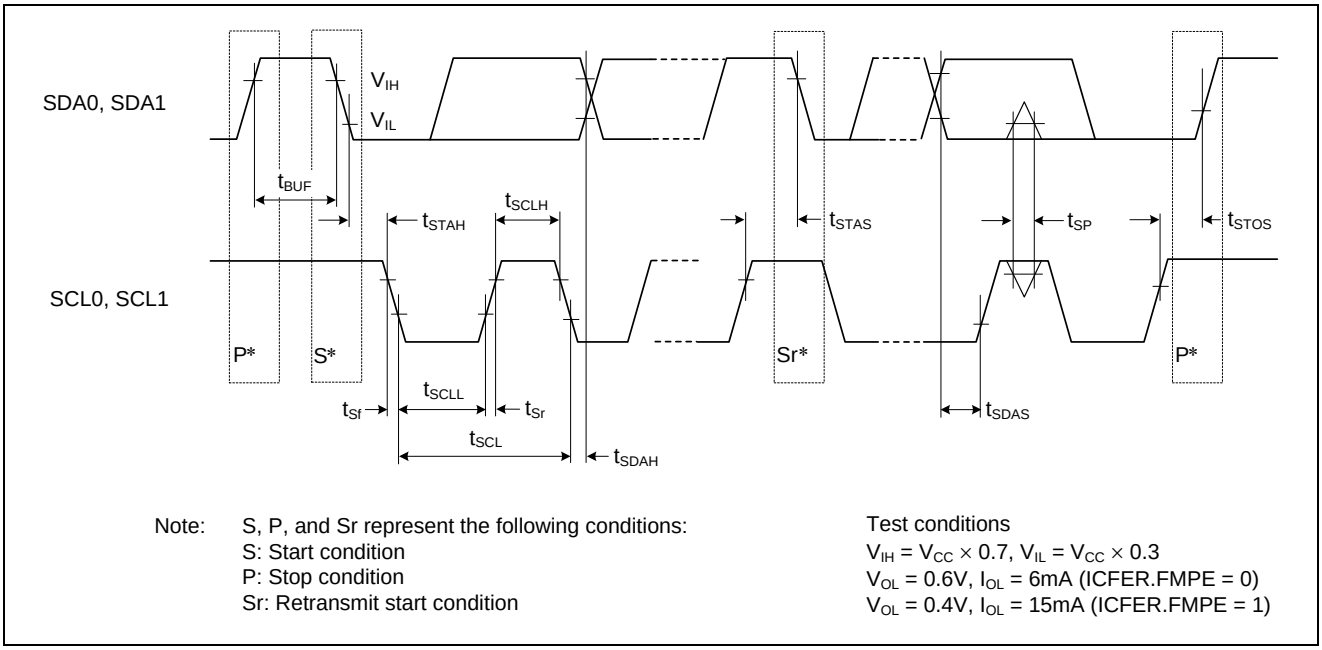



**Figure 29.25 I²C Bus Interface Input/Output Timing**

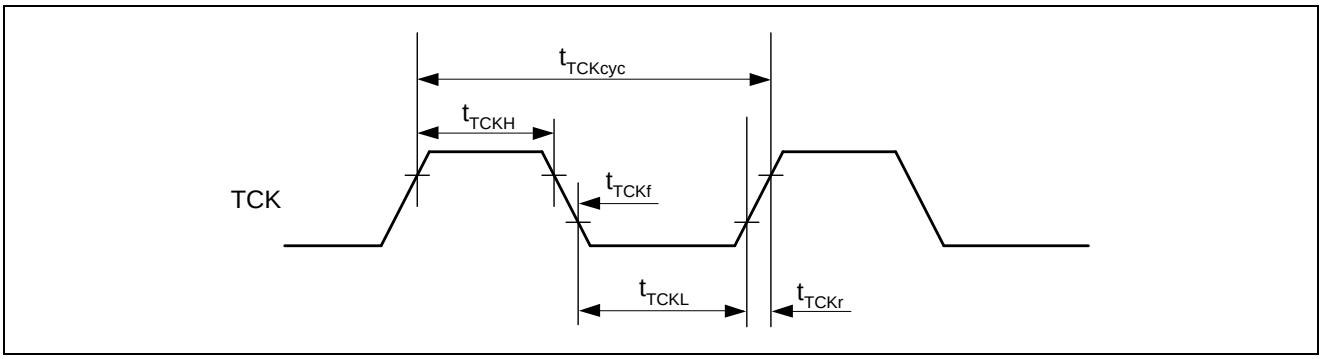



**Figure 29.26 Boundary Scan TCK Timing**

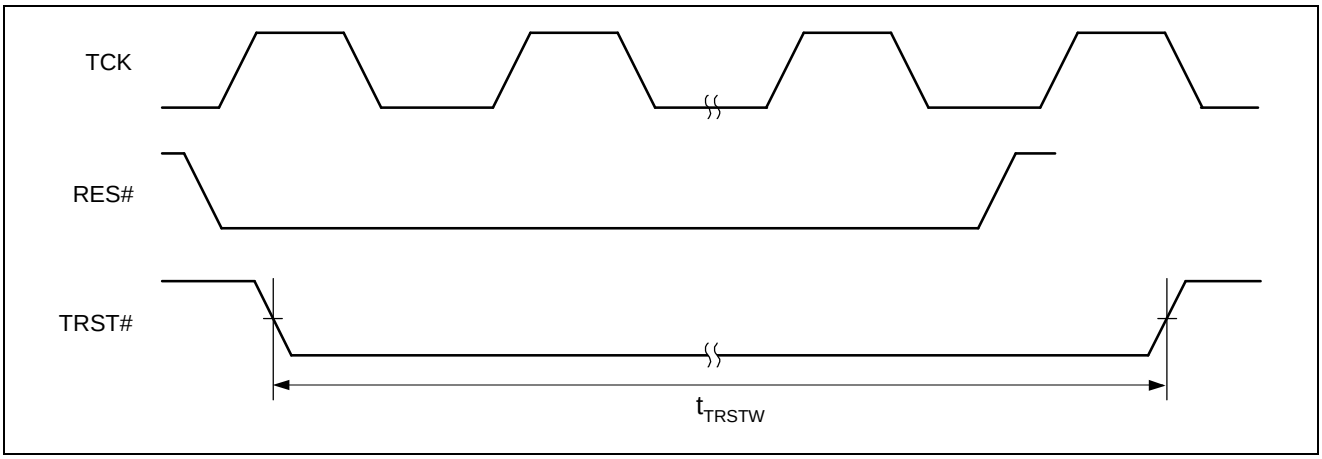



**Figure 29.27 Boundary Scan TRST# Timing**



Downloaded from Arrow.com.

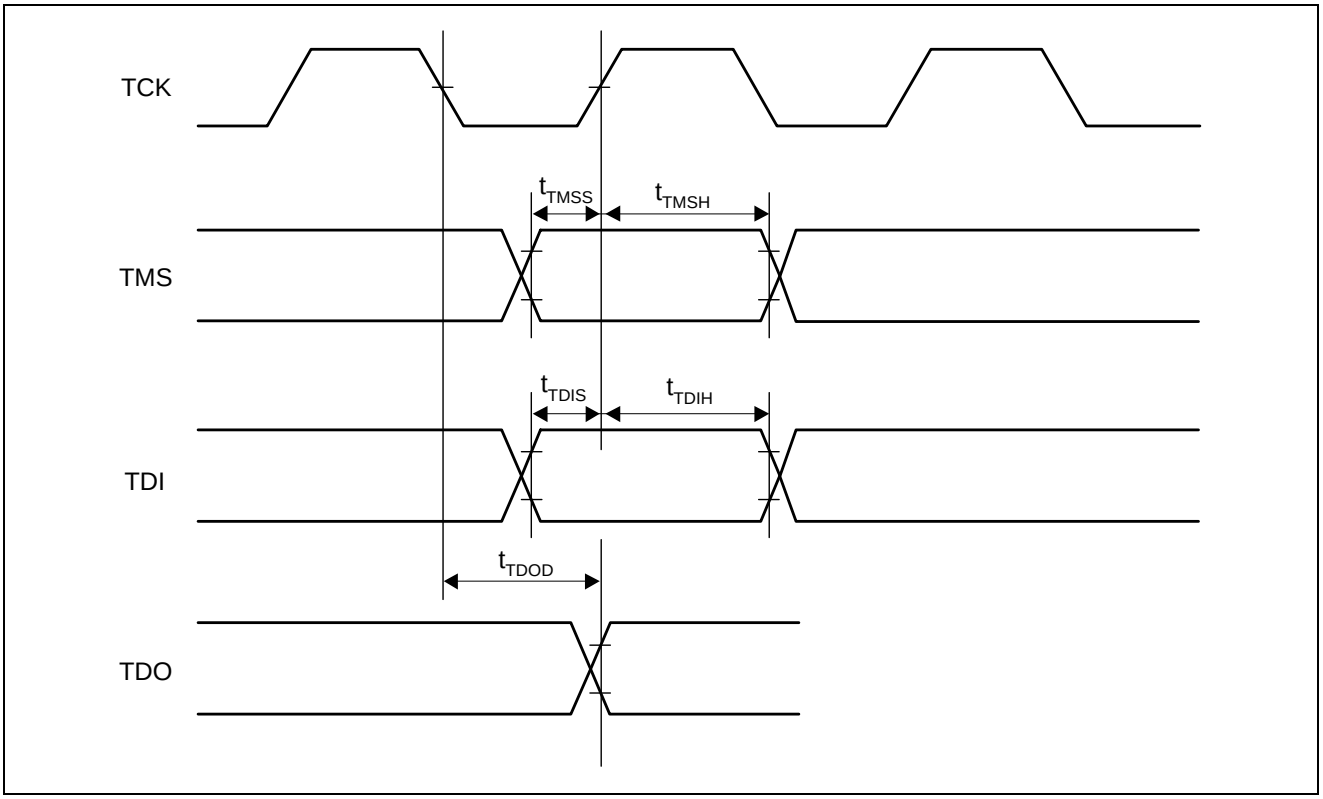



**Figure 29.28 Boundary Scan Input/Output Timing**



Downloaded from Arrow.com.

## 29.4 A/D Conversion Characteristics

**Table 29.9 A/D Conversion Characteristics**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V, PCLK = 8 to 50 MHz, ADCLK = 4 to 50 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Resolution | | | 10 | 10 | 10 | Bit | |
| Conversion time*1 (ADCLK = 50-MHz operation) | With 0.1-µF external capacitor | When the capacitor is charged enough*2 | 0.8 (0.3)*3 | — | — | µs | Sampling 15 states |
| | Without external capacitor | Permissible signal source impedance (max.) = 1.0 kΩ | 1.0 (0.5)*3 | — | — | | Sampling 25 states |
| | | Permissible signal source impedance (max.) = 5.0 kΩ | 2.6 (2.1)*3 | — | — | | Sampling 105 states |
| Analog input capacitance | | | — | — | 6.0 | pF | |
| INL integral nonlinearity error (INL) | | | — | ±1.5 | ±3.0 | LSB | |
| Offset error | | | — | ±1.5 | ±3.0 | LSB | |
| Full-scale error | | | — | ±1.5 | ±3.0 | LSB | |
| Quantization error | | | — | ±0.5 | — | LSB | |
| Absolute accuracy | | | — | ±1.5 | ±3.0 | LSB | |
| DNL differential nonlinearity error (DNL) | | | — | ±0.5 | ±1.0 | LSB | |

Notes: 1. The conversion time includes the sampling time and the comparison time. As the test conditions, the number of sampling states is indicated.
2. The scanning is not supported.
3. The value in parentheses indicates the sampling time.

## 29.5 D/A Conversion Characteristics

**Table 29.10 D/A Conversion Characteristics**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V, PCLK = 8 to 50 MHz
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|
| Resolution | 10 | 10 | 10 | Bit | |
| Conversion time | — | — | 3 | µs | 20-pF capacitive load |
| Absolute accuracy | — | ±2.0 | ±4.0 | LSB | 2-MΩ resistive load |
| | — | — | ±3.0 | LSB | 4-MΩ resistive load |
| | — | — | ±2.0 | LSB | 10-MΩ resistive load |
| RO output resistance | — | 3.6 | — | kΩ | |



Downloaded from Arrow.com.

## 29.6 ROM (Flash Memory for Code Storage) Characteristics

**Table 29.11 ROM (Flash Memory for Code Storage) Characteristics**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
Operating temperature range during programming/erasing:
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Programming time | 256 bytes | $t_{P256}$ | — | 2 | 12 | ms | PCLK = 50 MHz<br>$N_{PEC} \leq 100$ |
| | 8 Kbytes | $t_{P8K}$ | — | 45 | 100 | ms | |
| | 256 bytes | $t_{P256}$ | — | 2.4 | 14.4 | ms | PCLK = 50 MHz<br>$N_{PEC} > 100$ |
| | 8 Kbytes | $t_{P8K}$ | — | 54 | 120 | ms | |
| Erasure time | 8 Kbytes | $t_{E8K}$ | — | 50 | 120 | ms | PCLK = 50 MHz<br>$N_{PEC} \leq 100$ |
| | 64 Kbytes | $t_{E64K}$ | — | 400 | 875 | ms | |
| | 128 Kbytes | $t_{E128K}$ | — | 800 | 1750 | ms | |
| | 8 Kbytes | $t_{E8K}$ | — | 60 | 144 | ms | PCLK = 50 MHz<br>$N_{PEC} > 100$ |
| | 64 Kbytes | $t_{E64K}$ | — | 480 | 1050 | ms | |
| | 128 Kbytes | $t_{E128K}$ | — | 960 | 2100 | ms | |
| Rewrite/erase cycle*[1] | | $N_{PEC}$ | 1000*[2] | — | — | Times | |
| Suspend delay time during writing | | $t_{SPD}$ | — | — | 120 | µs | Figure 29.29<br>PCLK = 50 MHz |
| First suspend delay time during erasing (in suspend priority mode) | | $t_{SESD1}$ | — | — | 120 | µs | |
| Second suspend delay time during erasing (in suspend priority mode) | | $t_{SESD2}$ | — | — | 1.7 | ms | |
| Suspend delay time during erasing (in erasure priority mode) | | $t_{SEED}$ | — | — | 1.7 | ms | |
| Data hold time*[3] | | $T_{DRP}$ | 10 | — | — | Year | |

Notes: 1. Definition of rewrite/erase cycle:
The rewrite/erase cycle is the number of erasing for each block. When the rewrite/erase cycle is n times (n = 1000), erasing can be performed n times for each block. For instance, when 256-byte writing is performed 32 times for different addresses in 8-Kbyte block and then the entire block is erased, the rewrite/erase cycle is counted as one. However, writing to the same address for several times as one erasing is not enabled (over writing is prohibited).
2. This indicates the minimum number that guarantees the characteristics after rewriting. (The guaranteed value is in the range from one to the minimum number.)
3. This indicates the characteristic when rewrite is performed within the specification range including the minimum number.



Downloaded from Arrow.com.

## 29.7 Data Flash (Flash Memory for Data Storage) Characteristics

**Table 29.12 Data Flash (Flash Memory for Data Storage) Characteristics**

Conditions: $V_{CC}$ = $PLLV_{CC}$ = $AV_{CC}$ = 3.0 to 3.6 V, $V_{REFH}$ = 3.0 V to $AV_{CC}$, $V_{SS}$ = $PLLV_{SS}$ = $V_{REFL}$ = 0 V
Operating temperature range during programming/erasing:
$T_a$ = -20 to +85°C (regular specifications), $T_a$ = -40 to +85°C (wide-range specifications)

| Item | | Symbol | Min. | Typ. | Max. | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Programming time | 8 bytes | $t_{DP8}$ | — | 0.4 | 2 | ms | PCLK = 50-MHz operation |
| | 128 bytes | $t_{DP128}$ | — | 1 | 5 | ms | |
| Erasure time | 8 Kbytes | $t_{DE8K}$ | — | 300 | 900 | ms | PCLK = 50-MHz operation |
| Blank check time | 8 bytes | $t_{DBC8}$ | — | — | 30 | µs | PCLK = 50-MHz operation |
| | 8 Kbytes | $t_{DBC8K}$ | — | — | 2.5 | ms | |
| Rewrite/erase cycle*1 | | $N_{DPEC}$ | 30000*2 | — | — | Times | |
| Suspend delay time during writing | | $t_{DSPD}$ | — | — | 120 | µs | Figure 29.29 PCLK = 50-MHz operation |
| First suspend delay time during erasing (in suspend priority mode) | | $t_{DSESD1}$ | — | — | 120 | µs | |
| Second suspend delay time during erasing (in suspend priority mode) | | $t_{DSESD2}$ | — | — | 1.7 | ms | |
| Suspend delay time during erasing (in erasure priority mode) | | $t_{DSEED}$ | — | — | 1.7 | ms | |
| Data hold time*3 | | $T_{DDRP}$ | 10 | — | — | Year | |

Notes: 1. Definition of rewrite/erase cycle:
The rewrite/erase cycle is the number of erasing for each block. When the rewrite/erase cycle is n times (n = 30000), erasing can be performed n times for each block. For instance, when 128-byte writing is performed 64 times for different addresses in 8-Kbyte block and then the entire block is erased, the rewrite/erase cycle is counted as one. However, writing to the same address for several times as one erasing is not enabled (over writing is prohibited).
2. This indicates the minimum number that guarantees the characteristics after rewriting. (The guaranteed value is in the range from one to the minimum number.)
3. This indicates the characteristic when rewrite is performed within the specification range including the minimum number.



Downloaded from Arrow.com.

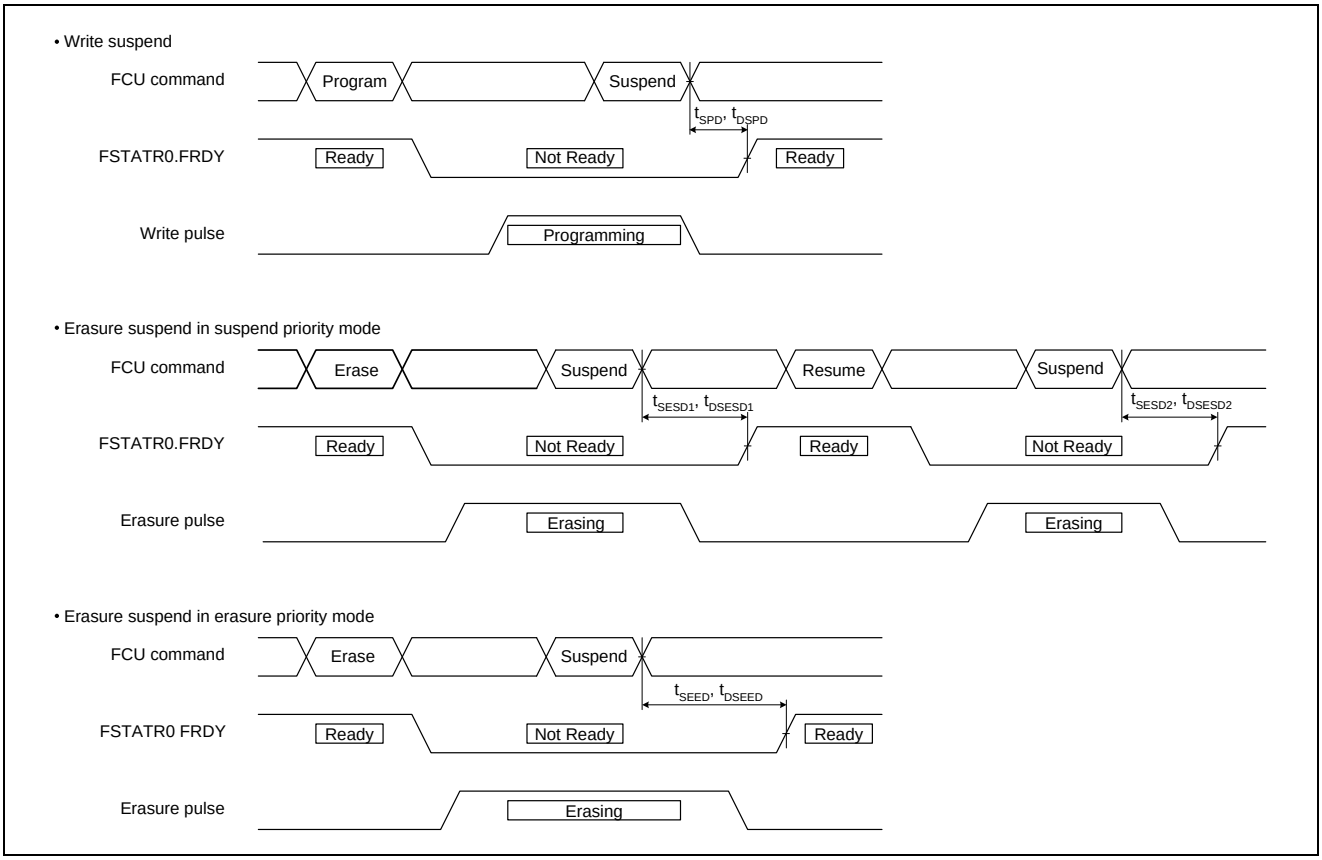



**Figure 29.29 ROM, Data Flash Write/Erase Suspend Timing**



Downloaded from Arrow.com.

# Appendix 1. Port States in Each Processing Mode

**Table 1.1 Port States in Each Processing State**

| Port Name / Pin Name | Operating Mode According to Registers Setting | Reset | Software Standby Mode | | Deep Software Standby Mode | After Deep Software Standby Mode is Canceled (Return of Start-up Mode) | |
|---|---|---|---|---|---|---|---|
| | | | OPE = 1 | OPE = 0 | IOKEE P = 1/0 | IOKEEP = 1* | IOKEEP = 0 |
| Port 0 | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| Port 1 | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| Port 2 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| P30 to P33 | All | HiZ | Keep-O*1 | | Keep-O*2 | Keep | HiZ |
| P34/PO12/TIOCA1/IRQ4-A | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| P35 to P37 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port 4 | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| P50/WR0/WR | Single-chip mode (EXBE = 0) | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | [WR0/WR output] H | [WR0/WR output] HiZ | | | |
| P51/WR1/BC1 | Single-chip mode (EXBE = 0) | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | [WR1/WR output] H | [WR1/WR output] HiZ | | | |
| | | | [Other than the above] Keep-O | | | | |
| P52/RD | Single-chip mode (EXBE = 0) | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | [RD output] H | [RD output] HiZ | | | |
| P53/BCLK | All | HiZ | [Clock output] H [Other than the above] Hi-Z | | Keep | Keep | HiZ |
| P54/TRDATA0 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| P55/TRDATA1 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| P56/TRDATA2 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| P57/TRDATA3/WAIT | All | HiZ | Keep-O | | Keep | Keep | HiZ |



Downloaded from Arrow.com.

| Port Name / Pin Name | Operating Mode According to Registers Setting | Reset | Software Standby Mode | | Deep Software Standby Mode | After Deep Software Standby Mode is Canceled (Return of Start-up Mode) | |
|---|---|---|---|---|---|---|---|
| | | | OPE = 1 | OPE = 0 | IOKEE P = 1/0 | IOKEEP = 1* | IOKEEP = 0 |
| P60/CS0/ CS4-A/CS5-B | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |
| P61/CS1/ CS2-B/CS5-A/ CS6-B/CS7-B | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |
| P62/CS2-A/ CS6-A | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |
| P63/CS3-A/ CS7-A | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |
| P64/CS4-B | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |
| P65/ RQ15-A | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| P66/DA0 | All | HiZ | [DAOE0=1] DA output retained [DAOE0=0] Keep-O | | [DAOE0=1] HiZ [DAOE0=0] Keep | Keep | HiZ |
| P67/DA1 | All | HiZ | [DAOE1=1] DA output retained [DAOE1=0] Keep-O | | [DAOE1=1] HiZ [DAOE1=0] Keep-O | Keep | HiZ |
| P70/CS3-B/ ADTRG2 | All | HiZ | [CS output] H [Other than the above] Keep-O | [CS output] HiZ [Other than the above] Keep-O | Keep | Keep | HiZ |



Downloaded from Arrow.com.

| Port Name<br>Pin Name | Operating Mode According to Registers Setting | Reset | Software Standby Mode | | Deep Software Standby Mode<br>IOKEE P = 1/0 | After Deep Software Standby Mode is Canceled (Return of Start-up Mode) | |
|---|---|---|---|---|---|---|---|
| | | | OPE = 1 | OPE = 0 | | IOKEEP = 1* | IOKEEP = 0 |
| P71/CS4-C/<br>CS5-C/CS6-C/<br>CS7-C | All | HiZ | [CS output]<br>H<br>[Other than the above]<br>Keep-O | [CS output]<br>HiZ<br>[Other than the above]<br>Keep-O | Keep | Keep | HiZ |
| P72 to P75 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| P76/ RQ14-A | All | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| P77 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port 8 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port 9 | All | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port A | Single-chip mode<br>(EXBE = 0) | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode<br>(EXBE = 1) | | [Address output]<br>Address output retained<br>[Other than the above]<br>HiZ | [Address output]<br>HiZ<br>[Other than the above]<br>HiZ | | | |
| Port B | Single-chip mode<br>(EXBE = 0) | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled extended mode | | [Address output]<br>Address output retained<br>[Other than the above]<br>Keep-O | [Address output]<br>HiZ<br>[Other than the above]<br>Keep-O | | | |



Downloaded from Arrow.com.

| Port Name / Pin Name | Operating Mode According to Registers Setting | | Reset | Software Standby Mode | | Deep Software Standby Mode IOKEE P = 1/0 | After Deep Software Standby Mode is Canceled (Return of Start-up Mode) | |
|---|---|---|---|---|---|---|---|---|
| | | | | OPE = 1 | OPE = 0 | | IOKEEP = 1* | IOKEEP = 0 |
| PC0 to PC4 | Single-chip mode (EXBE = 0) | | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | | [Address output] Address output retained [Other than the above] Keep-O | [Address output] HiZ [Other than the above] Keep-O | | | |
| PC5 to PC7 | Single-chip mode (EXBE = 0) | | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | | [Address output] Address output retained [CS output] H [Other than the above] Keep-O | [Address output] HiZ [CS output] HiZ [Other than the above] Keep-O | | | |
| Port D | Single-chip mode (EXBE = 0)) | | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/disabled extended mode (EXBE = 1) | | | HiZ | | | | |
| PE0 to PE4 | Single-chip mode (EXBE = 0) | | HiZ | Keep-O | | Keep | Keep | HiZ |
| | On-chip ROM enabled/ disabled extended mode (EXBE = 1) | 8-bit bus | | Keep-O | | | | |
| | | 16-bit bus | | HiZ | | | | |



Downloaded from Arrow.com.

| Port Name / Pin Name | Operating Mode According to Registers Setting | | Reset | Software Standby Mode | | Deep Software Standby Mode IOKEE P = 1/0 | After Deep Software Standby Mode is Canceled (Return of Start-up Mode) | |
|---|---|---|---|---|---|---|---|---|
| | | | | OPE = 1 | OPE = 0 | | IOKEEP = 1* | IOKEEP = 0 |
| PE5 to PE7 | Single-chip mode (EXBE = 0) | | HiZ | Keep-O*1 | | Keep | Keep | HiZ |
| | On-chip ROM enabled/ disabled extended mode (EXBE = 1) | 8-bit bus | | Keep-O*1 | | | | |
| | | 16-bit bus | | HiZ | | | | |
| Port F | All | | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port G | All | | HiZ | Keep-O | | Keep | Keep | HiZ |
| Port H | All | | HiZ | Keep-O | | Keep | Keep | HiZ |
| WDTOVF | All | | WDTOVF output | H | | H | H | |

[Legend]

H: High-level

L: Low-level

Keep-O: Output pins retain their previous values, and input pins become high-impedance.

*1. Input to pins in use as external interrupt pins and set up as triggers for release from software standby mode is possible.

*2. Input to pins set up as triggers for release from deep software standby mode is possible.

Keep: Pin states are retained during periods on software standby.

HiZ: High-impedance

IOKEEP = 1*: I/O pins retain their states until the IOKEEP bit in DPSBYCR is cleared to 0.

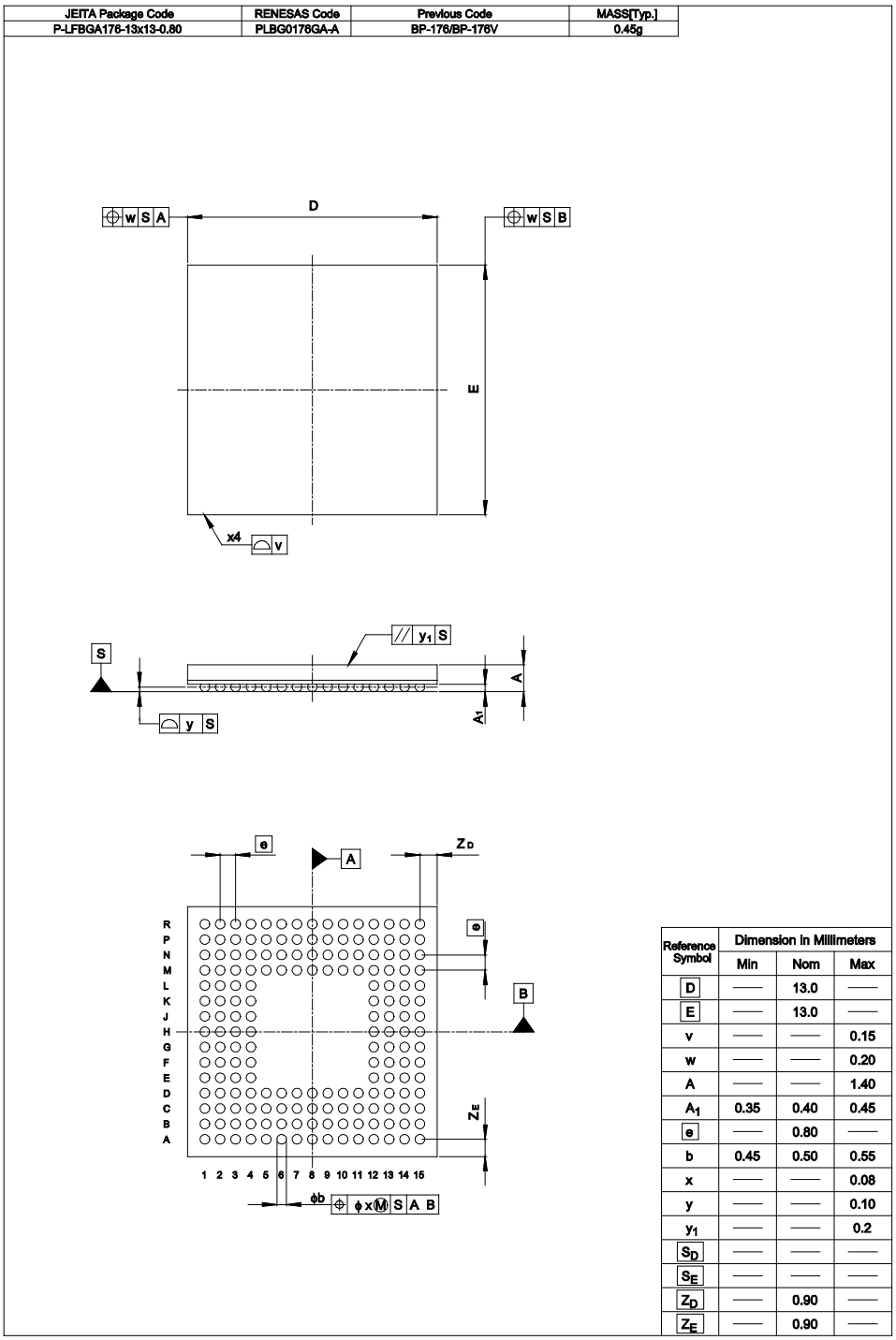


Downloaded from Arrow.com.

# Appendix 2. Package Dimensions

Information on the latest version of the package dimensions or mountings has been displayed in "Packages" on Renesas Technology Corp. website.

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LFBGA176-13x13-0.80 | PLBG0176GA-A | BP-176/BP-176V | 0.45g |



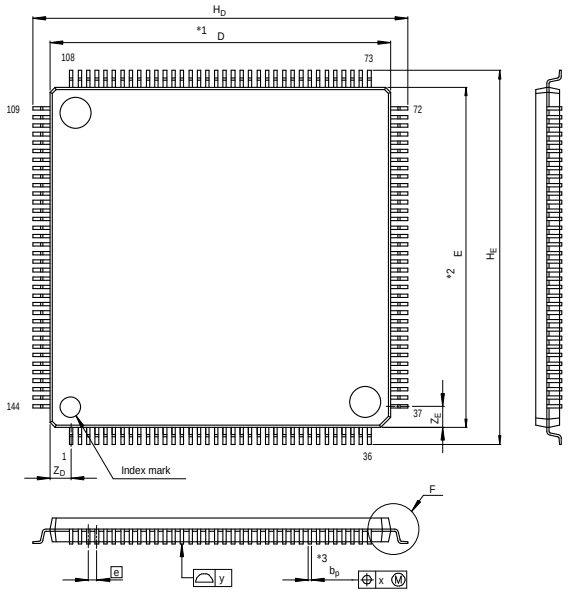

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | — | 13.0 | — |
| E | — | 13.0 | — |
| v | — | — | 0.15 |
| w | — | — | 0.20 |
| A | — | — | 1.40 |
| $A_1$ | 0.35 | 0.40 | 0.45 |
| e | — | 0.80 | — |
| b | 0.45 | 0.50 | 0.55 |
| x | — | — | 0.08 |
| y | — | — | 0.10 |
| $y_1$ | — | — | 0.2 |
| $S_D$ | — | — | — |
| $S_E$ | — | — | — |
| $Z_D$ | — | 0.90 | — |
| $Z_E$ | — | 0.90 | — |

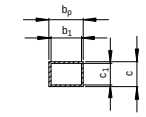
176-pin LFBGA (PLBG0176GA-A)



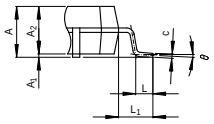
Downloaded from Arrow.com.

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LQFP144-20x20-0 50 | PLQP0144KA-A | 144P6Q-A / FP-144L / FP-144LV | 1.2g |




Terminal cross section




Detail F

NOTE)
1. DIMENSIONS " *1" AND " *2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION " *3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 19 9 | 20.0 | 20.1 |
| E | 19 9 | 20.0 | 20.1 |
| $A_2$ | — | 1.4 | — |
| $H_D$ | 21 8 | 22.0 | 22.2 |
| $H_E$ | 21 8 | 22.0 | 22.2 |
| A | — | — | 1.7 |
| $A_1$ | 0.05 | 0.1 | 0.15 |
| $b_p$ | 0.17 | 0.22 | 0.27 |
| $b_1$ | — | 0.20 | — |
| c | 0.09 | 0.145 | 0.20 |
| $c_1$ | — | 0.125 | — |
| θ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.08 |
| y | — | — | 0.10 |
| $Z_D$ | — | 1.25 | — |
| $Z_E$ | — | 1.25 | — |
| L | 0.35 | 0.5 | 0.65 |
| $L_1$ | — | 1.0 | — |

144-pin LQFP (PLQP0144KA-A)



Downloaded from Arrow.com.

# REVISION HISTORY RX610 Group Hardware Manual

| | | Description | |
|---|---|---|---|
| **Rev.** | **Data** | **Page** | **Summary** |
| 0.11 | Apr 15, 2009 | – | First edition issued |
| 0.12 | Aug 07, 2009 | | 1.Overview |
| | | 24 | Table 1.1 Outline of Specifications: Interrupt control unit, changed |
| | | 24 | Table 1.1 Outline of Specifications: Programmable I/O ports, changed |
| | | 27 | Figure 1.2 Block Diagram changed |
| | | | 2.CPU |
| | | 44 | 2.2.2.4 Processor Status Word (PSW): Note 1. changed |
| | | 50 | 2.3.3 Privileged Instruction: MVTIPL Instruction deleted |
| | | 64 | Table 2.13 Instructions that are Converted into a Single Micro-Operation: System manipulation instructions, "MVTIPL#IMM" deleted |
| | | | 5. I/O Registers |
| | | 83 to 103 | Table 5.1 List of I/O Registers (Address Order) changed |
| | | 104 to 131 | Table 5.2 List of I/O Registers (bit Order) changed |
| | | | 7. Clock Generation Circuit |
| | | 142 | Figure 7.2 Example of Crystal Resonator Connection changed |
| | | 142 | Table 7.4 Damping Resistance: Reference values added |
| | | 142 | Table 7.5 Crystal Resonator Characteristics: Reference values added |
| | | 143 | 7.3.2 External Clock Input: Parasitic capacitance added |
| | | | 8. Power-Down Modes |
| | | 163 | 8.2.8 Deep Standby Interrupt Flag Register (DPSIFR): Description added |
| | | 168 | 8.5.2.1 Transitions to All-Module Clock Stop Mode: Note 5. added |
| | | 170 | 8.5.3.1 Transition to Software Standby Mode: Note 3. added |
| | | 178 | Figure 8.4 Example of Flowchart to Use Deep Software Standby Mode changed |
| | | 180 | 8.7.5 Input Buffer Control by DIRQiE Bit (i = 3 to 0) changed |
| | | | 10. Interrupt Control Unit (ICU) |
| | | 193 | 10.1 Overview changed |
| | | 193 | Table 10.1 Specifications of the Interrupt Control Unit changed |
| | | 197 to 199 | Table 10.3 Registers of the Interrupt Control Unit changed |
| | | 203 | 10.2.1 Interrupt Request Register n (IRn) changed |
| | | 205 | 10.2.2 Interrupt Destination Setting Register n (ISELRn) changed |
| | | 206 | 10.2.3 Interrupt Request Enable Register m (IERm) changed |
| | | 207 | 10.2.4 Interrupt Priority Register m (IPRm) changed |
| | | 208 | 10.2.5 Fast Interrupt Register (FIR) changed |
| | | 209 | 10.2.6 IRQ Detection Enable Register i (IRQERi) changed |
| | | 210 | 10.2.7 IRQ Control Register i (IRQCRi) changed |
| | | 212 | 10.2.9 NMI Pin Interrupt Control Register (NMICR) changed |
| | | 215 | 10.2.12 Software Standby Release IRQ Enable Register (SSIER) changed |
| | | 216 | 10.3 Vector Table changed |
| | | 216 | 10.3.1 Interrupt Vector Table changed |
| | | 216 | Table 10.4 Interrupt Vector Table: In the Interrupt Source column, FCUIF is changed to FCU |
| | | 221 | 10.3.2 Fast Interrupt Vector Address changed |
| | | 221 | 10.3.3 Non-maskable Interrupt Vector Address changed |
| | | 222 | 10.4 Operation changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.12 | Aug 07, 2009 | 222 | 10.4.1 Enabling and Disabling Interrupts changed |
| | | 223 | 10.4.2 Interrupt Status Flag changed |
| | | 223 | 10.4.2.1 Interrupt Status Flag in Edge Detection changed |
| | | 224 | 10.4.2.2 Interrupt Status Flag in Level Detection changed |
| | | 225 | 10.4.3 Selecting Interrupt Request Destinations changed |
| | | 228 | 10.4.4 Determining Priority changed |
| | | 228 | 10.4.5 Fast Interrupt changed |
| | | 229 | 10.4.6 External Interrupts changed |
| | | 230 | 10.5 Non-maskable Interrupt changed |
| | | 231 | Table 10.5 List of Interrupt Sources changed |
| | | 231 | 10.6.1 Returning from Sleep Mode and All-Module Clock Stop Mode changed |
| | | 232 | 10.6.2 Returning from Software Standby Mode changed |
| | | | 11. Buses |
| | | | 11.3.5 CSi Wait Control Register 2 (CSiWCNT2) (i = 0 to 7): |
| | | 252 | Description on the WDOFF[2:0] bit and the WDON[2:0] bit changed |
| | | 265 | Figure 11.10 Example of Normal-Read Operation (when Two Rounds of Bus Access are Generated in Response to a Single Request for Transfer) changed |
| | | 265 | Figure 11.11 Example of Normal-Write Operation (when Two Rounds of Bus Access are Generated in Response to a Single Request for Transfer) changed |
| | | 275 | 11.5.5.4 Point for Caution Regarding Register Settings added |
| | | | 12. DMA Controller (DMAC) |
| | | 285 | Table 12.4 Setting of DCTG[5:0] Bits: DMA Request Source changed for 100111b, 101000b, 101001b, and 101010b on the DCTG[5:0] bit |
| | | | 13. Data Transfer Controller (DTC) |
| | | 338 | Figure 13.14 Procedure to Activate the DTC by an Interrupt changed |
| | | | 14. I/O Ports |
| | | 345 | Table 14.1 Specifications of I/O Ports changed |
| | | 353 | 14.2.2 Data Register (DR) Bit layout: Bit order changed Table of bits: Note changed |
| | | | 16. Programmable Pulse Generator (PPG) |
| | | 516 | Figure 16.11 Sample Setup Procedure for Non-Overlapping Pulse Output (PPG0 Setting) changed |
| | | 517 | Figure 16.12 Sample Setup Procedure for Non-Overlapping Pulse Output (PPG1 Setting) changed |
| | | 521 | 16.3.8 Pulse Output Triggered by Input Capture changed |
| | | | 20. Serial Communications Interface (SCI) |
| | | | 20.2.6 Serial Control Register (SCR) |
| | | 582 | (1) Serial Communications Interface Mode (SMIF in SCMR = 0) Description on the TEIE, RIE, and TIE bits changed |
| | | 584 | (2) Smart Card Interface Mode (SMIF in SCMR = 1) Description on the RIE and TIE bits change |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.12 | Aug 07, 2009 | | 20.2.7 Serial Status Register (SSR) |
| | | | (1) Serial Communications Interface Mode (SMIF in SCMR = 0) |
| | | 585 to 586 | Description on the TEND, PER, FER, and ORER flags changed |
| | | | (2) Smart Card Interface Mode (SMIF in SCMR = 1) |
| | | 587 to 588 | Description on the TEND, PER, and ORER flags changed |
| | | 604 | 20.3.4 SCI Initialization (Asynchronous Mode) changed |
| | | 604 | Figure 20.7 Sample SCI Initialization Flowchart (Asynchronous Mode) changed |
| | | 605 | 20.3.5 Serial Data Transmission (Asynchronous Mode) changed |
| | | 606 | Figure 20.9 Example of Serial Transmission Flowchart in Asynchronous Mode changed |
| | | 608 | 20.3.6 Serial Data Reception (Asynchronous Mode) changed |
| | | 609 | Figure 20.11 Example of Serial Reception Flowchart (1) (Asynchronous Mode) changed |
| | | 610 | Figure 20.11 Example of Serial Reception Flowchart (2) (Asynchronous Mode) changed |
| | | 612 | 20.4.2 SCI Initialization (Clock Synchronous Mode) changed |
| | | 612 | Figure 20.13 Example of SCI Initialization Flowchart (Clock Synchronous Mode) changed |
| | | 613 | 20.4.3 Serial Data Transmission (Clock Synchronous Mode) changed |
| | | 614 | Figure 20.15 Example of Serial Transmission Flowchart (Clock Synchronous Mode) changed |
| | | 615 | 20.4.4 Serial Data Reception (Clock Synchronous Mode) Changed |
| | | 616 | Figure 20.17 Example of Serial Reception Flowchart (Clock Synchronous Mode) changed |
| | | 617 | 20.4.5 Simultaneous Serial Data Transmission and Reception (Clock Synchronous Mode) changed |
| | | 618 | Figure 20.18 Example of Simultaneous Serial Transmission/Reception Flowchart (Clock Synchronous Mode) changed |
| | | 623 | 20.5.5 Initialization of the SCI changed |
| | | 627 | Figure 20.26 Sample Serial Transmission Flowchart changed |
| | | 629 | Figure 20.28 Sample Serial Reception Flowchart changed |
| | | 631 | 20.6.1 Interrupts in Serial Communications Interface Mode changed |
| | | 634 | 20.7.6 Restrictions on Clock Synchronous Transmission changed |
| | | | 22. I2C Bus Interface (RIIC) |
| | | | (Configuration in this section changed) |
| | | 647 | Figure 22.1 Block Diagram of RIIC changed |
| | | 649 | Table 22.3 Registers of the RII changed |
| | | 650 | 22.2.1 I2C Bus Control Register 1 (ICCR1) changed |
| | | 653 | 22.2.2 I2C Bus Control Register 2 (ICCR2) changed |
| | | 657 | 22.2.3 I2C Bus Mode Register 1 (ICMR1) changed |
| | | 659 | 22.2.4 I2C Bus Mode Register 2 (ICMR2) changed |
| | | 661 | 22.2.5 I2C Bus Mode Register 3 (ICMR3) changed |
| | | 664 | 22.2.6 I2C Bus Function Enable Register (ICFER) changed |
| | | 670 | 22.2.9 I2C Bus Status Register 1 (ICSR1) changed |
| | | 675 | 22.2.10 I2C Bus Status Register 2 (ICSR2) changed |
| | | 679 | 22.2.11 Slave Address Register L0 (SARL0) changed |
| | | 680 | 22.2.12 Slave Address Register U0 (SARU0) changed |
| | | 681 | 22.2.13 Slave Address Register L1 (SARL1) changed |
| | | 682 | 22.2.14 Slave Address Register U1 (SARU1) changed |
| | | 683 | 22.2.15 Slave Address Register L2 (SARL2) changed |
| | | 684 | 22.2.16 Slave Address Register U2 (SARU2) changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.12 | Aug 07, 2009 | 685 | 22.2.17 I2C Bus Bit Rate Low-Level Register (ICBRL) changed |
| | | 685 | 22.2.18 I2C Bus Bit Rate High-Level Register (ICBRH) changed |
| | | 686 | Table 22.6 Examples of ICBRH/ICBRL Settings for Transfer Rate changed |
| | | 687 | 22.2.20 I2C Bus Receive Data Register (ICDRR) changed |
| | | 687 | 22.2.21 I2C Bus Shift Register (ICDRS) changed |
| | | 688 to 704 | 22.3 Operation changed |
| | | 706 | 22.5 Facility for Delaying SDA Output added changed |
| | | 711 | 22.7.3 Device-ID Address Detection changed |
| | | 718 | 22.9.1 Master Arbitration Lost Detection (MALE Bit) changed |
| | | 720 | 22.9.2 Function to Detect Loss of Arbitration during NACK Transmission (NALE Bit) changed |
| | | 721 | 22.9.3 Slave Arbitration Lost Detection (SALE Bit) changed |
| | | 724 | 22.11.1 Timeout Function changed |
| | | 725 | Figure 22.38 Timeout Function (TMOE, TMOS, TMOH, and TMOL Bits) changed |
| | | 726 | 22.11.2 Extra SCL Clock Cycle Output Function changed |
| | | 727 | 22.11.3 RIIC Reset and Internal Reset changed |
| | | 728 | 22.12 SMBus Operation changed |
| | | 731 | 22.13 Interrupt Request changed |
| | | | 23. A/D Converter |
| | | 761 | 23.6.5 Permiss ble Impedance of Signal Sources changed |
| | | 762 | 23.6.7 Ranges of Settings for Analog Power Supply and Other Pins changed |
| | | 762 | Figure 23.16 Example of Connections for AVcc = Vcc and AVss = Vss changed |
| | | 762 | 23.6.9 Point for Caution Regarding Countermeasures for Noise changed |
| | | 763 | Figure 23.17 Example of a Protective Circuit for Analog Inputs changed |
| | | 763 | 23.6.10 Realizing High-Speed Conversion changed |
| | | 763 | Figure 23.18 Example of an Externally Connected Capacitor for High-Speed Conversion changed |
| | | | 26. ROM (Flash Memory for Code Storage)<br>(Configuration in this section changed) |
| | | 772 | Table 26.1 Specifications of the ROM changed |
| | | 773 | Figure 26.1 Block Diagram of ROM changed |
| | | 773 | Table 26.2 Input and Output Pins Associated with the ROM changed |
| | | 775 | 26.2.1 Flash Mode Register (FMODR)<br>Table of bits: Function on the FRDMD bit changed<br>Description on the FRDMD bit changed |
| | | 792 | 26.2.14 Peripheral Clock Notification Register (PCKAR) changed |
| | | 793 | 26.2.15 Flash Write Erase Protection Register (FWEPROR) changed |
| | | 794 | 26.3 Configuration of Memory Mats for the ROM added |
| | | 794 | 26.4 Block Configuration added |
| | | 795 | 26.5 Operating Modes Associated with the ROM changed |
| | | 795 | Figure 26.4 Transitions between Operating Modes in Terms of the ROM added |
| | | 796 | Table 26.5 Differences between Modes changed |
| | | 797 | 26.6 Programming and Erasing the ROM added |
| | | 797 | 26.6.1 FCU Modes added |
| | | 798 | 26.6.1.1 ROM Read Modes changed |
| | | 798 | 26.6.1.2 ROM P/E Modes added |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.12 | Aug 07, 2009 | 799 | 26.6.2 FCU Commands changed |
| | | 799 | Table 26.6 FCU Commands for Use with the ROM changed |
| | | 800 | Table 26.7 FCU Command Formats changed |
| | | 801 | 26.6.3 Connections between FCU Modes and Commands added |
| | | 802 | 26.6.4 FCU Command Usage changed |
| | | 802 | 26.6.4.1 Mode Transitions changed |
| | | 806 | 26.6.4.2 Programming and Erasure changed |
| | | 815 | 26.6.4.3 Error Processing added |
| | | 816 | 26.6.4.4 Suspension and Resumption added |
| | | 824 | 26.10 Boot Mode added |
| | | 824 | 26.10.1 System Configuration added |
| | | 825 | 26.10.2 State Transitions in Boot Mode added |
| | | 827 | 26.10.3 Automatic Adjustment of the Bit Rate added |
| | | 828 | 26.10.4 ID Code Protection in Boot Mode added |
| | | 829 | 26.11 ID Code Protection on Connection of the On-Chip Debugger added |
| | | 830 | 26.12 ROM Code Protection added |
| | | 831 | 26.13 Usage Notes changed |
| | | | 27. Data Flash Memory (Flash Memory for Data Storage)<br>(Configuration in this section changed) |
| | | 832 | Table 27.1 Specifications of Data Flash Memory added |
| | | 833 | Figure 27.1 Block Diagram of Data Flash Memory changed |
| | | 833 | Table 27.2 Input and Output Pins Associated with the Data Flash changed |
| | | 835 | 27.2.1 Flash Mode Register (FMODR)<br>Table of bits: Function on the FRDMD bit changed<br>Description on the FRDMD bit changed |
| | | 846 | 27.3 Configuration of Memory Mat for the Data Flash Memory added |
| | | 846 | 27.4 Block Configuration added |
| | | 847 | 27.5 Operating Modes Associated with the Data Flash added |
| | | 847 | Table 27.4 Differences between Modes changed |
| | | 848 | 27.6 Programming and Erasing the Data Flash Memory added |
| | | 848 | 27.6.1 FCU Modes added |
| | | 849 | 27.6.1.1 ROM P/E Modes changed |
| | | 849 | 27.6.1.2 ROM/Data Flash Read Mode changed |
| | | 849 | 27.6.1.3 Data Flash P/E Modes changed |
| | | 850 | 27.6.2 FCU Commands changed |
| | | 850 | Table 27.5 FCU Commands for Use with Data Flash Memory changed |
| | | 850 | Table 27.6 FCU Command Formats changed |
| | | 851 | 27.6.3 Connections between FCU Modes and Commands changed |
| | | 851 | Table 27.7 Acceptable Commands and the State and Mode (Data Flash P/E Mode) of the FCU |
| | | 852 | 27.6.4 FCU Command Usage changed |
| | | 856 | 27.7.1 Software Protection changed |
| | | 857 | 27.7.2 Error Protection changed |
| | | 858 | 27.8 Usage Notes changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.12 | Aug 07, 2009 | | 28. Electrical Characteristics |
| | | 861 | Table 28.2 DC Characteristics (1): Note 4 added |
| | | 868 | Table 28.9 Timing of On-Chip Peripheral Modules (1) changed |
| | | 869 | Table 28.9 Timing of On-Chip Peripheral Modules (2) changed |
| | | 870 | Table 28.9 Timing of On-Chip Peripheral Modules (3) changed |
| | | | Appendix 1.Port States in Each Processing Mode |
| | | 878 to 881 | Table 1.1Port States in Each Processing State (by Activation Mode) changed |
| 0.40 | Dec. 16, 2009 | All | Register description (reserved bit) reviewed |
| | | | "General Precautions in the Handling of MPU/MCU Products" added |
| | | | "How to Use This Manual" added |
| | | | Section 1 Overview |
| | | 27 to 29 | Table 1.1 Outline of Specifications, changed |
| | | | Reference power supply pin for the A/D and D/A converters: Vref (pin no. 142) → VREFH, |
| | | | Reference ground pin: AVSS (pin no. 140) → VREFL, Pin name changed |
| | | 32 | Figure 1.3 Pin Assignment of the 144-Pin LQFP, changed |
| | | 33 | Figure 1.4 Pin Assignment (Assistance Diagram) of the 144-Pin LQFP, changed |
| | | 38 | Table 1.3 List of Pins and Pin Functions, changed |
| | | 43 | Table 1.4 Pin Functions, changed |
| | | | Section 2 CPU |
| | | 44 | 2.1 Features: Register set of the CPU, Accumulator, changed |
| | | 45 | Figure 2.1 Register Set of the CPU, changed |
| | | | 2.2.2.8 Floating-Point Status Word (FPSW) |
| | | 51 to 54 | RM[1:0], Floating-Point Rounding-Mode Setting: Bit name changed |
| | | | Bit description added |
| | | 55 | 2.2.3 Accumulator (ACC), changed |
| | | 59 | 2.5.1 Switching the Endian, changed |
| | | 65 | Figure 2.8 Fixed Vector Table, changed |
| | | 68 | 2.8 Pipeline, 2.8.1 Overview (4), changed |
| | | 74 | Figure 2.15 MOV Instruction (Memory-Memory), Bit Manipulation Instruction (Memory |
| | | | Source Operand), changed |
| | | 78 | 2.8.4 Interrupt Response Cycles, added |
| | | | Section 3 Operating Modes |
| | | 82, 83 | 3.2.3 System Control Register 0 (SYSCR0), bits ROME and KEY[7:0]: Bit description |
| | | | changed |
| | | 84 | 3.2.4 System Control Register 1 (SYSCR1), bit RAME: Bit description changed |
| | | | Section 4 Address Space |
| | | 88 | Figure 4.1 Memory Map of the R5F56108, added |
| | | 89 | Figure 4.2 Memory Map of the R5F56107, added |
| | | 90 | Figure 4.3 Memory Map of the R5F56106, added |
| | | 91 | Figure 4.4 Memory Map of the R5F56105, added |
| | | | Section 5 I/O Registers |
| | | 93, 94 | 3. Notes on writing to I/O registers, added |
| | | 116 to 143 | Table 5.2 List of I/O Registers (Bit Order), changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.40 | Dec 16, 2009 | | Section 8 Low Power Consumption (section title changed) |
| | | 160 | Table 8.1 States of Operation, Note 7. added |
| | | 164, 165 | 8.2.1 Standby Control Register (SBYCR), bit SSBY: Bit description changed |
| | | 172 | 8.2.5 Deep Standby Control Register (DPSBYCR), bit DPSBY: Bit description changed |
| | | 175 | 8.2.8 Deep Standby Interrupt Flag Register (DPSIFR), Register description changed |
| | | 181 | 8.5.2.1 Transitions to All-Module Clock Stop Mode, changed |
| | | 184 | 8.5.3.2 Canceling Software Standby Mode, added |
| | | 187 | 8.5.4.1 Transition to Deep Software Standby Mode, Note 2. added |
| | | 190 | 8.5.4.5 Example of Deep Software Standby Mode Application, changed |
| | | 191 | Figure 8.4 Example of Flowchart to Use Deep Software Standby Mode, changed |
| | | 193 | 8.7.7 Timing of Wait Instructions, changed |
| | | | Section 9 Exceptions |
| | | 199 | Table 9.2 Vector and Site for Saving the Values in the PC and PSW Registers, changed |
| | | 200 | 9.4 Hardware Processing for Accepting and Returning from Exceptions, added |
| | | 200 | (b) Updating of the PM, U, and I bits in the PSW register, changed |
| | | 203 | 9.6 Return from Exception Processing Routines, changed |
| | | | Section 10 Interrupt Control Unit (ICU) |
| | | 217 | 10.2.2 Interrupt Request Destination Setting Register n (ISELRn), Bit description list: Notes 1. and 2. added |
| | | 219 | 10.2.4 Interrupt Priority Register m (IPRm), bits IPR[2:0]: Bit name changed |
| | | 228 to 233 | Table 10.4 Interrupt Vector Table, changed |
| | | 244 | 10.6.2 Returning from Software Standby Mode, added |
| | | | Section 11 Buses |
| | | 251 | Table 11.6 Registers of the External Bus Controller, changed |
| | | | 11.3.1 CSi Control Register (CSiCNT): |
| | | 253 | Bit allocation: Value of CS0CNT after a reset, changed, Note deleted, Bit description list: Note 2. changed |
| | | 254 | Bits BSIZE[1:0]: Bit description changed |
| | | 257 | 11.3.3 CSi Mode Register (CSiMOD), bit PRMOD: Bit description changed |
| | | | 11.3.4 CSi Wait Control Register 1 (CSiWCNT1): |
| | | 259 to 262 | Bits CSPWWAIT[2:0] , CSPRWAIT[2:0] , CSWWAIT[4:0] , and CSRWAIT[4:0] : Note added |
| | | | 11.3.5 CSi Wait Control Register 2 (CSiWCNT2) : |
| | | 263 to 266 | Bits CSWOFF[2:0] , WDOFF[2:0] , RDON[2:0] , and WRON[2:0] : Note added |
| | | 287 | 11.5.3 Insertion of Recovery Cycles, changed |
| | | 287 | Figure 11.21 Example of Recovery Timing, changed |
| | | 290 | 11.6.2 Operations When a Bus Error Occurs, added |
| | | | Section 12 DMA Controller (DMAC) |
| | | 299 | Table 12.4 Setting of DCTG[5:0] Bits, changed |
| | | 313 | 12.2.16 DMA Transfer End Detect Register (DMEDET), flag DEDETn: Bit description added |
| | | 314 | 12.3.1 Bus Mastership Release Timing, corrected |
| | | 317 | Figure 12.4 Register Setting Procedure, changed |
| | | 322 | 12.4 Interrupts, changed, Figure 12.8 Schematic Logic Diagram of Interrupt Outputs, added |
| | | 324 | 12.6.1 Register Settings, (7) added |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.40 | Dec 16, 2009 | | Section 13 Data Transfer Controller (DTC) |
| | | 338 | Table 13.3 Correspondence between Interrupt Sources, DTC Vector Addresses, and the ISELRn Register of the ICU, changed |
| | | 349 | 13.4.6 Chain Transfer, changed |
| | | 349 | Figure 13.9 Chain Transfer Operation, changed |
| | | 353 | 13.5 DTC Setting Procedure, changed |
| | | 353 | Figure 13.14 Procedure to Set the DTC, changed |
| | | 358 | 13.8.1 Setting the DTC Module Start Register, Description on (1) and (2), changed |
| | | | Section 14 I/O Ports |
| | | 379 to 381 | 14.2.13 Port Function Control Register 6 (PFCR6), changed |
| | | | Table 14.6 Correspondences between PFCR6 and TPUn.TMDR Settings, Input Capture Inputs, and External Pins, added |
| | | 382 to 384 | 14.2.14 Port Function Control Register 7 (PFCR7), changed |
| | | | Table 14.7 Correspondences between PFCR7 and TPUn.TMDR Settings and Input Capture Inputs and External Pins, added |
| | | 387 | 14.3 Settings of Ports, changed |
| | | 429 | Table 14.10 Treatment of Unused Pins, changed |
| | | | Section 15 16-Bit Timer Pulse Unit (TPU) |
| | | 439 to 441 | Table 15.5 Registers of TPU, changed |
| | | 460 | 15.2.5 Timer Status Register (TSR), Bit allocation: Value after a reset, changed |
| | | 462 | 15.2.8 Timer Start Register (TSTRA, TSTRB), Bit allocation: TSTRB address, changed |
| | | 463 | 15.2.9 Timer Synchronous Register (TSYRA, TSYRB), Bit allocation: TSYRB address, changed |
| | | | Section 17 8-Bit Timer (TMR) |
| | | 543 | Table 17.3 Registers of TMR, changed |
| | | 549 | 17.2.6 Timer Control/Status Register (TCSR), Bit allocation: Value after a reset, changed |
| | | | Section 18 Compare Match Timer (CMT) |
| | | 566 | Table 18.2 List of CMT Registers, changed |
| | | 569 | 18.2.3 Compare Match Timer Control Register (CMCR), Bit allocation: Value after a reset, changed |
| | | | Section 19 Watchdog Timer (WDT) |
| | | 576 | Table 19.3 WDT Registers, changed |
| | | 577 | 19.2.2 Timer Control/Status Register (TCSR), Bit allocation: Value after a reset, changed |
| | | | Section 20 Serial Communications Interface (SCI) |
| | | 589 to 590 | Table 20.4 Registers of SCI, changed |
| | | 601, 603 | 20.2.7 Serial Status Register (SSR), Bit allocation: Value after a reset, changed |
| | | 602 | Flag PER, corrected |
| | | 647 | 20.6.1 Interrupts in Serial Communications Interface Mode, changed |
| | | | Section 22 $I^2C$ Bus Interface (RIIC) |
| | | 661 to 745 | RIIC interrupt names changed (EEI → ICEEI, RXI → ICRXI, TXI → ICTXI, TEI → ICTEI) |
| | | 663 | Figure 22.1 Block Diagram of RIIC, changed |
| | | 664 | Figure 22.2 Connections to the External Circuit by the I/O Pins ($I^2C$ Bus Configuration Example), changed |
| | | 665 | Table 22.3 Registers of the RIIC, changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.40 | Dec 16, 2009 | | 22.2.2 I²C Bus Control Register 2 (ICCR2): |
| | | 669 | Bit allocation: Value after a reset, changed |
| | | 669 to 672 | Bit description changed |
| | | 673 | 22.2.3 I²C Bus Mode Register 1 (ICMR1), bits BC[2:0]: Bit description changed |
| | | 678 | 22.2.5 I²C Bus Mode Register 3 (ICMR3), bits ACKBR and ACKBT: Bit description changed |
| | | | 22.2.7 I²C Bus Status Enable Register (ICSER): |
| | | 682 | Bits SAR0E, SAR1E, and SAR2E → Bits SARmE (m = 0 to 2), Bit description changed |
| | | 687 | 22.2.9 I²C Bus Status Register 1 (ICSR1), Bit description list: Note added |
| | | 687 | Flags AAS0, AAS1, and AAS2 → Flag AASm (m = 0 to 2), Bit description changed |
| | | 687 to 688 | Bit description changed |
| | | 690 | 22.2.10 I²C Bus Status Register 2 (ICSR2), Bit description list: Note added |
| | | 690 to 693 | Bit description changed |
| | | 694 | (Registers SARL0, SARL1, and SARL2 → Register SARLm (m = 0 to 2), Register description changed) |
| | | | 22.2.11 Slave Address Register Lm (SARLm) (m = 0 to 2) |
| | | 695 | (Registers SARU0, SARU1, and SARU2 → Register SARUm (m = 0 to 2), Register description changed) |
| | | | 22.2.12 Slave Address Register Um (SARUm) (m = 0 to 2) |
| | | 696 | 22.2.13 I²C Bus Bit Rate Low-Level Register (ICBRL), Register description changed |
| | | | 22.2.14 I²C Bus Bit Rate High-Level Register (ICBRH) |
| | | 697 | Transfer rate expression, changed |
| | | 698 | Table 22.6 Examples of ICBRH/ICBRL Settings for Transfer Rate, changed |
| | | 701 | Figure 22.5 Example of RIIC Initialization Flow, changed |
| | | 702 | 22.3.3 Master Transmitter Operation, changed |
| | | 703 | Figure 22.6 Example of Master Transmission Flowchart, changed |
| | | 706 to 707 | 22.3.4 Master Receiver Operation, changed |
| | | 708 | Figure 22.10 Example of Master Reception Flowchart (7-Bit Address Format), changed |
| | | 711 | 22.3.5 Slave Transmitter Operation, changed |
| | | 712 | Figure 22.14 Example of Slave Transmission Flowchart, changed |
| | | 714 | 22.3.6 Slave Receiver Operation, changed |
| | | 723 | 22.7.3 Device-ID Address Detection, changed |
| | | 727 | Figure 22.30 Suspension of Data Transfer when NACK is Received (NACKE = 1), changed |
| | | 736 | 22.11.1 Timeout Function, changed |
| | | 743 | 22.13 Interrupt Request, Notes on interrupt processing 4., changed |
| | | 743 | Table 22.7 Interrupt Sources, changed |
| | | 745 | 22.15 Usage Notes, changed |
| | | | Section 23 A/D Converter<br>Reference power supply pin: Vref → VREFH, Reference ground pin: AVSS → VREFL, Pin name changed |
| | | 748 to 751 | Figures 23.1 to 23.4, Block Diagram of A/D Converter, changed |
| | | 752 | Table 23.3 Input Pins of A/D Converter, changed |
| | | 753 | Table 23.4 Registers of A/D Converter, changed |
| | | 755 | 23.2.2 A/D Control/Status Register (ADCSR), Bit allocation: Value after a reset, changed |
| | | 768 | Figure 23.10 Connections between Compare-Match/Input-Capture A to D Signals from TPU0 and the Respective Converter Units, changed |
| | | 769 | Figure 23.11 Connections between Compare-Match/Input-Capture A Signals and the Respective Converter Units, changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.40 | Dec 16, 2009 | 770 | Figure 23.12 Connections between Compare-Match A Signals from TMR0 and TMR2 and the Respective Converter Units, changed |
| | | 775 | 23.6.7 Ranges of Settings for Analog Power Supply and Other Pins, changed |
| | | 775 | Figure 23.16 Example of Connections for AVcc = Vcc and AVss = Vss = VREFL, changed |
| | | 775 | 23.6.8 Point for Caution Regarding Board Design, changed |
| | | 776 | 23.6.9 Point for Caution Regarding Countermeasures for Noise, changed |
| | | 776 | Figure 23.17 Example of a Protective Circuit for Analog Inputs, changed |
| | | 777 | 23.6.10 Realizing High-Speed Conversion, changed |
| | | 777 | Figure 23.18 Example of an Externally Connected Capacitor for High-Speed Conversion, changed |
| | | | Section 24 D/A Converter<br>Reference power supply pin: Vref → VREFH, Reference ground pin: AVSS → VREFL, Pin name changed |
| | | 778 | Figure 24.1 Block Diagram of D/A Converter, changed |
| | | 779 | Table 24.2 Pin Configuration of D/A Converter, changed |
| | | 781 | 24.2.2 D/A Control Register (DACR), Bit description list: Note 2. changed |
| | | | Section 26 ROM (Flash Memory for Code Storage)<br>User mat: 2 Mbytes → 2 Mbytes, 1.5 Mbytes, 1 Mbyte, or 768 Kbytes, changed<br>Flash interface error name: FCUERR → FIFERR, changed |
| | | 787, 788 | Table 26.1 Specifications of the ROM, changed and Note added |
| | | 789 | Figure 26.1 Block Diagram of ROM, changed |
| | | 790 | Table 26.3 Registers Related to ROM, changed |
| | | 793 | 26.2.2 Flash Access Status Register (FASTAT), bit ROMAE: Bit description (Note) added |
| | | 799 | 26.2.6 Flash Status Register 1 (FSTATR1), Bit allocation: Value after a reset |
| | | 800 | 26.2.7 Flash Ready Interrupt Enable Register (FRDYIE), Bit FRDYIE, Bit description added |
| | | | 26.2.8 Flash P/E Mode Entry Register (FENTRYR): |
| | | 801 | Bit description list (FENTRY1, FENTRY0), changed |
| | | 801 | Register description, changed |
| | | 802 | Bit FENTRY1: Bit description (Note) added |
| | | 803 | Table 26.4 Correspondence of FENTRY1 and FENTRY0 Bits in Each Product, added |
| | | 811 | Figure 26.2 Memory Mat Configuration of ROM, changed |
| | | 812 | Figure 26.3 Configuration of Erasure Blocks for the User Mat, changed |
| | | | 26.6.1.2 ROM P/E Modes |
| | | 816 | (1) ROM P/E Normal Mode, Note added |
| | | 816 | (2) ROM Status Read Mode, Note added |
| | | 817 | (3) ROM Lock-Bit Read Mode, Note added |
| | | 818 | Table 26.8 FCU Command Formats, changed |
| | | | 26.6.4.2 Programming and Erasure Procedures |
| | | 825 | (2) Jumping to Locations in On-chip RAM, Note added |
| | | 826 | (4) Using the Peripheral Clock Notification Command, Note added |
| | | 828 | (5) Programming, Note added |
| | | | 26.8.1 Software Protection |
| | | 840 | (2) Protection through FENTRYR, Note added |
| | | 840 | 26.8.2 Error Protection, changed |
| | | 841 | Table 26.10 Error Protection Types (Types Dedicated to ROM and Types Common to ROM and Data Flash), changed |
| | | 842 | 26.9 User Boot Mode, changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 0.40 | Dec 16, 2009 | 842 | 26.10 Boot Mode, 26.10.1 System Configuration, changed |
| | | 842 | Figure 26.24 System Configuration for Operations in Boot Mode, changed |
| | | 843 to 844 | 26.10.2 ID Code Protection, changed |
| | | | 26.10.3 State Transitions in Boot Mode |
| | | 845 | Figure 26.26 State Transitions in Boot Mode, changed |
| | | 846 | (2) Waiting for a Host Command for Inquiry or Selection, (3) Judging ID Code Protection, |
| | | 846 | (4) Waiting for an ID Code, changed |
| | | 848 to 861 | 26.10.5 Inquiry/Selection Host Command Wait State, added |
| | | 862 | 26.10.6 ID Code Wait State, added |
| | | 863 to 871 | 26.10.7 Programming/Erasure Host Command Wait State, added |
| | | 872 | 26.11 ID Code Protection on Connection of the On-Chip Debugger, changed |
| | | | Section 27 Data Flash (Flash Memory for Data Storage) |
| | | | Flash interface error name: FCUERR → FIFERR, changed |
| | | 876 | Figure 27.1 Block Diagram of Data Flash Memory, changed |
| | | 885 | 27.2.6 Flash P/E Mode Entry Register (FENTRYR), Register description, changed |
| | | 900 | 27.7.2 Error Protection, changed |
| | | 900 | Table 27.8 Error Protection Types (for Data Flash Only), changed |
| | | 901 to 904 | 27.8 Boot Mode, added |
| | | | Section 28 Electrical Characteristics |
| | | 905 to 925 | Vref → VREFH, AVSS → VREFL, changed |
| | | 909 | Table 28.5 Clock Timing, (tcyc, tCH, tCL) changed |
| 1.00 | Mar. 16, 2010 | | 1. Overview |
| | | 31 | Figure 1.2 Block Diagram: Ports F to H added |
| | | 32 | Figure 1.3 Pin Assignment of the 176-pin LFBGA, added |
| | | 35 to 40 | Table 1.3 List of Pins and Pin Functions (176-Pin LFBGA), added |
| | | | Table 1.5 Pin Functions: |
| | | 46, 50 | Description on the BSCANP, PF0 to PF6, PG0 to PG7, and PH0 to PH7 pins added |
| | | | 2. CPU |
| | | 55 | 2.2.2.4 Processor Status Word (PSW): Note 4 added to the IPL[2:0] bits |
| | | 78 | Table 2.14 Instructions that are Converted into Multiple Micro-Operations: Instructions DIV and DIVU added |
| | | 84 | 2.8.4 Numbers of Cycles for Response to Interrupts, changed |
| | | | 5. I/O Registers |
| | | 100 | 4.Number of Access Cycles to I/O Registers, added |
| | | | Table 5.1 List of I/O Registers (Address Order): |
| | | 101 to 121 | Column for number of access cycles, and notes 7 and 8, added |
| | | | 6. Resets |
| | | 156 | 6.5 Usage Notes, added |
| | | | 7. Clock Generation Circuit |
| | | 165 | 7.7.3 Notes on Board Design, changed |
| | | | 8. Low Power Consumption |
| | | 189 | 8.5.3.1 Transition to Software Standby Mode: Note 3 changed |
| | | 190 | 8.5.3.2 Canceling Software Standby Mode: |
| | | 192 | Description on 1. Canceling by an interrupt, added |
| | | 196 | Figure 8.2 Example of Software Standby Mode Application, changed |
| | | | Figure 8.3 Example of Deep Software Standby Mode Application, changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 1.00 | Mar 16, 2010 | | 10. Interrupt Control Unit (ICU) |
| | | 249 | 10.6.2 Returning from Software Standby Mode, changed |
| | | 250 | 10.7 Usage Notes, added |
| | | | 14. I/O Ports |
| | | 362 | Table 14.1 Specifications of I/O Ports: 176-pin LFBGA added |
| | | 367 | Table 14.2 Port Functions: Description on ports F to H, added |
| | | | Table 14.3 Registers for Each Port: |
| | | 369 | Registers DDR, DR, PORT, and ICR for PF, PG, and PH, added |
| | | 371 | 14.2.1 Data Direction Register (DDR): |
| | | | Description on PF.DDR, PG.DDR, and PH.DDR added |
| | | 372 | 14.2.2 Data Register (DR): Description on PF.DR, PG.DR, and PH.DR, added |
| | | 373 | 14.2.3 Port Register (PORT): Description on PF.PORT, PG.PORT, and PH.PORT, added |
| | | 374 | 14.2.4 Input Buffer Control Register (ICR): |
| | | | Description on PF.ICR, PG.ICR, and PH.ICR, added |
| | | 425 to 426 | 14.3.16 Port F (PF), added |
| | | 427 to 428 | 14.3.17 Port G (PG), added |
| | | 429 to 430 | 14.3.18 Port H (PH), added |
| | | 438 | Table 14.9 Pin Functions in Each Operating Mode: Description on PF, PG, and PH, added |
| | | 439 | Table 14.10 Treatment of Unused Pins, added |
| | | 440 | 14.6.3 Port Setting when A/D Converter Input is Used, added |
| | | | 22. $I^2C$ Bus Interface (RIIC) |
| | | 757 | 22.15.5 Notes when Communication is Restarted with the NACK Reception in Master Mode, added |
| | | 757 | 22.15.6 Notes on the RDRF Flag Set Timing Selection Bit (RDRFS), added |
| | | | 23. A/D Converter |
| | | 787 | 23.6.7 Ranges of Settings for Analog Power Supply and Other Pins, changed |
| | | 787 | Figure 23.16 Example of Connection for Power-Supply Pins, changed |
| | | | 26. ROM (Flash Memory for Code Storage) |
| | | 813 | 26.2.8 Flash P/E Mode Entry Register (FENTRYR): Note 1 added |
| | | | Table 26.8 FCU Command Formats: |
| | | 830 | Command name amended (peripheral clock notification) |
| | | 833 | Figure 26.7 Procedure for Transition to ROM Read Mode: |
| | | | Setting for the FRDY bit check, amended |
| | | 839 | Figure 26.13 Using the Peripheral Clock Notification Command: Timeout (tPCKA) added |
| | | | Table 26.12 Conditions for Automatic Bit-Rate Adjustment to be Possible: |
| | | 860 | Range of frequency for the EXTAL signal, changed |
| | | | 27. Data Flash (Flash Memory for Data Storage) |
| | | 912 | Table 27.8 Error Protection Types (for Data Flash Only): |
| | | | Description on the data flash programming protect error, changed |
| | | 917 to 933 | 28. Boundary Scan, added |



Downloaded from Arrow.com.

| Rev. | Data | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Mar 16, 2010 | | 29. Electrical Characteristics |
| | | 938 | Table 29.5 Clock Timing: Oscillation settling time after leaving deep software standby mode (crystal), $t_{OSC3}$, added |
| | | 939 | Figure 29.2 Oscillation Settling Timing after Software Standby Mode, changed |
| | | 940 | Figure 29.3 Oscillation Settling Timing after Deep Software Standby Mode, added |
| | | 950 | Table 29.8 Timing of On-Chip Peripheral Modules (3): Boundary scan added |
| | | 954 | Figure 29.26 Boundary Scan TCK Timing, added |
| | | 954 | Figure 29.27 Boundary Scan TRST# Timing, added |
| | | 955 | Figure 29.28 Boundary Scan Input/Output Timing, added |
| 1.10 | Apr 05, 2011 | | 1. Overview |
| | | 37 | Table 1.3 List of Pins and Pin Functions (176-Pin LFBGA), Pin no. P11: Pin name changed |
| | | | 2. CPU |
| | | 72 | 2.5.5 Notes on Arrangement of Instruction Code, added |
| | | 82 | Table 2.15 Numbers of Cycles for Response to Interrupts, changed |
| | | | 3. Operating Modes |
| | | 87 | Table 3.2 Selection of Operating Modes by Register Setting, changed |
| | | 95 | Figure 3.2 Setting of Bits ROME and EXBE and Operating Modes, changed |
| | | | 5. I/O Registers |
| | | 101 | At the beginning of the section, 1. I/O register addresses (address order), changed |
| | | 110 | Table 5.1 List of I/O Registers (Address Order): ISELR253 register, added |
| | | 135 | Table 5.2 List of I/O Registers (Bit Order): ISELR253 register, added |
| | | 144 to 146 | Table 5.2 List of I/O Registers (Bit Order): Bit name in SSR register, changed |
| | | | 8. Low Power Consumption |
| | | 170 | Figure 8.1 Mode Transitions, changed |
| | | 175 | 8.2.2 Module Stop Control Register A (MSTPCRA): Description added |
| | | 183 | 8.2.7 Deep Standby Interrupt Enable Register (DPSIER): Description added |
| | | 196 | 8.5.4.2 Canceling Deep Software Standby Mode: Description added |
| | | | 9. Exceptions |
| | | 204 | Figure 9.2 Outline of the Exception Handling Procedure, changed |
| | | | 10. Interrupt Control Unit (ICU) |
| | | 222 | 10.2.1 Interrupt Request Register i (IRi): Note and description, changed, added |
| | | 224 | 10.2.2 Interrupt Request Destination Setting Register i (ISELRi): Addresses changed |
| | | 225 | 10.2.3 Interrupt Request Enable Register i (IERi): Description changed, added |
| | | 226 | 10.2.4 Interrupt Priority Register i (IPRi): Description added |
| | | 227 | 10.2.5 Fast Interrupt Register (FIR): Description changed, added |
| | | 229 | 10.2.7 IRQ Control Register n (IRQCRn): Description added |
| | | 232 | 10.2.10 Non-maskable Interrupt Status Register (NMISR): Description added |
| | | 243 | 10.4.2.1 Interrupt Status Flag in Edge Detection: Description added |
| | | 242 | Table 10.4 Interrupt Vector Table, changed |
| | | 252 | 10.7.3 Notes on Transferring DMAC/DTC Using Communication Function (SCI, RIIC), added |
| | | 253 | (1) Software Preventive Measures, added |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 1.10 | Apr 05, 2011 | | 11. Buses |
| | | 258 | Table 11.5 Pin Configuration of the External Bus: Note added |
| | | 266 | 11.3.3 CSi Mode Register (CSiMOD): Description on bit WRMOD, changed |
| | | 266 | Table 11.8 Control Signals for Write Access Mode, changed |
| | | 296 | 11.5.5.3 Restrictions in Relation to RMPA and String-Manipulation Instructions: Title and description, changed |
| | | 297 | 11.5.5.5 Restriction on Instruction Code, added |
| | | | 12. DMA Controller (DMAC) |
| | | 328 | Figure 12.5 Timing Chart of the DREQ Bit in DMCRD of DMACm, changed |
| | | | 13. Data Transfer Controller (DTC) |
| | | 342 | 13.2.10 DTC Module Start Register (DTCST): Bit description, changed |
| | | 351 | Figure 13.5 Example of Transfer Data Read Skip, changed |
| | | 352 | 13.4.2 Transfer Data Write-Back Skip Function: Description changed |
| | | 358 | Figure 13.10 Example of DTC Operation Timing 1 (Short-Address Mode, Normal Transfer Mode, Repeat Transfer Mode), changed |
| | | 358 | Figure 13.11 Example of DTC Operation Timing 2 (Short-Address Mode, Normal Transfer Mode, Block Size = 3): Title and timing, changed |
| | | 359 | Figure 13.12 Example of DTC Operation Timing 3 (Short-Address Mode, Chain Transfer), changed |
| | | 359 | Figure 13.13 Example of DTC Operation Timing 4 (Full-Address Mode, Normal Transfer Mode, Repeat Transfer Mode), changed |
| | | 360 | 13.4.8 Execution State of the DTC, changed |
| | | 360 | Table 13.10 Execution State of the DTC, changed |
| | | | 14. I/O Ports |
| | | 437 to 443 | Table 14.8 Settings to Enable Output of the Signals from Each Port: Description changed |
| | | | 15. 16-Bit Timer Pulse Unit (TPU) |
| | | 459 to 460 | Table 15.6 Bits TPSC[2:0] (TPU0, TPU6) to Table 15.11 Bits TPSC[2:0] (TPU5, TPU11), changed |
| | | 508 | Figure 15.32 Count Timing in External Clock Operation, changed |
| | | 509 | Figure 15.33 Output Compare Output Timing, changed |
| | | 509 | Figure 15.34 Input Capture Signal Timing, changed |
| | | | 16. Programmable Pulse Generator (PPG) |
| | | 543 | Figure 16.5 Timing of Transfer and Output of the Values in NDR (Example), changed |
| | | | 17. 8-Bit Timer (TMR) |
| | | 557 | Figure 17.1 Block Diagram of TMR (Unit 0), changed |
| | | 572 | Figure 17.8 Timing of Timer Output at Compare Match A, changed |
| | | | 18. Compare Match Timer (CMT) |
| | | 591 | 18.5.4 Notes on Rewriting Compare Match Timer Control Register (CMCR), added |
| | | 591 | 18.5.5 Notes on Compare Match Timer Counter (CMCNT) and Compare Match Constant Register (CMCOR), added |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Summary |
|---|---|---|---|
| 1.10 | Apr 05, 2011 | | 20. Serial Communications Interface (SCI) |
| | | 607 | Table 20.4 Registers of SCI: Value after reset in SSR, changed |
| | | 614 | 20.2.6 Serial Control Register (SCR): Description on bits CKE[1:0], changed |
| | | | 20.2.7 Serial Status Register (SSR) |
| | | 619 | (1) Serial Communications Interface Mode (SMIF in SCMR = 0): |
| | | | Bits b7 and b6 and Note, changed |
| | | 622 | (2) Smart Card Interface Mode (SMIF in SCMR = 1): |
| | | | Bits b7 and b6 and Note, changed |
| | | 641 | Figure 20.8 Example of Operation for Serial Transmission in Asynchronous Mode |
| | | | (Example with 8-Bit Data, Parity, One Stop Bit), changed |
| | | 643 | Figure 20.10 Example of SCI Operation for Serial Reception in Asynchronous Mode |
| | | | (Example with 8-Bit Data, Parity, One Stop Bit), changed |
| | | 650 | Figure 20.15 Example of Serial Transmission Flowchart (Clock Synchronous Mode), |
| | | | changed |
| | | 652 | Figure 20.17 Example of Serial Reception Flowchart (Clock Synchronous Mode), changed |
| | | 661 | Figure 20.24 Data Retransfer Operation in SCI Transmission Mode, changed |
| | | 664 | Figure 20.27 Data Retransfer Operation in SCI Reception Mode, changed |
| | | | 22. $I^2C$ Bus Interface (RIIC) |
| | | 697 | 22.2.5 $I^2C$ Bus Mode Register 3 (ICMR3): Description on the bit table, changed |
| | | 717 | 22.2.14 $I^2C$ Bus Bit Rate High-Level Register (ICBRH), changed |
| | | 726 | 22.3.4 Master Receiver Operation (5),(6) changed |
| | | 728 | Figure 22.10 Example of Master Reception Flowchart (7-Bit Address Format) changed |
| | | 730 | Figure 22.13 Master Receive Operation Timing (3) (when RDRFS=0) changed |
| | | 751 | Figure 22.31 Automatic Low-Hold Operation in Receive Mode (Using RDRFS and WAIT |
| | | | Bits), changed |
| | | 760 | 22.12 SMBus Operation: Description changed |
| | | | 23. A/D Converter |
| | | 796 | 23.6.8 Point for Caution Regarding Board Design, changed |
| | | 798 | 23.6.10 Realizing High-Speed Conversion, changed |
| | | 799 | 23.6.11 Notes when Using Multiple Units of A/D Converter, added |
| | | | 24. D/A Converter |
| | | 808 | 24.4.5 Notes when Using the A/D Converter and D/A Converter Simultaneously, added |
| | | | 26. ROM (Flash Memory for Code Storage) |
| | | 825 | 26.2.8 Flash P/E Mode Entry Register (FENTRYR): Bit chart, changed |
| | | 833 | 26.2.14 Peripheral Clock Notification Register (PCKAR): Note on the bit description, |
| | | | changed |
| | | 847 | Table 26.10 Error Protection Types (Types Dedicated to ROM and Types Common to |
| | | | ROM and Data Flash): Parts of the description, deleted |
| | | 850 | 26.6.4.2 (4) Using the Peripheral Clock Notification Command: Description changed |
| | | 898 | 26.13 (2) Suspending Programming or Erasure: Description added |
| | | | 29. Electrical Characteristics |
| | | 949 | Table 29.3 Permissible Output Currents, changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Rev. |
|---|---|---|---|
| 1.20 | Feb 20, 2013 | | 1. Overview |
| | | 32 | Table 1.2 List of Products, product lineup added |
| | | 35 | Figure 1.3 Pin Assignment of the 176-pin LFBGA, changed |
| | | 43 | Table 1.3 List of Pins and Pin Functions (176-Pin LFBGA), changed |
| | | 49 to 53 | Table 1.5 Pin Functions, description on bus control changed, note added |
| | | | 2. CPU |
| | | 73 | 2.5.5 Notes on Arrangement of Instruction Code, description changed |
| | | | 5. I/O register |
| | | 104 to 124 | Table 5.1 List of I/O Registers (Address Order), changed |
| | | 125 to 153 | Table 5.2 List of I/O Registers (Bit Order), changed |
| | | | 7. Clock Generation Circuit |
| | | 164 | 7.3.1 Connecting a Crystal Resonator, description deleted |
| | | 164 | Figure 7.3 Equivalent Circuit of Crystal Resonator, description deleted |
| | | | 8. Low Power Consumption |
| | | 184 | 8.2.7 Deep Standby Interrupt Enable Register (DPSIER), description changed |
| | | 189 | 8.5.1.1 Transition to Sleep Mode, description changed |
| | | | 11. Buses |
| | | 263 to 264 | 11.3.1 CSi Control Register (CSiCNT) (i = 0 to 7), EMODE bit description changed |
| | | 297 | 11.5.5.3 Restrictions in Relation to RMPA and String-Manipulation Instructions, description changed |
| | | 298 | 11.5.5.5 Restriction on Instruction Code, description changed |
| | | | 13. Data Transfer Controller (DTC) |
| | | 359 | Figure 13.11 Example of DTC Operation Timing 2 (Short-Address Mode, Block Transfer Mode, Block Size = 3), title changed |
| | | 360 | Figure 13.12 Example of DTC Operation Timing 3 (Short-Address Mode, Chain Transfer) and Figure 13.13 Example of DTC Operation Timing 4 (Full-Address Mode, Normal Transfer Mode, Repeat Transfer Mode), changed |
| | | | 14. I/O Ports |
| | | 447 to 450 | 14.6 I/O Port Configuration, added |
| | | | 20. Serial Communications Interface (SCI) |
| | | 624 | 20.2.7 Serial Status Register (SSR), changed |
| | | 637 | Table 20.9 BRR Settings for Various Bit Rates (Clock Synchronous Mode), changed |
| | | 638 | Table 20.11 BRR Settings for Various Bit Rates (Smart Card Interface Mode, n = 0, S = 372), changed |
| | | 645 | Figure 20.7 Example of SCI Initialization Flowchart (Asynchronous Mode) , note added |
| | | 646 | Figure 20.8 Example of Operation for Serial Transmission in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit), changed |
| | | 653 | Figure 20.14 Example of SCI Initialization Flowchart (Clock Synchronous Mode), note added |
| | | 654 | Figure 20.15 Example of Operation for Serial Transmission in Clock Synchronous Mode |
| | | | 21. CRC Calculator (CRC) |
| | | 680 | 21.2.1 CRC Control Register (CRCCR), note deleted |
| | | 681 | 21.2.3 CRC Data Output Register (CRCDOR), changed |



Downloaded from Arrow.com.

| Rev. | Data | Description Page | Rev. |
|---|---|---|---|
| 1.20 | Feb 20, 2013 | | 22. I²C Bus Interface (RIIC) |
| | | 690 | Table 22.3 Registers of the RIIC, changed |
| | | 701 | 22.2.4 I²C Bus Mode Register 2 (ICMR2), changed |
| | | 726 | 22.2.18 Internal Counter for Timeout (TMOCNT), added |
| | | 728 | Figure 22.5 Example of RIIC Initialization Flow, changed |
| | | 730 | Figure 22.6 Example of Master Transmission Flowchart, changed |
| | | 735 | Figure 22.10 Example of Master Reception Flowchart (7-Bit Address Format), changed |
| | | 739 | Figure 22.14 Example of Slave Transmission Flowchart, changed |
| | | 742 | Figure 22.17 Example of Slave Reception Flowchart, changed |
| | | 762 | Figure 22.36 Start Condition/Restart Condition Issue Timing (ST and RS Bits), changed |
| | | | 23. A/D Converter |
| | | 803 | 23.6.8 Point for Caution Regarding Board Design, changed |
| | | | 26. ROM |
| | | 832 | 26.2.8 Flash P/E Mode Entry Register (FENTRYR), changed |
| | | 852 | Figure 26.7 Procedure for Transition to ROM Read Mode, changed |
| | | 890 | 26.10.5 (11) New Bit Rate Selection, calculation formula of bit rate selection error changed |
| | | 905 to 906 | 26.13 Usage Notes (4), (5), (8) changed, (7) added |
| | | | 27. Data Flash (Flash Memory for Data Storage) |
| | | 936 | 27.9 Usage Notes (2) Other Points to Note, changed |



Downloaded from Arrow.com.

RX610 Group User's Manual: Hardware

Publication Date: Rev.0.11 Apr 15, 2009
Rev.1.20 Feb 20, 2013

Published by: Renesas Electronics Corporation

Downloaded from Arrow.com.

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

**Renesas Electronics America Inc.**
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

**Renesas Electronics Hong Kong Limited**
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics Korea Co., Ltd.**
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141



© 2013 Renesas Electronics Corporation. All rights reserved.


Colophon 1.3



Downloaded from Arrow.com.

# RX610 Group

Renesas Electronics Corporation

R01UH0032EJ0120

Downloaded from Arrow.com.