To our customers,

## Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Downloaded from Arrow.com.

# Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "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.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. 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.
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.

Downloaded from Arrow.com.

## M30245 Group

The M30245 group is a 16-bit microcomputer based on the M16C family core technology that uses a high performance silicon gate CMOS process with an M16C/62 Series CPU core. It comes packaged in a 100-pin molded plastic LQFP. They are single-chip USB peripheral microcontrollers that operate at Full Speed (12 Mbps) and are compliant with the Universal Serial Bus (USB) Version 2.0 specification. They also include many built-in peripherals including: A/D converter, Timers, UARTs, Serial Sound Interface, $I^2C$, DMAC, CRC and more. These microcontrollers operate using sophisticated instructions featuring a high level of instruction efficiency, making them capable of executing instructions at high speed.

### Features

| | |
|---|---|
| Number of instructions | 91 |
| Shortest instruction execution time | 62.5ns f(XIN)=16MHz, Vcc=3V with no wait |
| USB features: | Supports full-speed operation (12 Mbps)<br>3.25K programmable FIFO, 9 endpoints<br>Integrated transceiver<br>Conforms to USB V 2.0 Specification |
| Frequency synthesizer | PLL for 48MHz clock |
| Memory capacity | 64K ROM / 5K RAM (M30245M8-XXXGP)<br>128K ROM / 10K RAM (M30245MC-XXXGP)<br>128K Flash /10K RAM (M30245FCGP) |
| Supply voltage | 3.0 to 3.6V (f(XIN)=16MHz) |
| Processor modes | Single chip, Memory expansion, Microprocessor |
| Interrupts | 31 internal and 5 external interrupt sources<br>4 software interrupt sources<br>7 levels (including key input interrupt X 8) |
| Multifunction 16-bit timer | 5 16-bit timers |
| Serial Communication | 2 X 7/8/9, 2 X 7/8/9/16/24/32 bits<br>Configurable for synchronous or asynchronous mode,<br>Serial Sound Interface, $I^2C$ Bus |
| DMAC | 4 channels |
| A/D Converter | 10 bits X 8 channels |
| CRC calculation circuit | 2 polynomials with MSB/LSB selectable |
| Watchdog timer | 1 line |
| Key-on wake up | 8 inputs |
| Programmable I/O | 82 lines |
| AND flash control circuit | Built-in |
| Clock-generating circuit | 2 built-in clock generation circuits<br>(built-in feedback resistor, and external ceramic or quartz oscillator) |



Downloaded from Arrow.com.

# Table of Contents


Features ..... 1
Description ..... 3
Memory ..... 12
Central Processing Unit ..... 13
Reset ..... 16
Special Function Registers ..... 18
Processor Modes ..... 26
System Clock ..... 38
Power Control ..... 45
Frequency synthesizer circuit ..... 47
Interrupts ..... 50
INT interrupt ..... 62
NMI Interrupt ..... 62
Key-Input Interrupt ..... 62
Address-Match Interrupt ..... 65
Watchdog Timer ..... 68
Universal Serial Bus ..... 70
Vbus Detect ..... 98
Direct memory access controller ..... 100
Timer A ..... 110
Serial Communication ..... 125
Clock synchronous serial I/O mode ..... 132
Clock asynchronous serial I/O (UART) mode ..... 137
UART mode (compliant with the SIM interface) ..... 142
$I^2C$ Bus interface mode ..... 145
Serial Interface Special Function (SPI mode) ..... 151
IE mode ..... 154
Serial Sound Interface ..... 156
A/D converter ..... 168
CRC calculation circuit ..... 177
Programmable I/O ports ..... 180
AND Flash Control Circuit ..... 190
Flash memory ..... 195
CPU Rewrite Mode ..... 197
Parallel I/O Mode ..... 206
Standard Serial I/O Mode ..... 208
Standard serial I/O mode 1 ..... 211
Standard serial I/O mode 2 ..... 223
Electrical Specifications ..... 234
Usage Notes ..... 250




Downloaded from Arrow.com.

## Applications

USB peripherals, such as telephones, audio systems, office equipment, communications equipment, portable devices, scanners, digital cameras, and memory card readers.

## Block Diagram

Figure 1.1 is a block diagram of the M30245 group.

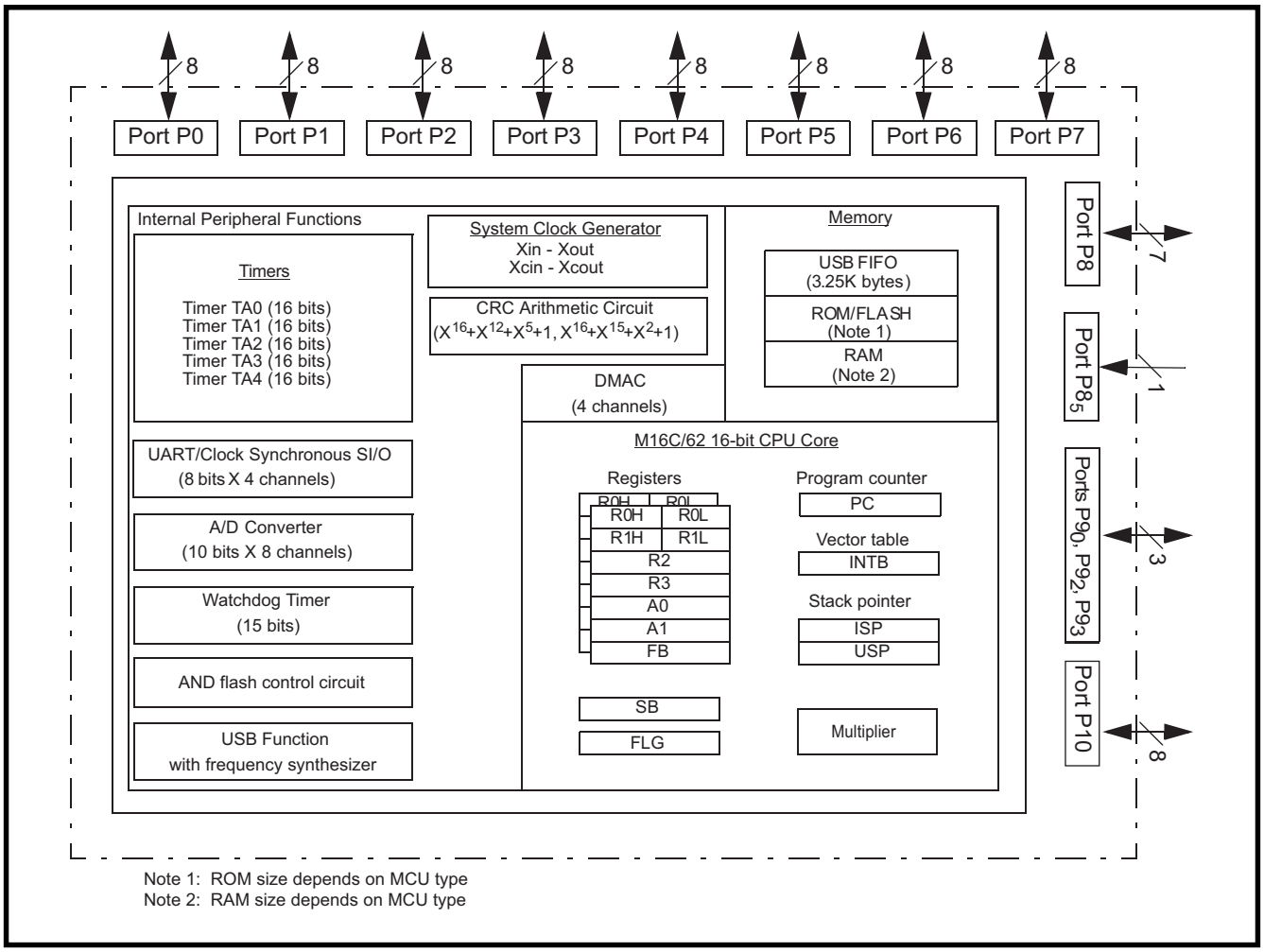



**Figure 1.1. Block diagram of M30245 group**



Downloaded from Arrow.com.

## Performance outline

Table 1.1 is a performance outline of the M30245 group.

**Table 1.1. M30245 Group performance outline**

| Parameters | | | Function Description |
|---|---|---|---|
| Number of basic Instructions | | | 91 |
| Shortest Instruction execution time | | | 62.5 ns f(Xin)= 16 MHz, Vcc = 3V |
| Memory size | ROM | | 128/64 Kbytes |
| | RAM | | 10/5 Kbytes |
| Input/Output ports | P0 to P8, P10 (excl $P8_5$) | I/O | 8 bits x 10 |
| | $P8_5$ | I | 1 bit x 1 |
| | P9 | I/O | 4 bits x 1 |
| Multifunction timer | TA0, TA1, TA2, TA3, TA4 | | 16 bits x 5 |
| Serial I/O | UART0 to 1 | | UART (or clock synchronous or Serial Sound Interface) x 2 |
| | UART2 to 3 | | UART (or clock synchronous) x 2 |
| A/D converter | | | 10 bits x 8 channels |
| DMAC | | | 4 channels (31 trigger sources) |
| CRC calculation circuits | | | CRC-CCITT and CRC-16 |
| AND flash control circuit | | | Communicate with external AND type flash memory |
| Watchdog timer | | | 15 bits x 1 (with prescaler) |
| Interrupts | | | 31 internal, 4 external sources, 4 software, 7 levels |
| Clock-generating circuit | | | 2 built-in clock generating circuits<br>(built in feedback resistor, and external ceramic or quartz oscillator) |
| Supply voltage | | | 3.0 ~ 3.6, f(XIN)=16MHz |
| Power consumption | | | 25mA (Vcc=3.3V, f(XIN)=16MHz no division, USB ON) |
| | | | 16mA (Vcc=3.3V, f(XIN)=16MHz no division, USB OFF) |
| Operating temperature | | | -20 to 85°C |
| Package | | | 100-pin plastic mold LQFP |



Downloaded from Arrow.com.

## Pin Configuration

Figure 1.2 shows the pin configuration (top view). Table 1.2 lists the pin cross references and Table 1.3 describes the pin functions.

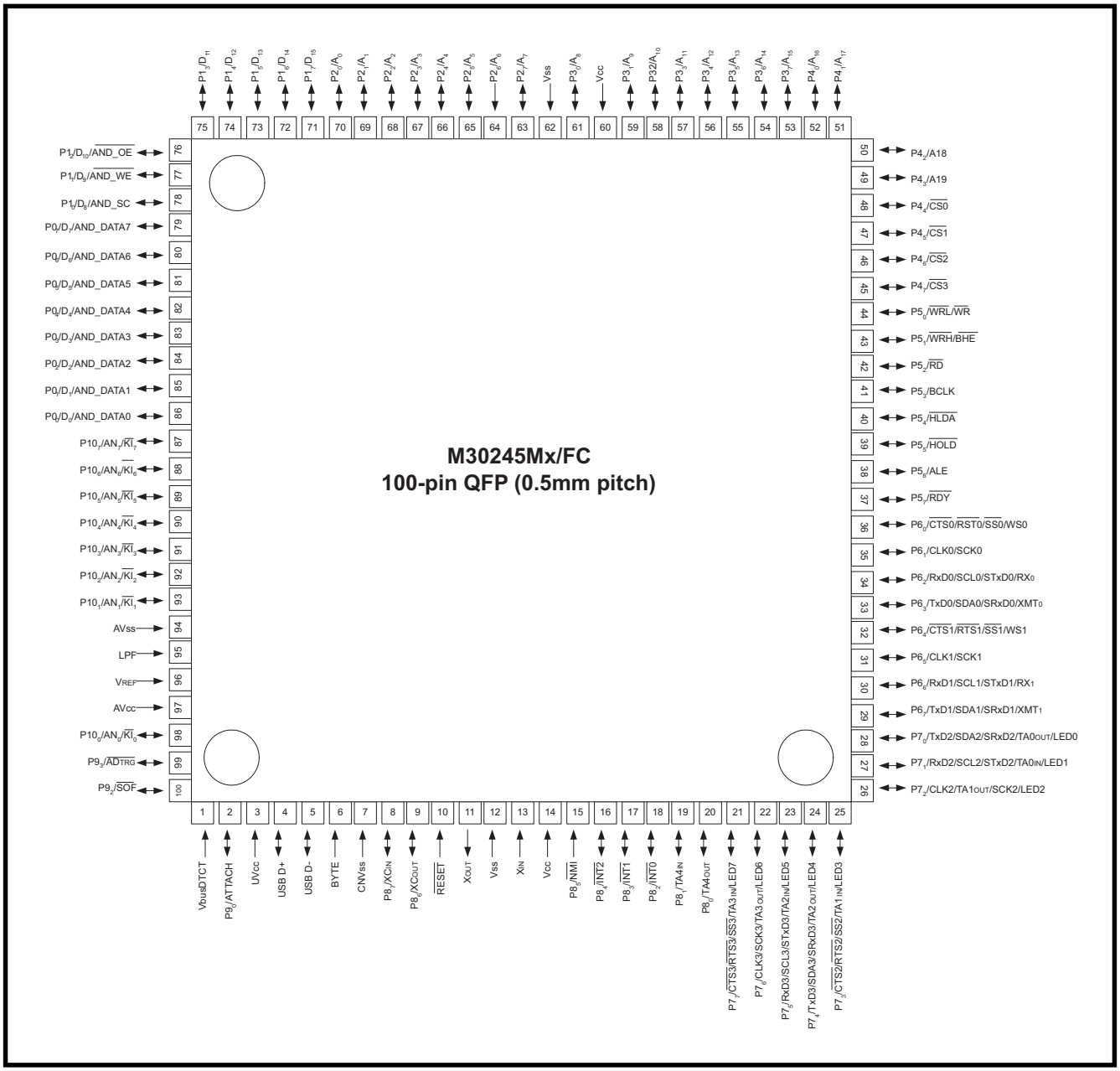



**Figure 1.2. Pin Configuration (top view)**



Downloaded from Arrow.com.

**Table 1.2. Pin cross reference**

| Pin No. | Control | Port | Interrupt | Timer | UART/ USB | SPI | $I^2C$ | Serial Sound Interface | Analog/ Other | Bus Control |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | Vbus DTCT | | | | | |
| 2 | | $P9_0$ | | | ATTACH | | | | | |
| 3 | UVcc | | | | | | | | | |
| 4 | | | | | USB D+ | | | | | |
| 5 | | | | | USB D- | | | | | |
| 6 | BYTE | | | | | | | | | |
| 7 | CNVss | | | | | | | | | |
| 8 | XCIN | $P8_7$ | | | | | | | | |
| 9 | XCOUT | $P8_6$ | | | | | | | | |
| 10 | $\overline{\text{RESET}}$ | | | | | | | | | |
| 11 | XOUT | | | | | | | | | |
| 12 | Vss | | | | | | | | | |
| 13 | XIN | | | | | | | | | |
| 14 | Vcc | | | | | | | | | |
| 15 | | $P8_5$ | $\overline{\text{NMI}}$ | | | | | | | |
| 16 | | $P8_4$ | $\overline{\text{INT2}}$ | | | | | | | |
| 17 | | $P8_3$ | $\overline{\text{INT1}}$ | | | | | | | |
| 18 | | $P8_2$ | $\overline{\text{INT0}}$ | | | | | | | |
| 19 | | $P8_1$ | | TA4IN | | | | | | |
| 20 | | $P8_0$ | | TA4OUT | | | | | | |
| 21 | | $P7_7$ | | TA3IN | $\overline{\text{CTS3}}/\overline{\text{RTS3}}$ | $\overline{\text{SS3}}$ | | | LED7 | |
| 22 | | $P7_6$ | | TA3OUT | CLK3 | SCK3 | CLK3 | | LED6 | |
| 23 | | $P7_5$ | | TA2IN | RxD3 | STxD3 | SCL3 | | LED5 | |
| 24 | | $P7_4$ | | TA2OUT | TxD3 | SRxD3 | SDA3 | | LED4 | |
| 25 | | $P7_3$ | | TA1IN | $\overline{\text{CTS2}}/\overline{\text{RTS2}}$ | $\overline{\text{SS2}}$ | | | LED3 | |
| 26 | | $P7_2$ | | TA1OUT | CLK2 | SCK2 | CLK2 | | LED2 | |
| 27 | | $P7_1$ | | TA0IN | RxD2 | STxD2 | SCL2 | | LED1 | |
| 28 | | $P7_0$ | | TA0OUT | TxD2 | SRxD2 | SDA2 | | LED0 | |
| 29 | | $P6_7$ | | | TxD1 | SRxD1 | SDA1 | XMT1 | | |
| 30 | | $P6_6$ | | | RxD1 | STxD1 | SCL1 | RX1 | | |
| 31 | | $P6_5$ | | | CLK1 | SCK1 | CLK1 | SCK1 | | |
| 32 | | $P6_4$ | | | $\overline{\text{CTS1}}/\overline{\text{RTS1}}$ | $\overline{\text{SS1}}$ | | WS1 | | |
| 33 | | $P6_3$ | | | TxD0 | SRxD0 | SDA0 | XMT0 | | |
| 34 | | $P6_2$ | | | RxD0 | STxD0 | SCL0 | RX0 | | |
| 35 | | $P6_1$ | | | CLK0 | SCK0 | CLK0 | SCK0 | | |
| 36 | | $P6_0$ | | | $\overline{\text{CTS0}}/\overline{\text{RTS0}}$ | $\overline{\text{SS0}}$ | | WS0 | | |
| 37 | | $P5_7$ | | | | | | | | $\overline{\text{RDY}}$ |
| 38 | | $P5_6$ | | | | | | | | ALE |
| 39 | | $P5_5$ | | | | | | | | $\overline{\text{HOLD}}$ |
| 40 | | $P5_4$ | | | | | | | | $\overline{\text{HLDA}}$ |
| 41 | | $P5_3$ | | | | | | | | BCLK |
| 42 | | $P5_2$ | | | | | | | | $\overline{\text{RD}}$ |
| 43 | | $P5_1$ | | | | | | | | $\overline{\text{WRH}}/\overline{\text{BHE}}$ |
| 44 | | $P5_0$ | | | | | | | | $\overline{\text{WRL}}/\overline{\text{WR}}$ |
| 45 | | $P4_7$ | | | | | | | | $\overline{\text{CS3}}$ |
| 46 | | $P4_6$ | | | | | | | | $\overline{\text{CS2}}$ |
| 47 | | $P4_5$ | | | | | | | | $\overline{\text{CS1}}$ |
| 48 | | $P4_4$ | | | | | | | | $\overline{\text{CS0}}$ |
| 49 | | $P4_3$ | | | | | | | | A19 |
| 50 | | $P4_2$ | | | | | | | | A18 |
| 51 | | $P4_1$ | | | | | | | | A17 |



Downloaded from Arrow.com.

**Table 1.2 Pin cross reference**

| Pin No. | Control | Port | Interrupt | Timer | UART/ USB | SPI | $I^2C$ | Serial Sound Interface | Analog/ Other | Bus Control |
|---|---|---|---|---|---|---|---|---|---|---|
| 52 | | $P4_0$ | | | | | | | | $A_{16}$ |
| 53 | | $P3_7$ | | | | | | | | $A_{15}$ |
| 54 | | $P3_6$ | | | | | | | | $A_{14}$ |
| 55 | | $P3_5$ | | | | | | | | $A_{13}$ |
| 56 | | $P3_4$ | | | | | | | | $A_{12}$ |
| 57 | | $P3_3$ | | | | | | | | $A_{11}$ |
| 58 | | $P3_2$ | | | | | | | | $A_{10}$ |
| 59 | | $P3_1$ | | | | | | | | $A_9$ |
| 60 | Vcc | | | | | | | | | |
| 61 | | $P3_0$ | | | | | | | | $A_8$ |
| 62 | Vss | | | | | | | | | |
| 63 | | $P2_7$ | | | | | | | | $A_7$ |
| 64 | | $P2_6$ | | | | | | | | $A_6$ |
| 65 | | $P2_5$ | | | | | | | | $A_5$ |
| 66 | | $P2_4$ | | | | | | | | $A_4$ |
| 67 | | $P2_3$ | | | | | | | | $A_3$ |
| 68 | | $P2_2$ | | | | | | | | $A_2$ |
| 69 | | $P2_1$ | | | | | | | | $A_1$ |
| 70 | | $P2_0$ | | | | | | | | $A_0$ |
| 71 | | $P1_7$ | | | | | | | | $D_{15}$ |
| 72 | | $P1_6$ | | | | | | | | $D_{14}$ |
| 73 | | $P1_5$ | | | | | | | | $D_{13}$ |
| 74 | | $P1_4$ | | | | | | | | $D_{12}$ |
| 75 | | $P1_3$ | | | | | | | | $D_{11}$ |
| 76 | | $P1_2$ | | | | | | | $\overline{\text{AND_OE}}$ | $D_{10}$ |
| 77 | | $P1_1$ | | | | | | | $\overline{\text{AND_WE}}$ | $D_9$ |
| 78 | | $P1_0$ | | | | | | | AND_SC | $D_8$ |
| 79 | | $P0_7$ | | | | | | | AND_DATA7 | $D_7$ |
| 80 | | $P0_6$ | | | | | | | AND_DATA6 | $D_6$ |
| 81 | | $P0_5$ | | | | | | | AND_DATA5 | $D_5$ |
| 82 | | $P0_4$ | | | | | | | AND_DATA4 | $D_4$ |
| 83 | | $P0_3$ | | | | | | | AND_DATA3 | $D_3$ |
| 84 | | $P0_2$ | | | | | | | AND_DATA2 | $D_2$ |
| 85 | | $P0_1$ | | | | | | | AND_DATA1 | $D_1$ |
| 86 | | $P0_0$ | | | | | | | AND_DATA0 | $D_0$ |
| 87 | | $P10_7$ | $\overline{KI_7}$ | | | | | | $AN_7$ | |
| 88 | | $P10_6$ | $\overline{KI_6}$ | | | | | | $AN_6$ | |
| 89 | | $P10_5$ | $\overline{KI_5}$ | | | | | | $AN_5$ | |
| 90 | | $P10_4$ | $\overline{KI_4}$ | | | | | | $AN_4$ | |
| 91 | | $P10_3$ | $\overline{KI_3}$ | | | | | | $AN_3$ | |
| 92 | | $P10_2$ | $\overline{KI_2}$ | | | | | | $AN_2$ | |
| 93 | | $P10_1$ | $\overline{KI_1}$ | | | | | | $AN_1$ | |
| 94 | AVss | | | | | | | | | |
| 95 | LPF | | | | | | | | | |
| 96 | | | | | | | | | $V_{REF}$ | |
| 97 | AVcc | | | | | | | | | |
| 98 | | $P10_0$ | $\overline{KI_0}$ | | | | | | $AN_0$ | |
| 99 | | $P9_3$ | | | | | | | $\overline{AD_{TRG}}$ | |
| 100 | | $P9_2$ | | | $\overline{SOF}$ | | | | | |



Downloaded from Arrow.com.

**Table 1.3 Pin description**

| Port | Function | Pin Name | I/O | Description |
|---|---|---|---|---|
| | Power supply input | Vcc | | 3.0 to 3.6V |
| | | Vss | | 0V |
| | CPU mode switch | CNVss | I | Connect to Vss: single-chip or memory expansion mode<br>Connect to Vcc: Microprocessor mode only |
| | External data bus width select input | BYTE | I | Selects external memory data bus width.<br>Connect to Vss: 16-bit.<br>Connect to Vcc: 8-bit. |
| | Reset input | $\overline{\text{RESET}}$ | I | "L" resets the microcomputer. |
| | Clock input | XIN | I | These pins support the main clock generating circuit. Connect a crystal between the XIN and XOUT pins. To use an external clock, input it to the XIN pin and leave the XOUT pin open. |
| | Clock output | XOUT | O | |
| | Analog power supply input | AVcc | | Connect to Vcc. |
| | | AVss | | Connect to Vss. |
| | Reference voltage input | VREF | I | This is a reference voltage for A/D converter. |
| | Low pass filter | LPF | O | Loop filter for the frequency synthesizer circuit. |
| | USB power supply input | UVcc | | Power pin for USB |
| | Vbus detect | VbusDTCT | I | Detects USB host power |
| | USB D+ | USB D+ | I/O | USB D+ voltage line interface |
| | USB D- | USB D- | I/O | USB D- voltage line interface |
| P0 | I/O port | $P0_0$ to $P0_7$ | I/O | This is an 8-bit CMOS I/O port. The input/output port direction register allows each pin to be set individually. When used for input, the port can be set to include internal pull-up resistors in 4-pin blocks. |
| | Data bus | $D_0$ to $D_7$ | I/O | These pins input and output 8 low-order data bits when set as a separate bus. |
| | AND Flash control | AND_DATA0 to 7 | I/O | Data pins for communicating with AND type flash memory devices |
| P1 | I/O port | $P1_0$ to $P1_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Data bus | $D_8$ to $D_{15}$ | I/O | These pins input and output 8 high-order data bits when set as a separate bus. |
| | AND Flash control | AND_SC<br>$\overline{\text{AND_WE}}$<br>$\overline{\text{AND_OE}}$ | O | Control signal pins for communicating with AND type flash memory devices |
| P2 | I/O port | $P2_0$ to $P2_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Address bus | $A_0$ to $A_7$ | O | These pins output 8 low-order address bits. |
| P3 | I/O port | $P3_0$ to $P3_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Address bus | $A_8$ to $A_{15}$ | O | These pins output 8 middle-order address bits. |
| P4 | I/O port | $P4_0$ to $P4_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Address bus | $A_{16}$ to $A_{19}$ | O | These pins output 4 high-order address bits. |
| | Chip select | $\overline{\text{CS0}}$ to $\overline{\text{CS3}}$ | O | $P4_4$ to $P4_7$ are chip select output pins that specify access areas. |
| P5 | I/O port | $P5_0$ to $P5_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Bus control | $\overline{\text{WRL}}$/$\overline{\text{WR}}$<br>$\overline{\text{WRH}}$/$\overline{\text{BHE}}$<br>$\overline{\text{RD}}$ | O<br>O<br>O | Ouput $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ ($\overline{\text{WR}}$, $\overline{\text{BHE}}$), and $\overline{\text{RD}}$ bus control signals. Using $\overline{\text{WRL}}$ and $\overline{\text{WRH}}$ or $\overline{\text{WR}}$ and $\overline{\text{BHE}}$ can be switched using software control.<br>■ $\overline{\text{WRL}}$, $\overline{\text{WRH}}$, and $\overline{\text{RD}}$ selected.<br>With a 16-bit external data bus, data is written to even addresses when the $\overline{\text{WRL}}$ signal is "L", and to the odd addresses when the $\overline{\text{WRH}}$ signal is "L". Data is read when $\overline{\text{RD}}$ is "L".<br>■ $\overline{\text{WR}}$, $\overline{\text{BHE}}$, and $\overline{\text{RD}}$ selected.<br>Data is written when $\overline{\text{WR}}$ is "L". Data is read when $\overline{\text{RD}}$ is "L". Odd addresses are accessed when $\overline{\text{BHE}}$ is "L". Use this mode when using an 8-bit external data bus. |
| | | BCLK | O | Output operation clock for the CPU. |
| | | $\overline{\text{HOLD}}$ | I | While the $\overline{\text{HOLD}}$ pin input is "L", the MCU is placed in the Hold state. |
| | | $\overline{\text{HLDA}}$ | O | The $\overline{\text{HLDA}}$ pin output is "L" while the MCU is in the hold state. |
| | | ALE | O | The ALE pin can be used to latch the address. |
| | | $\overline{\text{RDY}}$ | I | While the $\overline{\text{RDY}}$ pin input is "L", the MCU is in the ready state. |



Downloaded from Arrow.com.

**Table 1.3 Pin description**

| Port | Function | Pin Name | I/O | Description |
|---|---|---|---|---|
| P6 | I/O port | $P6_0$ to $P6_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | UART/SSI | $\overline{CTS}/\overline{RTS}/\overline{SS}/\overline{WS}$<br>CLK/SCK<br>RxD/SCL/STxD/RX<br>TxD/SDA/SRxD/XMT | I/O | $P6_0$ to $P6_3$ are I/O ports for UART0.<br>$P6_4$ to $P6_7$ are I/O ports for UART1.<br>These pins can be used for Serial Sound Interface, $I^2C$ and SPI communication. |
| P7 | I/O port | $P7_0$ to $P7_7$ | I/O | This is an 8-bit I/O port equivalent to P0. $P7_0$ and $P7_1$ are N-channel open drain output. |
| | Timer A | TAIN | I | $P7_0$ to $P7_7$ are I/O ports for Timer A0 to A3. |
| | | $\overline{TAOUT}$ | O | |
| | UART | $\overline{CTS}/\overline{RTS}/\overline{SS}/\overline{WS}$<br>CLK/SCK<br>RxD/SCL/STxD<br>TxD/SDA/SRxD | I/O | $P7_0$ to $P7_3$ are I/O ports for UART2.<br>$P7_4$ to $P7_7$ are I/O ports for UART3.<br>These pins can be used for $I^2C$ and SPI communication. |
| | LED drive | $LED_0$ to $LED_7$ | O | These pins are capable of sinking 20mA for driving LEDs. |
| P8 | I/O port | $P8_0$ to $P8_4$, $P8_6$, $P8_7$ | I/O | This is a 7-bit I/O port equivalent to P0. |
| | External interrupt input | $\overline{INT0}$ to $\overline{INT2}$ | I | $P8_2$ to $P8_4$ are external interrupt input ports. |
| | Input | $P8_5/\overline{NMI}$ | I | Input port for $\overline{NMI}$ interrupt. |
| | Sub-Clock | XCIN, XCOUT | I, O | $P8_6$ and $P8_7$, connect an oscillator between these pins for sub-clock generation. |
| P9 | I/O port | $P9_0$, $P9_2$, $P9_3$ | I/O | This is a 3-bit I/O port equivalent to P0. |
| | A/D | $\overline{AD_{TRG}}$ | I | $P9_3$ is an A/D trigger input port. |
| | USB-ATTACH | ATTACH | O | $P9_0$ can be used to attach or detach from the USB host without disconnecting the USB cable. |
| | USB SOF | SOF | O | P92 is an output for the USB start of frame signal pulse. |
| P10 | I/O port | $P10_0$ to $P10_7$ | I/O | This is an 8-bit I/O port equivalent to P0. |
| | Key-input interrupt | $KI_0$ to $KI_7$ | I | $P10_0$ to $P10_7$ are key-input interrupt ports. |
| | Analog input | $AN_0$ to $AN_7$ | I | $P10_0$ to $P10_7$ are analog input ports for A/D converter. |



Downloaded from Arrow.com.

Renesas plans to release the following products in the M30245 group:

(1) Support for Flash memory version and mask ROM versions

(2) ROM capacity: 128 or 64 Kbytes

(3) Package: 100P6Q-A Plastic molded LQFP

Figure 1.3 shows the type number, memory size and package for the M30245 group. Table 1.4 lists the package number , type, ROM and RAM capacity for the M30245 group.

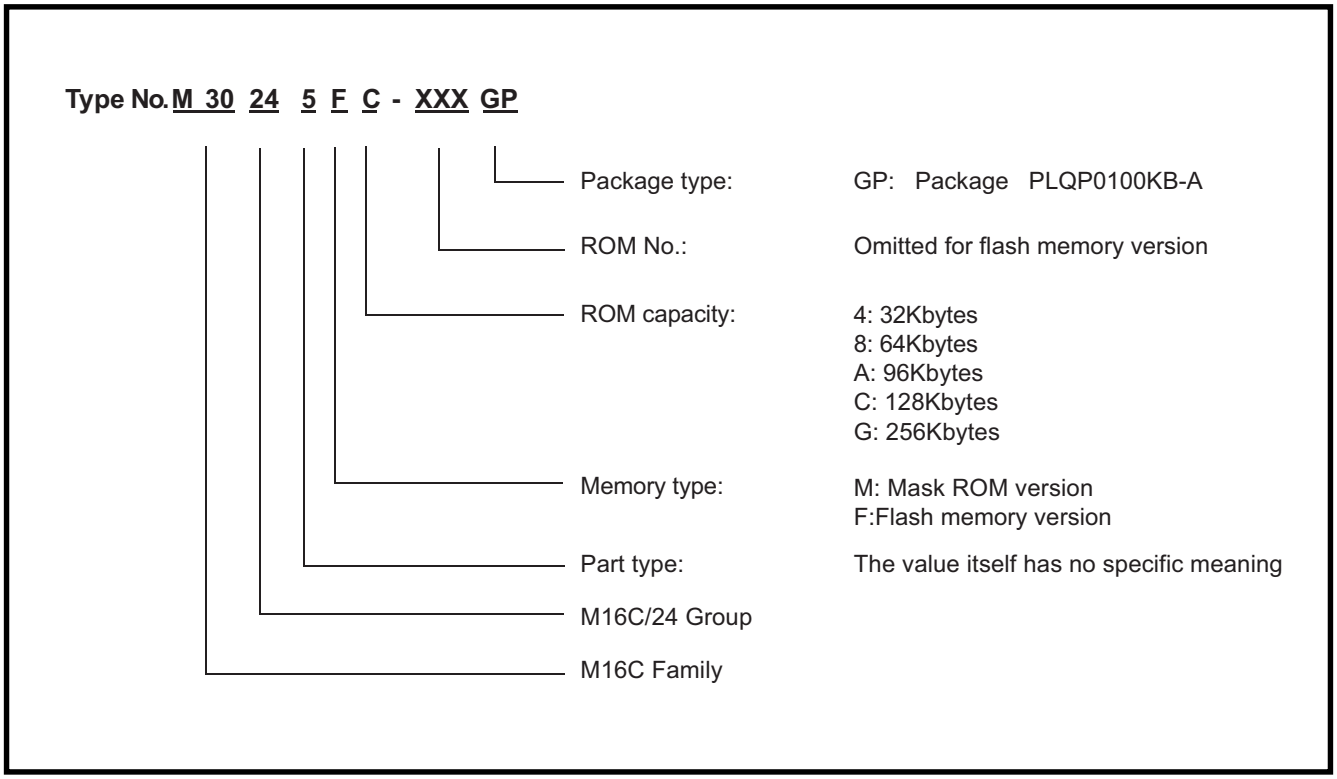



**Figure 1.3. Type number, memory size, and package**

**Table 1.4. Package number, type, ROM and RAM capacity for M30245 group**

| Type | ROM Capacity | RAM Capacity | Package Type | Remarks |
|---|---|---|---|---|
| **M30245FCGP** | 128K bytes | 10K bytes | PLQP0100KB-A | Flash Memory Version |
| **M30245MC-XXXGP** | 128K bytes | 10K bytes | PLQP0100KB-A | Mask ROM Version |
| **M30245M8-XXXGP** | 64K bytes | 5K bytes | PLQP0100KB-A | Mask ROM Version |



Downloaded from Arrow.com.

## USB Overview

The M30245 group is a single-chip PC peripheral microcontroller that is compliant with the Universal Serial Bus (USB) Version 2.0 specification for full-speed USB operation (12Mbps). This device provides an interface between a USB-equipped host computer and PC peripherals such as telephones, audio systems, and digital cameras. The M30245 architectural overview is shown in Figure 1.4.

The USB function control unit of the M30245 group supports full-speed operation and all four data transfer types listed in the USB specification. Each transfer type is used for controlling a different set of PC peripherals.

• **Isochronous transfers** provide guaranteed bus access, a constant data rate, and error tolerance for devices such as computer-telephone integration (CTI) and audio systems.

• **Interrupt transfers** are designed to support human input devices (HID) that communicate small amounts of data infrequently.

• **Bulk transfers** are necessary for devices such as digital cameras and scanners that communicate large amounts of data to the PC as bus bandwidth becomes free.

• **Control transfers** are supported and are useful for bursty, host-initiated type communication where bus management is the primary concern.

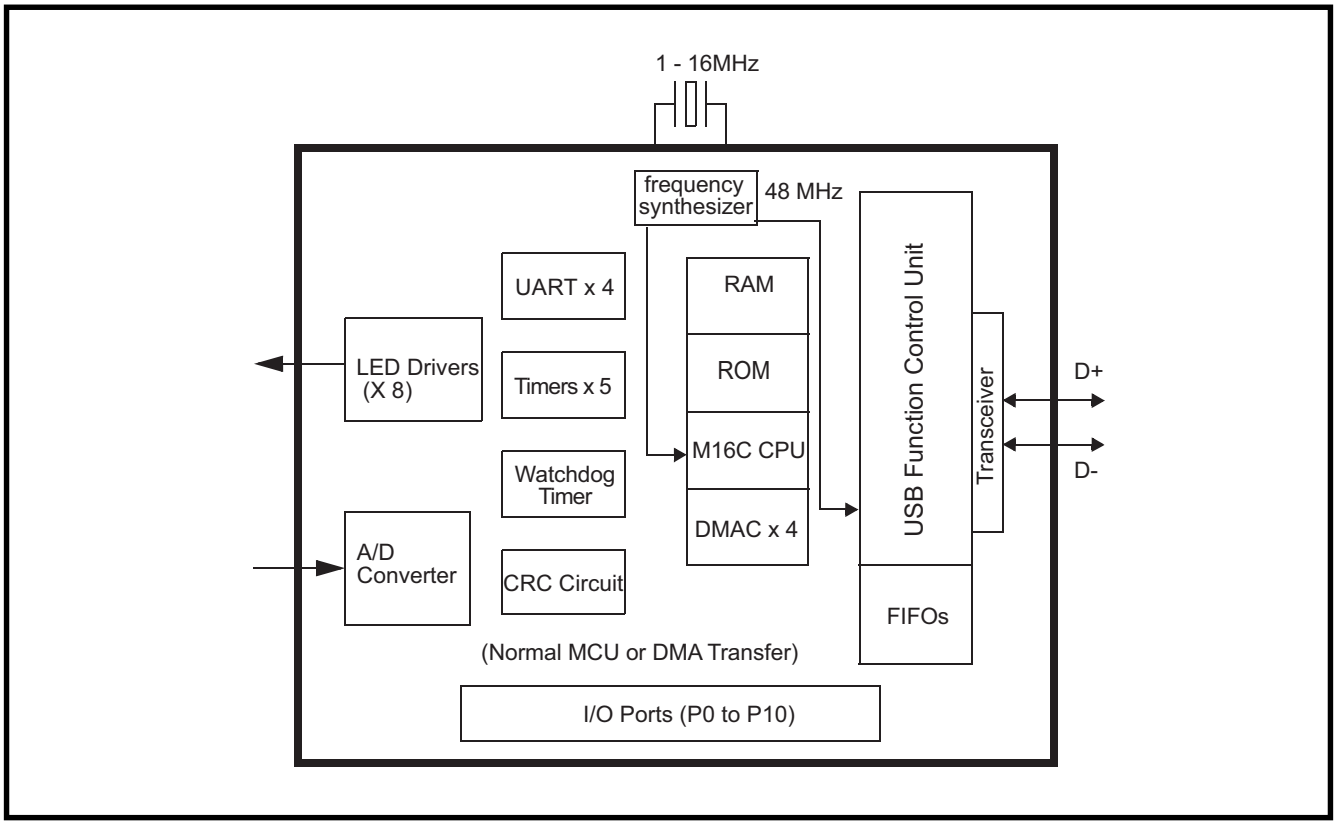



**Figure 1.4. M30245 architectural overview**



Downloaded from Arrow.com.

# Functional Block Operation

The M30245 group contains many functional blocks in a single chip. These blocks include ROM and RAM to store instructions and data and the central processing unit (CPU) to execute arithmetic/logic operations. Also included are peripheral blocks such as Timer A, serial I/O, DMAC, CRC calculation circuit, A/D converter, and I/O ports.

## Memory

Figure 1.5 is a memory map of the M30245 group. The address space extends the 1M bytes from address $00000_{16}$ to $FFFFF_{16}$. From $FFFFF_{16}$ down is ROM. For example, in the M30245MC-XXXGP, there is 128K bytes of internal ROM from $E0000_{16}$ to $FFFFF_{16}$. The vector table for fixed interrupts such as the reset and $\overline{NMI}$ are mapped to $FFFDC_{16}$ to $FFFFF_{16}$. The starting address of the interrupt routine is stored here. The address of the vector table for timer interrupts, etc., can be set as desired using the internal register (INTB). See the section on interrupts for details.

From $00400_{16}$ up is RAM. For example, in the M30245MC-XXXGP, 10K bytes of internal RAM is mapped to the space from $00400_{16}$ to $02BFF_{16}$. In addition to storing data, the RAM also stores the stack used when calling subroutines and when interrupts are generated.

The SFR area is mapped to $00000_{16}$ to $003FF_{16}$. This area accommodates the control registers for peripheral devices such as I/O ports, A/D converter, serial I/O, and timers, etc. Any part of the SFR area that is not occupied is reserved and cannot be used for other purposes.

The special page vector table is mapped to $FFE00_{16}$ to $FFFDB_{16}$. If the starting addresses of subroutines or the destination addresses of jumps are stored here, subroutine call instructions and jump instructions can be used as 2-byte instructions, reducing the number of program steps.

In memory expansion mode and microprocessor mode, some memory areas are reserved and cannot be used. For example, in the M30245MC-XXXGP, the following areas cannot be used.

- The space between $02C00_{16}$ and $03FFF_{16}$ (Memory expansion and microprocessor modes)
- The space between $D0000_{16}$ and $E0000_{16}$ (Memory expansion mode)

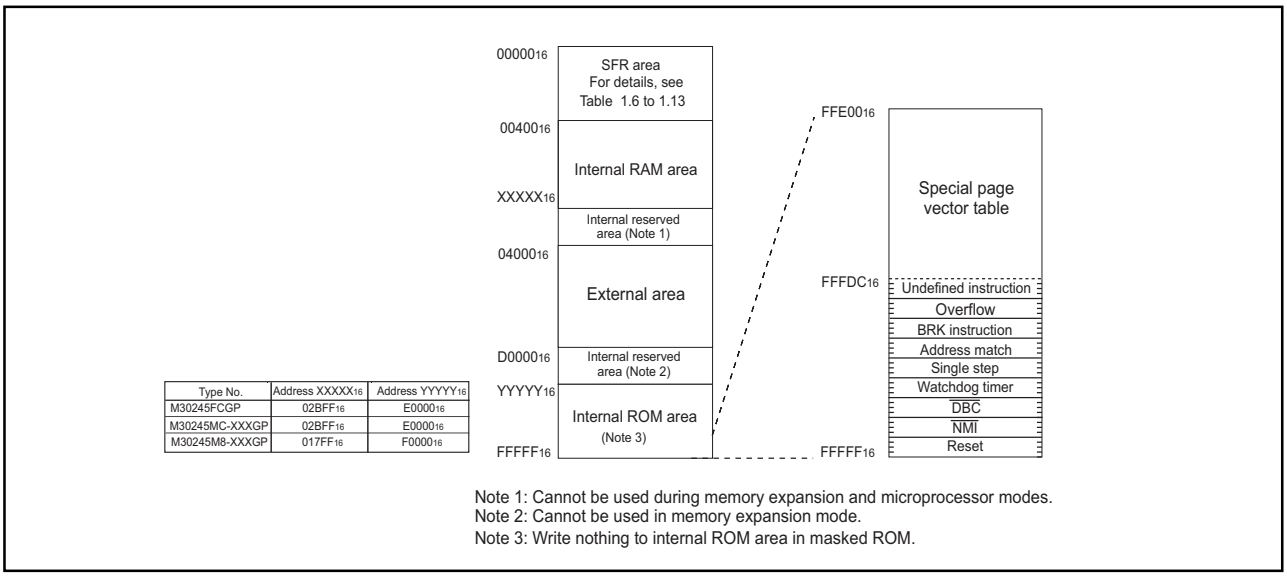


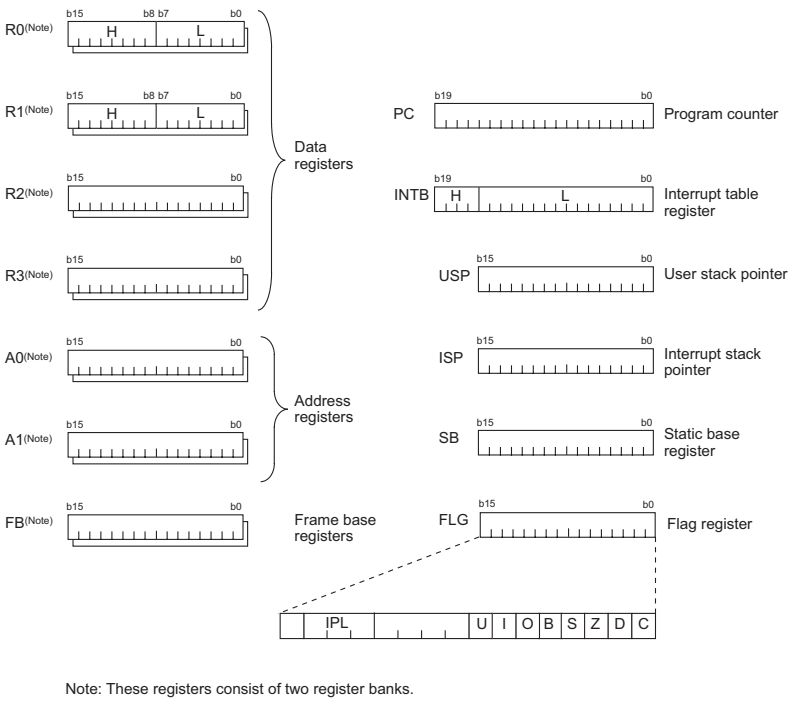

**Figure 1.5. Memory map**



Downloaded from Arrow.com.

## Central Processing Unit

The CPU has a total of 13 registers shown in Figure 1.6. Seven of these registers (R0, R1, R2, R3, A0, A1, and FB) come in two sets; therefore, these have two register banks.




Note: These registers consist of two register banks.

**Figure 1.6. Central processing unit register**

### Data registers (R0, R0H, R0L, R1, R1H, R1L, R2, and R3)

Data registers (R0, R1, R2, and R3) are configured with 16 bits, and are used primarily for transfer and arithmetic/logic operations.

Registers R0 and R1 each can be used as separate 8-bit data registers, high-order bits as (R0H/R1H), and low-order bits as (R0L/R1L). In some instructions, registers R2 and R0, as well as R3 and R1 can use as 32-bit data registers (R2R0/R3R1).

### Address registers (A0 and A1)

Address registers (A0 and A1) are configured with 16 bits, and have functions equivalent to those of data registers. These registers can also be used for address register indirect addressing and address register relative addressing.

In some instructions, registers A1 and A0 can be combined for use as a 32-bit address register (A1A0).

### Frame base register (FB)

Frame base register (FB) is configured with 16 bits, and is used for FB relative addressing.



Downloaded from Arrow.com.

**Program counter (PC)**

Program counter (PC) is configured with 20 bits, indicating the address of an instruction to be executed.

**Interrupt table register (INTB)**

Interrupt table register (INTB) is configured with 20 bits, indicating the start address of an interrupt vector table.

**Stack pointer (USP/ISP)**

The stack pointer comes in two types: user stack pointer (USP) and interrupt stack pointer (ISP), each configured with 16 bits.

The desired type of stack pointer (USP or ISP) can be selected by the stack pointer select flag (U flag). This flag is located at bit 7 of the flag register (FLG).

**Static base register (SB)**

Static base register (SB) is configured with 16 bits, and is used for SB relative addressing.

**Flag register (FLG)**

Flag register (FLG) is configured with 11 bits, each bit is used as a flag. Figure 1.7 shows the flag register (FLG). The following explains the function of each flag:

- **Bit 0: Carry flag (C flag)**

  This flag retains a carry, borrow, or shift-out bit that has occurred in the arithmetic/logic unit.

- **Bit 1: Debug flag (D flag)**

  This flag enables a single-step interrupt.

  When this flag is "1", a single-step interrupt is generated after instruction execution. This flag is cleared to "0" when the interrupt is acknowledged.

- **Bit 2: Zero flag (Z flag)**

  This flag is set to "1" when an arithmetic operation resulted in 0; otherwise, cleared to "0".

- **Bit 3: Sign flag (S flag)**

  This flag is set to "1" when an arithmetic operation resulted in a negative value; otherwise, cleared to "0".

- **Bit 4: Register bank select flag (B flag)**

  This flag chooses a register bank. Register bank 0 is selected when this flag is "0"; register bank 1 is selected when this flag is "1".

- **Bit 5: Overflow flag (O flag)**

  This flag is set to "1" when an arithmetic operation resulted in overflow; otherwise, cleared to "0".

- **Bit 6: Interrupt enable flag (I flag)**

  This flag enables all maskable interrupts.

  Interrupts are disabled when this flag is "0", and are enabled when this flag is "1". This flag is cleared to "0" when an interrupt is acknowledged.



Downloaded from Arrow.com.

• **Bit 7: Stack pointer select flag (U flag)**

Interrupt stack pointer (ISP) is selected when this flag is "0"; user stack pointer (USP) is selected when this flag is "1".

This flag is cleared to "0" when a hardware interrupt is acknowledged or an INT instruction of software interrupt Nos. 0 to 31 is executed.

• **Bits 8 to 11: Reserved area**

• **Bits 12 to 14: Processor interrupt priority level (IPL)**

Processor interrupt priority level (IPL) is configured with three bits, for specification of up to eight processor interrupt priority levels from level 0 to level 7.

If a requested interrupt has priority greater than the processor interrupt priority level (IPL), the interrupt is enabled.

• **Bit 15: Reserved area**

The C, Z, S, and O flags are changed when instructions are executed. See the software manual for details.

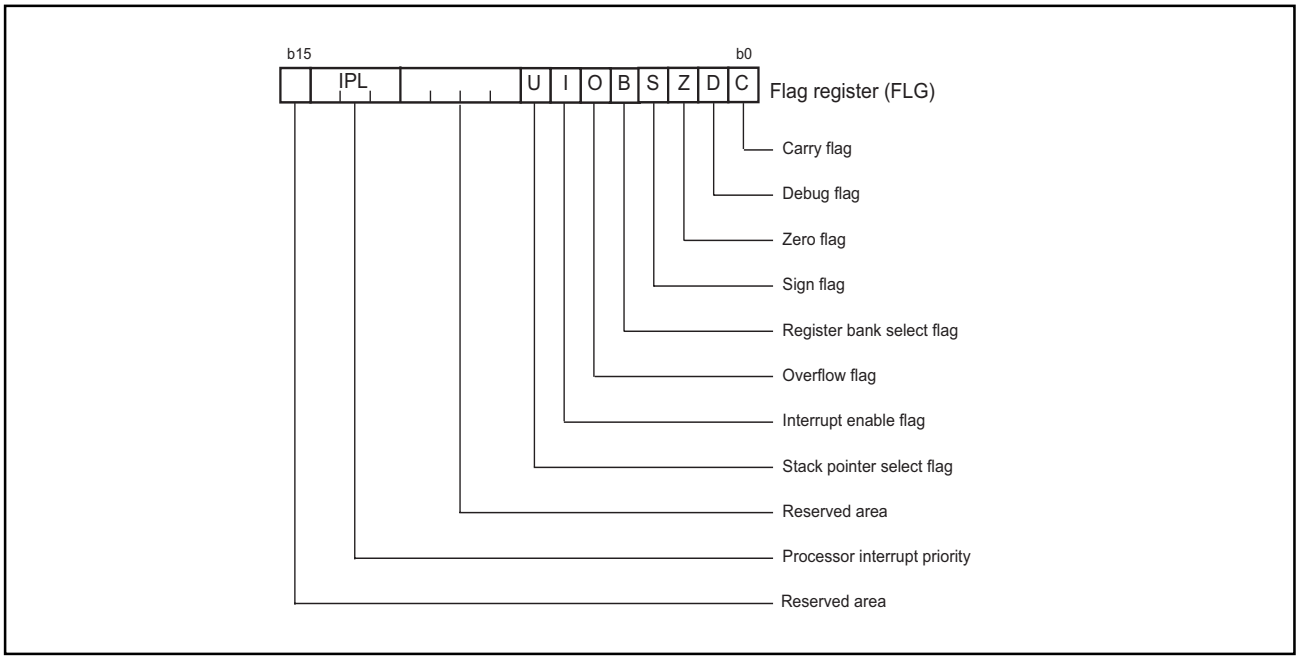



**Figure 1.7. Flag register**



Downloaded from Arrow.com.

## Reset

There are two kinds of resets: software and hardware. In both cases, operation is the same after the reset.

### Software Reset

Writing a "1" to bit 3 of the processor mode register 0 (address $0004_{16}$) applies a (software) reset to the microcomputer. A software reset has almost the same effect as a hardware reset with the following exceptions:

- The contents of internal RAM are preserved in a software reset.
- The contents of all USB and PLL SFR values are preserved in a software reset.
- For bit 0 and 1 of processor mode register 0 (address $0004_{16}$), and bit 1 of pull-up control register 1 (address $03FD_{16}$), the value after software reset will be different from the value after hardware reset.

When performing a software reset, select the main clock for the CPU clock source, and set the PM03 bit to "1" only when the main clock is stabilized.

### Hardware reset

When the supply voltage is in the range where operation is guaranteed, a reset is executed by holding the reset pin to "L" level (0.2VCC max.) for at least 20 cycles. When the reset pin level is then returned to the "H" level while the main clock is stable, the device exits reset and the program execution resumes from the address in the reset vector table.

Figure 1.8 shows an example reset circuit. Figure 1.9 shows the reset sequence.

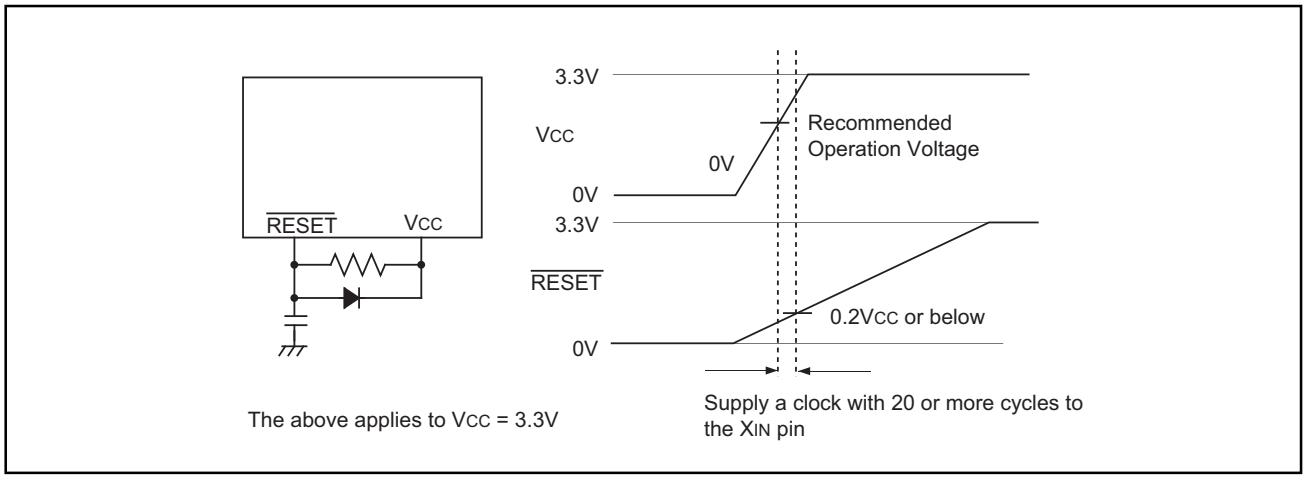



**Figure 1.8. Reset circuit**

### I/O Status during Reset

When the $\overline{\text{RESET}}$ pin level is "L", all ports change to input mode (floating) Table 1.5 shows the status of the other pins while the $\overline{\text{RESET}}$ pin level is "L".



Downloaded from Arrow.com.

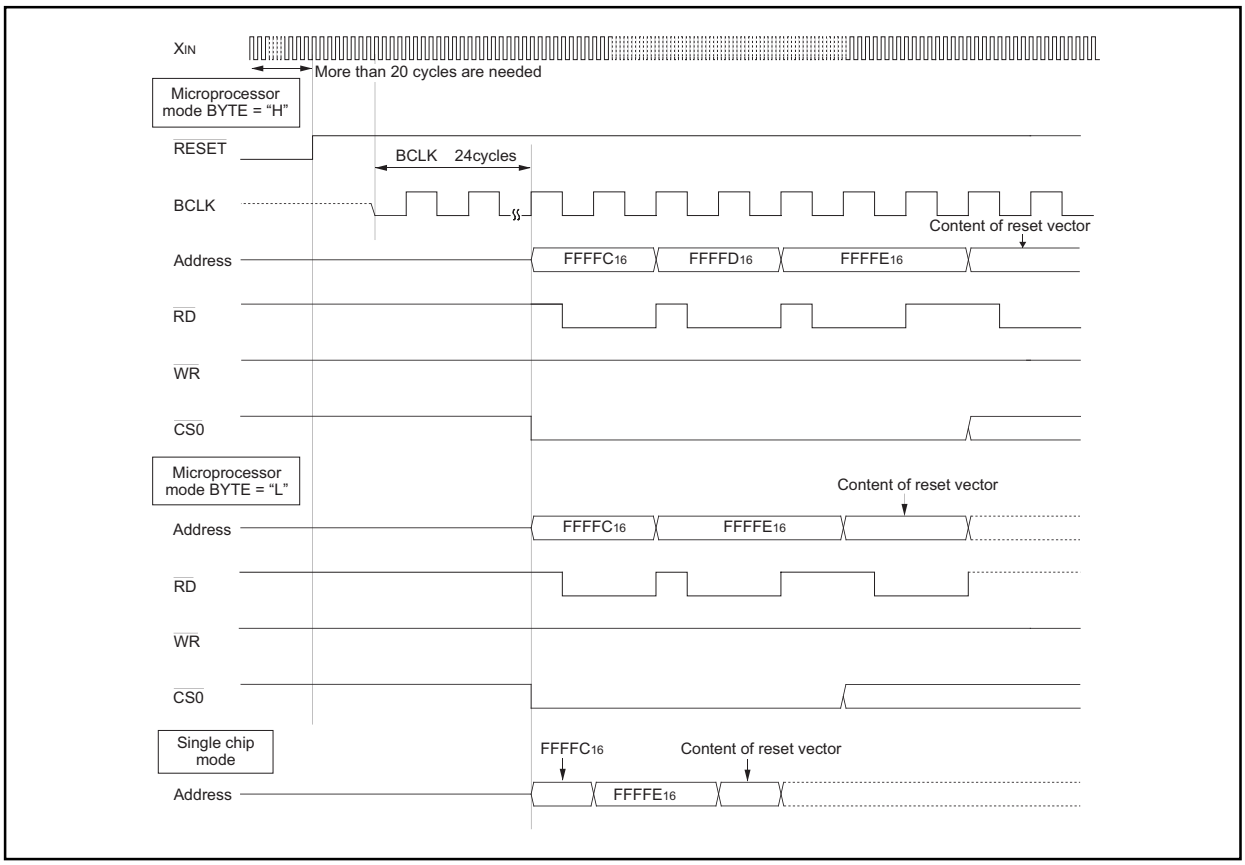



**Figure 1.9. Reset sequence**

**Table 1.5. Pin status when RESET pin level is "L"**

| Pin name | Status | | |
|---|---|---|---|
| | CNVss=Vss | CNVss=Vcc | |
| | | BYTE=Vss | BYTE=Vcc |
| $P_0$ | Input port (floating) | Data input (floating) | Data input (floating) |
| $P_1$ | Input port (floating) | Data input (floating) | Input port (floating) |
| $P_2$, $P_3$, $P4_0$ to $P4_3$ | Input port (floating) | Address output (undefined) | Address output (undefined) |
| $P4_4$ | Input port (floating) | $\overline{CS0}$ output ("H" level is output) | $\overline{CS0}$ output ("H" level is output) |
| $P4_5$ to $P4_7$ | Input port (floating) | Input port (floating) (pull-up resistor is on) | Input port (floating) (pull-up resistor is on) |
| $P5_0$ | Input port (floating) | $\overline{WR}$ output ("H" level is output) | $\overline{WR}$ output ("H" level is output) |
| $P5_1$ | Input port (floating) | $\overline{BHE}$ output (undefined) | $\overline{BHE}$ output (undefined) |
| $P5_2$ | Input port (floating) | $\overline{RD}$ output ("H" level is output) | $\overline{RD}$ output ("H" level is output) |
| $P5_3$ | Input port (floating) | BCLK output | BCLK output |
| $P5_4$ | Input port (floating) | HLDA output (The output value depends on the input to the $\overline{HOLD}$ pin) | HLDA output (The output value depends on the input to the $\overline{HOLD}$ pin) |
| $P5_5$ | Input port (floating) | $\overline{HOLD}$ input (floating) | $\overline{HOLD}$ input (floating) |
| $P5_6$ | Input port (floating) | ALE output ("L" level is output) | ALE output ("L" level is output) |
| $P5_7$ | Input port (floating) | $\overline{RDY}$ input (floating) | $\overline{RDY}$ input (floating) |
| P6, P7, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, P9, P10 | Input port (floating) | Input port (floating) | Input port (floating) |



Downloaded from Arrow.com.

## Special Function Registers

Tables 1.6 to 1.13 show the peripheral control registers, their addresses, names, acronyms, and values after reset.

**Table 1.6. SFR Map (1)**

| Address | Register name | Acronym | Value after reset |
|---|---|---|---|
| $0000_{16}$ | | | |
| $0001_{16}$ | | | |
| $0002_{16}$ | | | |
| $0003_{16}$ | | | |
| $0004_{16}$ | Processor mode register 0 (Note 3) | PM0 | $00_{16}$ |
| $0005_{16}$ | Processor mode register 1 | PM1 | 0 0 : : : : : 0 |
| $0006_{16}$ | System clock control register 0 | CM0 | $48_{16}$ |
| $0007_{16}$ | System clock control register 1 | CM1 | $20_{16}$ |
| $0008_{16}$ | Chip select control register | CSR | 0 0 0 0 0 0 0 1 |
| $0009_{16}$ | Address match interrupt enable register | AIER | : : : : : : 0 0 |
| $000A_{16}$ | Protect register | PRCR | : : : : : 0 0 0 |
| $000B_{16}$ | | | |
| $000C_{16}$ | USB control register | USBC | $00_{16}$ |
| $000D_{16}$ | | | |
| $000E_{16}$ | Watchdog timer start register | WDTS | |
| $000F_{16}$ | Watchdog timer control register | WDC | 0 0 0 ? ? ? ? ? |
| $0010_{16}$ | Address match interrupt register 0 | RMAD0 | $00_{16}$ |
| $0011_{16}$ | | | $00_{16}$ |
| $0012_{16}$ | | | : : : : 0 0 0 0 |
| $0013_{16}$ | | | |
| $0014_{16}$ | Address match interrupt register 1 | RMAD1 | $00_{16}$ |
| $0015_{16}$ | | | $00_{16}$ |
| $0016_{16}$ | | | : : : : 0 0 0 0 |
| $0017_{16}$ | | | |
| $0018_{16}$ | | | |
| $0019_{16}$ | | | |
| $001A_{16}$ | | | |
| $001B_{16}$ | Chip select expansion register | CSE | $00_{16}$ |
| $001C_{16}$ | | | |
| $001D_{16}$ | | | |
| $001E_{16}$ | Reserved | | |
| $001F_{16}$ | USB Attach/Detach register | USBAD | $00_{16}$ |
| $0020_{16}$ | DMA0 source pointer | SAR0 | |
| $0021_{16}$ | | | |
| $0022_{16}$ | | | |
| $0023_{16}$ | | | |
| $0024_{16}$ | DMA0 destination pointer | DAR0 | |
| $0025_{16}$ | | | |
| $0026_{16}$ | | | |
| $0027_{16}$ | | | |
| $0028_{16}$ | DMA0 transfer counter | TCR0 | |
| $0029_{16}$ | | | |
| $002A_{16}$ | | | |
| $002B_{16}$ | | | |
| $002C_{16}$ | DMA0 control register | DM0CON | 0 0 0 0 0 ? 0 0 |
| $002D_{16}$ | | | |
| $002E_{16}$ | | | |
| $002F_{16}$ | | | |
| $0030_{16}$ | DMA1 source pointer | SAR1 | |
| $0031_{16}$ | | | |
| $0032_{16}$ | | | |
| $0033_{16}$ | | | |
| $0034_{16}$ | DMA1 destination pointer | DAR1 | |
| $0035_{16}$ | | | |
| $0036_{16}$ | | | |
| $0037_{16}$ | | | |
| $0038_{16}$ | DMA1 transfer counter | TCR1 | |
| $0039_{16}$ | | | |
| $003A_{16}$ | | | |
| $003B_{16}$ | | | |
| $003C_{16}$ | DMA1 control register | DM1CON | 0 0 0 0 0 ? 0 0 |

? : Undefined

: Nothing is mapped to this bit

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.
Note 3: For hardware reset, when Vcc is applied to the CNVss pin, it is $03_{16}$ at reset. For software reset, the contents of bit 0 and 1 are preserved as before the reset.



Downloaded from Arrow.com.

**Table 1.7. SFR Map (2)**

| Address | Register name | Acronym | Value after reset | |
|---|---|---|---|---|
| 0040$_{16}$ | | | | |
| 0041$_{16}$ | Key input interrupt control register | KUPIC | ? 0 0 0 | |
| 0042$_{16}$ | UART2 receive/ACK interrupt control register | S2RIC | ? 0 0 0 | |
| 0043$_{16}$ | UART1/3 Bus collision interrupt control register | S13BCNIC | ? 0 0 0 | |
| 0044$_{16}$ | INT1 interrupt control register | INT1IC | 0 0 ? 0 0 0 | Polarity |
| 0045$_{16}$ | Timer A1 interrupt control register | TA1IC | ? 0 0 0 | |
| 0046$_{16}$ | USB Endpoint 0 interrupt control register | EP0IC | ? 0 0 0 | |
| 0047$_{16}$ | Timer A2 interrupt control register | TA2IC | ? 0 0 0 | |
| 0048$_{16}$ | UART1 receive/ACK/SSI1 interrupt control register | S1RIC | 0 0 ? 0 0 0 | Polarity |
| 0049$_{16}$ | UART0/2 Bus collision interrupt control register | S02BCNIC | 0 0 ? 0 0 0 | Polarity |
| 004A$_{16}$ | UART0 receive/ACK/SSI0 interrupt control register | S0RIC | ? 0 0 0 | |
| 004B$_{16}$ | AD conversion interrupt control register | ADIC | ? 0 0 0 | |
| 004C$_{16}$ | DMA0 interrupt conrol register | DM0IC | ? 0 0 0 | |
| 004D$_{16}$ | UART3 transmit/NACK interrupt control register | S3TIC | ? 0 0 0 | |
| 004E$_{16}$ | DMA1 interrupt control register | DM1IC | ? 0 0 0 | |
| 004F$_{16}$ | UART2 transmit/NACK interrupt control register | S2TIC | ? 0 0 0 | |
| 0050$_{16}$ | DMA2 interrupt control register | DM2IC | ? 0 0 0 | |
| 0051$_{16}$ | UART1 transmit/NACK/SSI1 interrupt control register | S1TIC | ? 0 0 0 | |
| 0052$_{16}$ | DMA3 interrupt control register | DM3IC | ? 0 0 0 | |
| 0053$_{16}$ | UART0 transmit/NACK/SSI0 interrupt control register | S0TIC | ? 0 0 0 | |
| 0054$_{16}$ | Timer A0 interrupt control register | TA0IC | ? 0 0 0 | |
| 0055$_{16}$ | UART3 receive/ACK interrupt control register | S3RIC | ? 0 0 0 | |
| 0056$_{16}$ | USB suspend interrupt control register | SUSPIC | ? 0 0 0 | |
| 0057$_{16}$ | Timer A3 interrupt control register | TA3IC | ? 0 0 0 | |
| 0058$_{16}$ | USB resume interrupt control register | RSMIC | ? 0 0 0 | |
| 0059$_{16}$ | Timer A4 interrupt control register | TA4IC | ? 0 0 0 | |
| 005A$_{16}$ | USB reset interrupt control register | RSTIC | ? 0 0 0 | |
| 005B$_{16}$ | USB SOF interrupt control register | SOFIC | ? 0 0 0 | |
| 005C$_{16}$ | USB Vbus detect interrupt control register | VBDIC | ? 0 0 0 | |
| 005D$_{16}$ | USB function interrupt control register | USBFIC | ? 0 0 0 | |
| 005E$_{16}$ | INT2 interrupt control register | INT2IC | 0 0 ? 0 0 0 | Polarity |
| 005F$_{16}$ | INT0 interrupt control register | INT0IC | 0 0 ? 0 0 0 | Polarity |
| 0180$_{16}$ | | | | |
| 0181$_{16}$ | DMA2 source pointer | SAR2 | | |
| 0182$_{16}$ | | | | |
| 0183$_{16}$ | | | | |
| 0184$_{16}$ | | | | |
| 0185$_{16}$ | DMA2 destination pointer | DAR2 | | |
| 0186$_{16}$ | | | | |
| 0187$_{16}$ | | | | |
| 0188$_{16}$ | DMA2 transfer counter | TCR2 | | |
| 0189$_{16}$ | | | | |
| 018A$_{16}$ | | | | |
| 018B$_{16}$ | | | | |
| 018C$_{16}$ | DMA2 control register | DM2CON | 0 0 0 0 0 ? 0 0 | |
| 018D$_{16}$ | | | | |
| 018E$_{16}$ | | | | |
| 018F$_{16}$ | | | | |
| 0190$_{16}$ | | | | |
| 0191$_{16}$ | DMA3 source pointer | SAR3 | | |
| 0192$_{16}$ | | | | |
| 0193$_{16}$ | | | | |
| 0194$_{16}$ | | | | |
| 0195$_{16}$ | DMA3 destination pointer | DAR3 | | |
| 0196$_{16}$ | | | | |
| 0197$_{16}$ | | | | |
| 0198$_{16}$ | DMA3 transfer counter | TCR3 | | |
| 0199$_{16}$ | | | | |
| 019A$_{16}$ | | | | |
| 019B$_{16}$ | | | | |
| 019C$_{16}$ | DMA3 control register | DM3CON | 0 0 0 0 0 ? 0 0 | |
| 019D$_{16}$ | | | | |
| 019E$_{16}$ | | | | |
| 019F$_{16}$ | | | | |

? : Undefined

: Nothing is mapped to this bit

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.



Downloaded from Arrow.com.

**Table 1.8. SFR Map (3)**

| Address | Register name | Acronym | Value after reset | |
|---|---|---|---|---|
| $0280_{16}$ $0281_{16}$ | USB address register | USBA | $0000_{16}$ | |
| $0282_{16}$ $0283_{16}$ | USB power management register | USBPM | $0000_{16}$ | |
| $0284_{16}$ $0285_{16}$ | USB interrupt status register | USBIS | $0000_{16}$ | |
| $0286_{16}$ $0287_{16}$ | USB interrupt clear register | USBIC | $0000_{16}$ | |
| $0288_{16}$ $0289_{16}$ | USB interrupt enable register | USBIE | $01FF_{16}$ | |
| $028A_{16}$ $028B_{16}$ | USB frame number register | USBFN | $0000_{16}$ | |
| $028C_{16}$ $028D_{16}$ | USB ISO control register | USBISOC | $0000_{16}$ | |
| $028E_{16}$ $028F_{16}$ | USB endpoint enable register | USBEPEN | $0000_{16}$ | |
| $0290_{16}$ $0291_{16}$ | USB DMA0 request register | USBDMA0 | $0000_{16}$ | |
| $0292_{16}$ $0293_{16}$ | USB DMA1 request register | USBDMA1 | $0000_{16}$ | |
| $0294_{16}$ $0295_{16}$ | USB DMA2 request register | USBDMA2 | $0000_{16}$ | |
| $0296_{16}$ $0297_{16}$ | USB DMA3 request register | USBDMA3 | $0000_{16}$ | |
| $0298_{16}$ $0299_{16}$ | USB EP0 control/status register | EP0CS | $2000_{16}$ | |
| $029A_{16}$ $029B_{16}$ | USB EP0 max packet size register | EP0MP | $0008_{16}$ | |
| $029C_{16}$ $029D_{16}$ | USB EP0 write count register | EP0WC | $0000_{16}$ | |
| $029E_{16}$ $029F_{16}$ | USB EP1 IN control/status register | EP1ICS | $0003_{16}$ | |
| $02A0_{16}$ $02A1_{16}$ | USB EP1 IN max packet size register | EP1IMP | $0000_{16}$ | |
| $02A2_{16}$ $02A3_{16}$ | USB EP1 IN FIFO configuration register | EP1IFC | $0000_{16}$ | |
| $02A4_{16}$ $02A5_{16}$ | USB EP2 IN control/status register | EP2ICS | $0003_{16}$ | |
| $02A6_{16}$ $02A7_{16}$ | USB EP2 IN max packet size register | EP2IMP | $0000_{16}$ | |
| $02A8_{16}$ $02A9_{16}$ | USB EP2 IN FIFO configuration register | EP2IFC | $0000_{16}$ | |
| $02AA_{16}$ $02AB_{16}$ | USB EP3 IN control/status register | EP3ICS | $0003_{16}$ | |
| $02AC_{16}$ $02AD_{16}$ | USB EP3 IN max packet size register | EP3IMP | $0000_{16}$ | |
| $02AE_{16}$ $02AF_{16}$ | USB EP3 IN FIFO configuration register | EP3IFC | $0000_{16}$ | |
| $02B0_{16}$ $02B1_{16}$ | USB EP4 IN control/status register | EP4ICS | $0003_{16}$ | |
| $02B2_{16}$ $02B3_{16}$ | USB EP4 IN max packet size register | EP4IMP | $0000_{16}$ | |
| $02B4_{16}$ $02B5_{16}$ | USB EP4 IN FIFO configuration register | EP4IFC | $0000_{16}$ | |
| $02B6_{16}$ $02B7_{16}$ | USB EP1 OUT control/status register | EP1OCS | $0000_{16}$ | |
| $02B8_{16}$ $02B9_{16}$ | USB EP1 OUT max packet size register | EP1OMP | $0000_{16}$ | |
| $02BA_{16}$ $02BB_{16}$ | USB EP1 OUT write count register | EP1WC | $0000_{16}$ | |
| $02BC_{16}$ $02BD_{16}$ | USB EP1 OUT FIFO configuration register | EP1OFC | $0000_{16}$ | |
| $02BE_{16}$ $02BF_{16}$ | USB EP2 OUT control /status register | EP2OCS | $0000_{16}$ | |

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.



Downloaded from Arrow.com.

**Table 1.9. SFR Map (4)**

| Address | Register name | Acronym | Value after reset | |
|---|---|---|---|---|
| $02C0_{16}$ $02C1_{16}$ | USB EP2 OUT max packet size register | EP2OMP | $0000_{16}$ | |
| $02C2_{16}$ $02C3_{16}$ | USB EP2 OUT write count register | EP2WC | $0000_{16}$ | |
| $02C4_{16}$ $02C5_{16}$ | USB EP2 OUT FIFO configuration register | EP2OFC | $0000_{16}$ | |
| $02C6_{16}$ $02C7_{16}$ | USB EP3 OUT control/status register | EP3OCS | $0000_{16}$ | |
| $02C8_{16}$ $02C9_{16}$ | USB EP3 OUT max packet size register | EP3OMP | $0000_{16}$ | |
| $02CA_{16}$ $02CB_{16}$ | USB EP3 OUT write count register | EP3WC | $0000_{16}$ | |
| $02CC_{16}$ $02CD_{16}$ | USB EP3 OUT FIFO configuration register | EP3OFC | $0000_{16}$ | |
| $02CE_{16}$ $02CF_{16}$ | USB EP4 OUT control/status register | EP4OCS | $0000_{16}$ | |
| $02D0_{16}$ $02D1_{16}$ | USB EP4 OUT max packet size register | EP4OMP | $0000_{16}$ | |
| $02D2_{16}$ $02D3_{16}$ | USB EP4 OUT write count register | EP4WC | $0000_{16}$ | |
| $02D4_{16}$ $02D5_{16}$ | USB EP4 OUT FIFO configuration register | EP4OFC | $0000_{16}$ | |
| $02D6_{16}$ | | | | |
| $02D7_{16}$ | | | | |
| $02D8_{16}$ | USB reserved | | | |
| $02D9_{16}$ | USB reserved | | | |
| $02DA_{16}$ | USB reserved | | | |
| $02DB_{16}$ | USB reserved | | | |
| $02DC_{16}$ | USB reserved | | | |
| $02DD_{16}$ | USB reserved | | | |
| $02DE_{16}$ | USB reserved | | | |
| $02DF_{16}$ | USB reserved | | | |
| $02E0_{16}$ $02E1_{16}$ | USB EP0 IN FIFO | EP0I | | |
| $02E2_{16}$ $02E3_{16}$ | USB EP0 OUT FIFO | EP0O | | |
| $02E4_{16}$ $02E5_{16}$ | USB EP1 IN FIFO | EP1I | | |
| $02E6_{16}$ $02E7_{16}$ | USB EP1 OUT FIFO | EP1O | | |
| $02E8_{16}$ $02E9_{16}$ | USB EP2 IN FIFO | EP2I | | |
| $02EA_{16}$ $02EB_{16}$ | USB EP2 OUT FIFO | EP2O | | |
| $02EC_{16}$ $02ED_{16}$ | USB EP3 IN FIFO | EP3I | | |
| $02EE_{16}$ $02EF_{16}$ | USB EP3 OUT FIFO | EP3O | | |
| $02F0_{16}$ $02F1_{16}$ | USB EP4 IN FIFO | EP4I | | |
| $02F2_{16}$ $02F3_{16}$ | USB EP4 OUT FIFO | EP4O | | |
| $02F4_{16}$ | | | | |
| $02F5_{16}$ | | | | |
| $02F6_{16}$ | | | | |
| $02F7_{16}$ | Flash memory control register 0 (Note 3) | FMR0 | $01_{16}$ | |
| $02F8_{16}$ | | | | |
| $02F9_{16}$ | | | | |
| $02FA_{16}$ | | | | |
| $02FB_{16}$ | | | | |
| $02FC_{16}$ | | | | |
| $02FD_{16}$ | | | | |
| $02FE_{16}$ | Reserved | | | |
| $02FF_{16}$ | Reserved | | | |

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.
Note 3: This register exists only in the flash memory version.



Downloaded from Arrow.com.

**Table 1.10. SFR Map (5)**

| Address | Register name | Acronym | Value after reset | |
|---|---|---|---|---|
| $0300_{16}$ | | | | |
| $0301_{16}$ | | | | |
| $0302_{16}$ | | | | |
| $0303_{16}$ | | | | |
| $0304_{16}$ | | | | |
| $0305_{16}$ | | | | |
| $0306_{16}$ | | | | |
| $0307_{16}$ | | | | |
| $0308_{16}$ | | | | |
| $0309_{16}$ | | | | |
| $030A_{16}$ | | | | |
| $030B_{16}$ | | | | |
| $030C_{16}$ | | | | |
| $030D_{16}$ | | | | |
| $030E_{16}$ | | | | |
| $030F_{16}$ | | | | |
| $0310_{16}$ | Serial Sound Interface 0 mode register 0 | SSI0MR0 | $00_{16}$ | |
| $0311_{16}$ | Serial Sound Interface 0 mode register 1 | SSI0MR1 | $00_{16}$ | |
| $0312_{16}$ | Reserved | | | |
| $0313_{16}$ | Reserved | | | |
| $0314_{16}$ $0315_{16}$ | Serial Sound Interface 0 transmit buffer register | SSI0TXB | $0000_{16}$ | |
| $0316_{16}$ $0317_{16}$ | Serial Sound Interface 0 receive buffer register | SSI0RXB | $0000_{16}$ | |
| $0318_{16}$ $0319_{16}$ | Serial Sound Interface 0 rate feedback register | SSI0RF | $0000_{16}$ | |
| $031A_{16}$ | Reserved | | | |
| $031B_{16}$ | Reserved | | | |
| $031C_{16}$ | | | | |
| $031D_{16}$ | | | | |
| $031E_{16}$ | | | | |
| $031F_{16}$ | | | | |
| $0320_{16}$ | | | | |
| $0321_{16}$ | | | | |
| $0322_{16}$ | | | | |
| $0323_{16}$ | | | | |
| $0324_{16}$ | UART3 special mode register 4 | U3SMR4 | $00_{16}$ | |
| $0325_{16}$ | UART3 special mode register 3 | U3SMR3 | $00_{16}$ | |
| $0326_{16}$ | UART3 special mode register 2 | U3SMR2 | $00_{16}$ | |
| $0327_{16}$ | UART3 special mode register | U3SMR | $00_{16}$ | |
| $0328_{16}$ | UART3 transmit / receive mode register | U3MR | $00_{16}$ | |
| $0329_{16}$ | UART3 bit rate generator | U3BRG | | |
| $032A_{16}$ $032B_{16}$ | UART3 transmit buffer register | U3TB | | |
| $032C_{16}$ | UART3 transmit / receive control register 0 | U3C0 | $08_{16}$ | |
| $032D_{16}$ | UART3 transmit / receive control register 1 | U3C1 | $02_{16}$ | |
| $032E_{16}$ $032F_{16}$ | UART3 receive buffer register | U3RB | | |
| $0330_{16}$ | | | | |
| $0331_{16}$ | | | | |
| $0332_{16}$ | | | | |
| $0333_{16}$ | | | | |
| $0334_{16}$ | UART2 special mode register 4 | U2SMR4 | $00_{16}$ | |
| $0335_{16}$ | UART2 special mode register 3 | U2SMR3 | $00_{16}$ | |
| $0336_{16}$ | UART2 special mode register 2 | U2SMR2 | $00_{16}$ | |
| $0337_{16}$ | UART2 special mode register | U2SMR | $00_{16}$ | |
| $0338_{16}$ | UART2 transmit / receive mode register | U2MR | $00_{16}$ | |
| $0339_{16}$ | UART2 bit rate generator | U2BRG | | |
| $033A_{16}$ $033B_{16}$ | UART2 transmit buffer register | U2TB | | |
| $033C_{16}$ | UART2 transmit / receive control register 0 | U2C0 | $08_{16}$ | |
| $033D_{16}$ | UART2 transmit / receive control register 1 | U2C1 | $02_{16}$ | |
| $033E_{16}$ $033F_{16}$ | UART2 receive buffer register | U2RB | | |

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.



Downloaded from Arrow.com.

**Table 1.11. SFR Map (6)**

| Address | Register name | Acronym | Value after reset | |
|---|---|---|---|---|
| $0340_{16}$ | | | | |
| $0341_{16}$ | | | | |
| $0342_{16}$ | | | | |
| $0343_{16}$ | | | | |
| $0344_{16}$ | | | | |
| $0345_{16}$ | | | | |
| $0346_{16}$ | | | | |
| $0347_{16}$ | | | | |
| $0348_{16}$ | | | | |
| $0349_{16}$ | | | | |
| $034A_{16}$ | | | | |
| $034B_{16}$ | | | | |
| $034C_{16}$ | | | | |
| $034D_{16}$ | | | | |
| $034E_{16}$ | | | | |
| $034F_{16}$ | | | | |
| $0350_{16}$ | | | | |
| $0351_{16}$ | | | | |
| $0352_{16}$ | | | | |
| $0353_{16}$ | | | | |
| $0354_{16}$ | | | | |
| $0355_{16}$ | | | | |
| $0356_{16}$ | | | | |
| $0357_{16}$ | | | | |
| $0358_{16}$ | | | | |
| $0359_{16}$ | | | | |
| $035A_{16}$ | | | | |
| $035B_{16}$ | | | | |
| $035C_{16}$ | | | | |
| $035D_{16}$ | | | | |
| $035E_{16}$ | | | | |
| $035F_{16}$ | Interrupt cause select register | IFSR | $00_{16}$ | |
| $0360_{16}$ | | | | |
| $0361_{16}$ | | | | |
| $0362_{16}$ | | | | |
| $0363_{16}$ | | | | |
| $0364_{16}$ | UART1 special mode register 4 | U1SMR4 | $00_{16}$ | |
| $0365_{16}$ | UART1 special mode register 3 | U1SMR3 | $00_{16}$ | |
| $0366_{16}$ | UART1 special mode register 2 | U1SMR2 | $00_{16}$ | |
| $0367_{16}$ | UART1 special mode register | U1SMR | $00_{16}$ | |
| $0368_{16}$ | UART1 transmit / receive mode register | U1MR | $00_{16}$ | |
| $0369_{16}$ | UART1 bit rate generator | U1BRG | | |
| $036A_{16}$ $036B_{16}$ | UART1 transmit buffer register | U1TB | | |
| $036C_{16}$ | UART1 transmit / receive control register 0 | U1C0 | $08_{16}$ | |
| $036D_{16}$ | UART1 transmit / receive control register 1 | U1C1 | $02_{16}$ | |
| $036E_{16}$ $036F_{16}$ | UART1 receive buffer register | U1RB | | |
| $0370_{16}$ | Serial Sound Interface 1 mode register 0 | SSI1MR0 | $00_{16}$ | |
| $0371_{16}$ | Serial Sound Interface 1 mode register 1 | SSI1MR1 | $00_{16}$ | |
| $0372_{16}$ | Reserved | | | |
| $0373_{16}$ | Reserved | | | |
| $0374_{16}$ $0375_{16}$ | Serial Sound Interface 1 transmit buffer register | SSI1TXB | $0000_{16}$ | |
| $0376_{16}$ $0377_{16}$ | Serial Sound Interface 1 receive buffer register | SSI1RXB | $0000_{16}$ | |
| $0378_{16}$ $0379_{16}$ | Serial Sound Interface 1 rate feedback register | SSI1RF‘ | $0000_{16}$ | |
| $037A_{16}$ | Reserved | | | |
| $037B_{16}$ | Reserved | | | |
| $037C_{16}$ | | | | |
| $037D_{16}$ | | | | |
| $037E_{16}$ | | | | |
| $037F_{16}$ | | | | |

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.



Downloaded from Arrow.com.

**Table 1.12. SFR Map (7)**

| Address | Register name | Acronym | Value after reset |
|---|---|---|---|
| 0380₁₆ | Count start flag | TABSR | 0 0 0 0 0 |
| 0381₁₆ | Clock prescaler reset flag | CPSRF | 0 |
| 0382₁₆ | One-shot start flag | ONSF | 0 0 0 0 0 0 0 |
| 0383₁₆ | Trigger select register | TRGSR | 00₁₆ |
| 0384₁₆ | Up-down flag | UDF | 00₁₆ |
| 0385₁₆ | | | |
| 0386₁₆ 0387₁₆ | Timer A0 | TA0 | |
| 0388₁₆ 0389₁₆ | Timer A1 | TA1 | |
| 038A₁₆ 038B₁₆ | Timer A2 | TA2 | |
| 038C₁₆ 038D₁₆ | Timer A3 | TA3 | |
| 038E₁₆ 038F₁₆ | Timer A4 | TA4 | |
| 0390₁₆ | | | |
| 0391₁₆ | | | |
| 0392₁₆ | | | |
| 0393₁₆ | | | |
| 0394₁₆ | | | |
| 0395₁₆ | | | |
| 0396₁₆ | Timer A0 mode register | TA0MR | 00₁₆ |
| 0397₁₆ | Timer A1 mode register | TA1MR | 00₁₆ |
| 0398₁₆ | Timer A2 mode register | TA2MR | 00₁₆ |
| 0399₁₆ | Timer A3 mode register | TA3MR | 00₁₆ |
| 039A₁₆ | Timer A4 mode register | TA4MR | 00₁₆ |
| 039B₁₆ | | | |
| 039C₁₆ | | | |
| 039D₁₆ | | | |
| 039E₁₆ | | | |
| 039F₁₆ | | | |
| 03A0₁₆ | | | |
| 03A1₁₆ | | | |
| 03A2₁₆ | | | |
| 03A3₁₆ | | | |
| 03A4₁₆ | UART0 special mode register 4 | U0SMR4 | 00₁₆ |
| 03A5₁₆ | UART0 special mode register 3 | U0SMR3 | 00₁₆ |
| 03A6₁₆ | UART0 special mode register 2 | U0SMR2 | 00₁₆ |
| 03A7₁₆ | UART0 special mode register | U0SMR | 00₁₆ |
| 03A8₁₆ | UART0 transmit / receive mode register | U0MR | 00₁₆ |
| 03A9₁₆ | UART0 bit rate generator | U0BRG | |
| 03AA₁₆ 03AB₁₆ | UART0 transmit buffer register | U0TB | |
| 03AC₁₆ | UART0 transmit / receive control register 0 | U0C0 | 08₁₆ |
| 03AD₁₆ | UART0 transmit / receive control register 1 | U0C1 | 02₁₆ |
| 03AE₁₆ 03AF₁₆ | UART0 receive buffer register | U0RB | |
| 03B0₁₆ | DMA2 cause select register | DM2SL | 00₁₆ |
| 03B1₁₆ | | | |
| 03B2₁₆ | DMA3 cause select register | DM3SL | 00₁₆ |
| 03B3₁₆ | | | |
| 03B4₁₆ | CRC snoop address register | CRCSAR | ? ? ? ? ? ? ? ? |
| 03B5₁₆ | | | 0 0 ? ? |
| 03B6₁₆ | CRC mode register | CRCMR | 0 0 |
| 03B7₁₆ | | | |
| 03B8₁₆ | DMA0 cause select register | DM0SL | 00₁₆ |
| 03B9₁₆ | | | |
| 03BA₁₆ | DMA1 cause select register | DM1SL | 00₁₆ |
| 03BB₁₆ | | | |
| 03BC₁₆ 03BD₁₆ | CRC data register | CRCD | |
| 03BE₁₆ | CRC input register | CRCIN | |
| 03BF₁₆ | | | |

? : Undefined

: Nothing is mapped to this bit

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.



Downloaded from Arrow.com.

**Table 1.13. SFR Map (8)**

| Address | Register name | Acronym | Value after reset |
|---|---|---|---|
| 03C0$_{16}$ 03C1$_{16}$ | AD register 0 | AD0 | |
| 03C2$_{16}$ 03C3$_{16}$ | AD register 1 | AD1 | |
| 03C4$_{16}$ 03C5$_{16}$ | AD register 2 | AD2 | |
| 03C6$_{16}$ 03C7$_{16}$ | AD register 3 | AD3 | |
| 03C8$_{16}$ 03C9$_{16}$ | AD register 4 | AD4 | |
| 03CA$_{16}$ 03CB$_{16}$ | AD register 5 | AD5 | |
| 03CC$_{16}$ 03CD$_{16}$ | AD register 6 | AD6 | |
| 03CE$_{16}$ 03CF$_{16}$ | AD register 7 | AD7 | |
| 03D0$_{16}$ | | | |
| 03D1$_{16}$ | | | |
| 03D2$_{16}$ | | | |
| 03D3$_{16}$ | | | |
| 03D4$_{16}$ | AD control register 2 | ADCON2 | 0 |
| 03D5$_{16}$ | | | |
| 03D6$_{16}$ | AD conrol register 0 | ADCON0 | 0 0 0 0 0 ? ? ? |
| 03D7$_{16}$ | AD conrol register 1 | ADCON1 | $00_{16}$ |
| 03D8$_{16}$ | | | |
| 03D9$_{16}$ | | | |
| 03DA$_{16}$ | | | |
| 03DB$_{16}$ | Frequency synthesizer clock control | FSCCR | $00_{16}$ |
| 03DC$_{16}$ | Frequency synthesizer control | FSC | $60_{16}$ |
| 03DD$_{16}$ | Frequency synthesizer multiplier control | FSM | $FF_{16}$ |
| 03DE$_{16}$ | Frequency synthesizer prescaler control | FSP | $FF_{16}$ |
| 03DF$_{16}$ | Frequency synthesizer divider | FSD | $FF_{16}$ |
| 03E0$_{16}$ | Port P0 | P0 | |
| 03E1$_{16}$ | Port P1 | P1 | |
| 03E2$_{16}$ | Port P0 direction register | PD0 | $00_{16}$ |
| 03E3$_{16}$ | Port P1 direction register | PD1 | $00_{16}$ |
| 03E4$_{16}$ | Port P2 | P2 | |
| 03E5$_{16}$ | Port P3 | P3 | |
| 03E6$_{16}$ | Port P2 direction register | PD2 | $00_{16}$ |
| 03E7$_{16}$ | Port P3 direction register | PD3 | $00_{16}$ |
| 03E8$_{16}$ | Port P4 | P4 | |
| 03E9$_{16}$ | Port P5 | P5 | |
| 03EA$_{16}$ | Port P4 direction register | PD4 | $00_{16}$ |
| 03EB$_{16}$ | Port P5 direction register | PD5 | $00_{16}$ |
| 03EC$_{16}$ | Port P6 | P6 | |
| 03ED$_{16}$ | Port P7 | P7 | |
| 03EE$_{16}$ | Port P6 direction register | PD6 | $00_{16}$ |
| 03EF$_{16}$ | Port P7 direction register | PD7 | $00_{16}$ |
| 03F0$_{16}$ | Port P8 | P8 | |
| 03F1$_{16}$ | Port P9 | P9 | 0 0 0 |
| 03F2$_{16}$ | Port P8 direction register | PD8 | 0 0 0 0 0 0 0 |
| 03F3$_{16}$ | Port P9 direction register | PD9 | 0 0 0 |
| 03F4$_{16}$ | Port P10 | P10 | |
| 03F5$_{16}$ | | | |
| 03F6$_{16}$ | Port P10 direction register | PD10 | $00_{16}$ |
| 03F7$_{16}$ | | | |
| 03F8$_{16}$ | | | |
| 03F9$_{16}$ | Key-input mode register | KUPM | $00_{16}$ |
| 03FA$_{16}$ | P7 drive capacity | P7DR | $00_{16}$ |
| 03FB$_{16}$ | | | |
| 03FC$_{16}$ | Pull-up control register 0 | PUR0 | $00_{16}$ |
| 03FD$_{16}$ | Pull-up control register 1 (Note 3) | PUR1 | $00_{16}$ |
| 03FE$_{16}$ | Pull-up control register 2 | PUR2 | 0 0 0 0 0 |
| 03FF$_{16}$ | Port control register | PCR | 0 |

? : Undefined

: Nothing is mapped to this bit

Note 1: The contents of other registers and RAM is undefined when the microcomputer is reset. The initial value must therefore be set.
Note 2: Locations in the SFR area where nothing is assigned are reserved areas. Do not access these areas for read or write.
Note 3: For hardware reset, when Vcc is applied to the CNVss pin, it is $02_{16}$ at reset. For a software reset, if bit 1 and bit 0 of the processor mode register 0 (address $0004_{16}$) are [$10_2$] or [$11_2$], then it becomes $02_{16}$ at a reset.



Downloaded from Arrow.com.

## Processor Modes

One of three processor modes can be selected: single-chip mode, memory expansion mode, and microprocessor mode. The functions of some pins, the memory map, and the access space differ according to the selected processor mode. Figure 1.10 shows the processor mode register 0 and 1.

- **Single-chip mode**

In single-chip mode, only internal memory space (SFR, internal RAM, and internal ROM) can be accessed. However, if microprocessor mode is set ("H" applied to the CNVss pin) when coming out of reset, the internal ROM cannot be accessed even if the CPU shifts to single-chip mode.

Ports P0 to P10 can be used as programmable I/O ports or I/O ports for the internal peripheral functions.

- **Memory expansion mode**

In memory expansion mode, external memory can be accessed in addition to the internal memory space (SFR, internal RAM, and internal ROM). However, if microprocessor mode is set ("H" applied to CNVss pin) when coming out of a reset, the internal ROM cannot be accessed even if the CPU shifts to memory expansion mode.

In memory expansion mode, some of the pins function as the address bus, the data bus, and as control signals. The number of pins assigned to these functions depends on the bus and register settings. (See "Bus Settings"for details.)

- **Microprocessor mode**

In microprocessor mode, the SFR, internal RAM, and external memory space can be accessed. However, the internal ROM area cannot be accessed.

In this mode, some of the pins function as the address bus, the data bus, and as control signals. The number of pins assigned to these functions depends on the bus and register settings. (See "Bus Settings" for details).

### Setting Processor Modes

The processor mode is set using the CNVss pin and the processor mode bits (bits 1 and 0 at address $0004_{16}$). Do not set the processor mode bits t o "$10_2$".

Regardless of the level of the CNVss pin, changing the processor mode bits selects the mode. However, the processor mode bits cannot be changed to "$01_2$" (memory expansion mode) or "$11_2$" (microprocessor mode) at the same time the PM07-PM02 bits are rewritten. Also do not attempt to change to or from microprocessor mode within the program stored in the internal ROM area.

- **Applying Vss to CNVss pin**

The microcomputer begins operation in single-chip mode after being reset. Memory expansion mode is selected by writing "$01_2$" to the processor mode bits in the Processor Mode register 0 ($0004_{16}$).

- **Applying Vcc to CNVss pin**

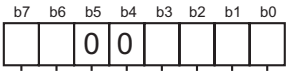
The microcomputer starts to operate in microprocessor mode after being reset.

Figure 1.11 shows the applicable memory maps for each mode. Figure 1.12 shows the memory maps and chip-select areas in normal mode.



Downloaded from Arrow.com.

## Processor mode register 0 (Note 1)

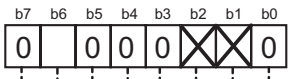



| Symbol | Address | When reset |
|---|---|---|
| PM0 | 0004₁₆ | 00₁₆ (Note 2) |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PM00 | Processor mode bit | b1 b0<br>0 0: Single-chip mode<br>0 1: Memory expansion mode<br>1 0: Inhibited<br>1 1: Microprocessor mode | | |
| PM01 | | | | |
| PM02 | R/W mode select bit | 0: $\overline{RD}$,$\overline{BHE}$,$\overline{WR}$<br>1: $\overline{RD}$,$\overline{WRH}$,$\overline{WRL}$ | | |
| PM03 | Software reset bit | The device is reset when this bit is set to "1". The value of this bit is "0" when read. | | |
| PM04 | Reserved | Must always be set to "0" | | |
| PM05 | | | | |
| PM06 | Port P4₀ to P4₃ function select bit (Note 3) | 0 : Address output<br>1 : Port function (Address is not output) | | |
| PM07 | BCLK output disable bit | 0 : BCLK is output<br>1 : BCLK is not output (Pin is left floating) | | |

Note 1: Set bit 1 of the protect register (address 000A₁₆) to "1" before writing new values to this register.
Note 2: For hardware reset: If Vcc voltage is applied to the CNVss pin, the value of this register when reset is 03₁₆. (PM00 and PM01 are both set to "1".)
For software reset: the value of PM00 and PM01 are preserved as before the reset.
Note 3: Valid in microprocessor and memory expansion modes.

## Processor mode register 1 (Note 1)




| Symbol | Address | When reset |
|---|---|---|
| PM1 | 0005₁₆ | 00000XX0₂ |

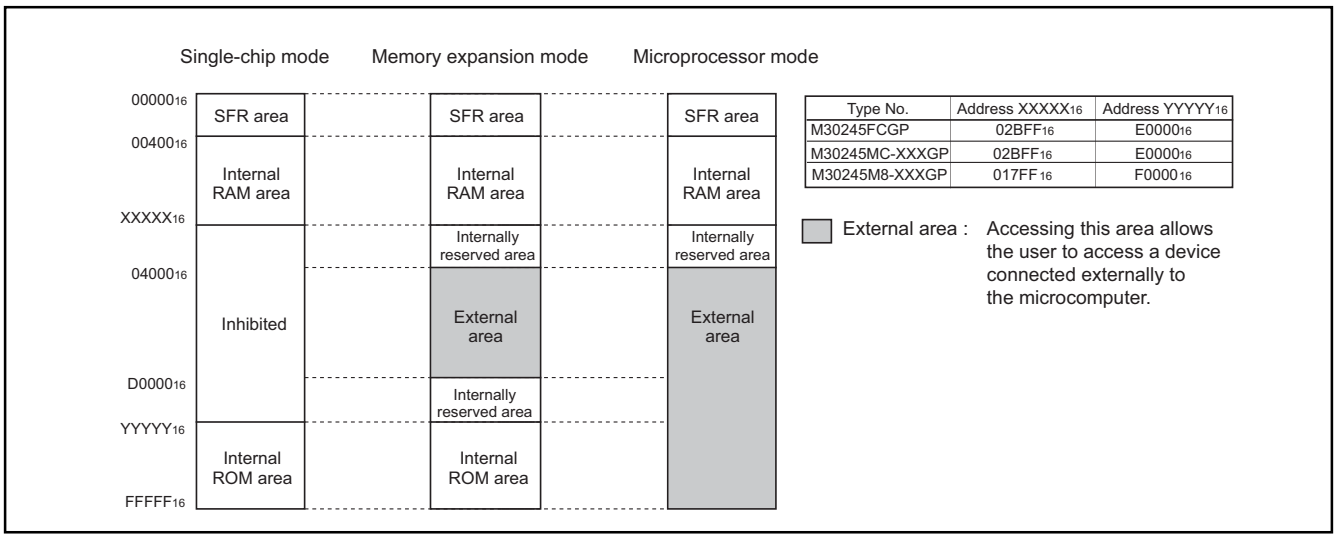
| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Reserved bit | | Must always be set to "0" | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is indeterminate if read. | | | – | – |
| Reserved bit | | Must always be set to "0" | O | O |
| PM16 | $\overline{WR}$ length control bit | 0 : Normal<br>1 : Extended | O | O |
| Reserved bit | | Must always be set to "0" | O | O |

Note 1: Set bit 1 of the protect register (address 000A₁₆) to "1" before writing new values to this register.

**Figure 1.10. Processor mode register 0 and 1**

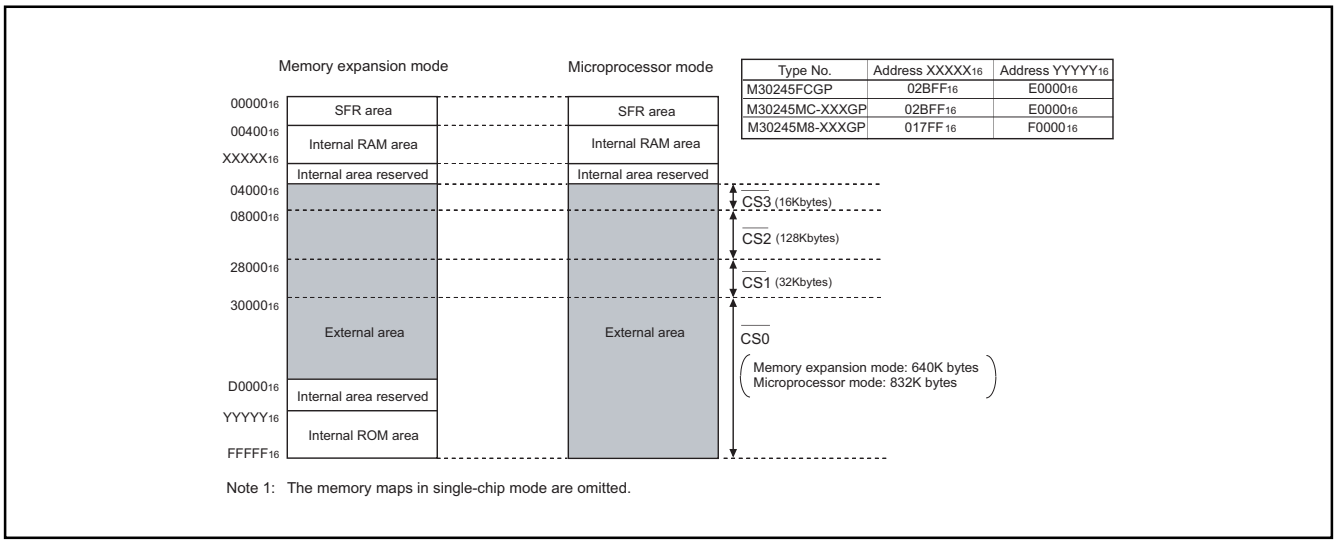


Downloaded from Arrow.com.




| Type No. | Address XXXXX16 | Address YYYYY16 |
|---|---|---|
| M30245FCGP | 02BFF16 | E000016 |
| M30245MC-XXXGP | 02BFF16 | E000016 |
| M30245M8-XXXGP | 017FF16 | F000016 |

**Figure 1.11. Memory map of each processor mode**




| Type No. | Address XXXXX16 | Address YYYYY16 |
|---|---|---|
| M30245FCGP | 02BFF16 | E000016 |
| M30245MC-XXXGP | 02BFF16 | E000016 |
| M30245M8-XXXGP | 017FF16 | F000016 |

Note 1: The memory maps in single-chip mode are omitted.

**Figure 1.12. Memory maps and chip-select areas**

## Bus Settings

The BYTE pin and bit 6 of the processor mode register 0 (address 00041₆) are used to change the bus settings. Table 1.14 shows the factors used to change the bus settings.

**Table 1.14. Switching bus settings**

| Bus setting | Switching factor |
|---|---|
| Switching external address bus width | b6 of processor mode register 0 |
| Switching external data bus width | BYTE pin |



Downloaded from Arrow.com.

**Selecting external address bus width**

The address bus width for external output in the 1M bytes of address space can be set to 16 bits (64K bytes address space) or 20 bits (1M bytes address space). When bit 6 of the processor mode register 0 is set to "1", the external address bus width is set to 16 bits, and P2 and P3 become part of the address bus. $P4_0$ to $P4_3$ can be used as programmable I/O ports. When bit 6 of processor mode register 0 is set to "0", the external address bus width is set to 20 bits, and P2, P3, and $P4_0$ to $P4_3$ become part of the address bus.

**Selecting external data bus width**

The external data bus width can be set to 8 or 16 bits. When the BYTE pin is "L", the bus width is set to 16 bits; when "H", it is set to 8 bits. (The internal bus width is permanently set to 16 bits.) While operating, fix the BYTE pin either to "H" or to "L". When the BYTE pin is "H", the data bus is set to 8 bits and P0 functions as the data bus and P1 as a programmable I/O port. When the BYTE pin is "L", the data bus is set to 16 bits and P0 and P1 are both used for the data bus. A software wait can also be added.

**Table 1.15. Pin functions for each processor mode**

| Processor mode | Single-chip mode | Memory expansion mode/microprocessor mode | |
|---|---|---|---|
| Data bus width<br>BYTE pin level | ———————— | 8 bits<br>BYTE = "H" | 16 bits<br>BYTE = "L" |
| $P0_0$ to $P0_7$ | I/O port | Data bus | Data bus |
| $P1_0$ to $P1_7$ | I/O port | I/O port | Data bus |
| $P2_0$ | I/O port | Address bus | Address bus |
| $P2_1$ to $P2_7$ | I/O port | Address bus | Address bus |
| $P3_0$ | I/O port | Address bus | Address bus |
| $P3_1$ to $P3_7$ | I/O port | Address bus | Address bus |
| $P4_0$ to $P4_3$<br>(function select bit =1) | I/O port | I/O port | I/O port |
| $P4_0$ to $P4_3$<br>(function select bit =0) | I/O port | Address bus | Address bus |
| $P4_4$ to $P4_7$ | I/O port | $\overline{CS}$ (chip select) or programmable I/O port<br>(Refer to "Bus control" for details) | |
| $P5_0$ to $P5_3$ | I/O port | Outputs $\overline{RD}$, $\overline{WRL}$, $\overline{WRH}$, and BCLK or $\overline{RD}$, $\overline{BHE}$, $\overline{WR}$ and BCLK (Refer to "Bus control" for details) | |
| $P5_4$ | I/O port | $\overline{HLDA}$ | $\overline{HLDA}$ |
| $P5_5$ | I/O port | $\overline{HOLD}$ | $\overline{HOLD}$ |
| $P5_6$ | I/O port | ALE | ALE |
| $P5_7$ | I/O port | $\overline{RDY}$ | $\overline{RDY}$ |



Downloaded from Arrow.com.

## Bus Control

The following explains the signals required for accessing external devices and software waits. The signals required for accessing the external devices are valid when the processor mode is set to memory expansion mode and microprocessor mode. The software waits are valid in all processor modes.

### Address bus/data bus

The address bus consists of the 20 pins $A_0$ to $A_{19}$ for accessing the 1M bytes of address space.

The data bus consists of the pins for data I/O. When the BYTE pin is "H", the 8 ports ($D_0$ to $D_7$) function as the data bus. When BYTE is "L", the 16 ports ($D_0$ to $D_{15}$) function as the data bus.

When a change is made from single-chip mode to memory expansion mode, the value of the address bus is undefined until external memory is accessed.

### Chip select signal

The chip select signal is output using the same pins as $P4_4$ to $P4_7$. Bits 0 to 3 of the chip select control register (address $0008_{16}$) set each pin to function as an I/O port or to output the chip select signal. The chip select control register is valid in memory expansion mode and microprocessor mode. In single-chip mode, $P4_4$ to $P4_7$ function as programmable I/O ports regardless of the value in the chip select control register.

In microprocessor mode, only CS0 outputs the chip select signal after reset. CS1 to CS3 function as input ports. Figure 1.13 shows the chip select control register.

The chip select signal can be used to split the external area into as many as four blocks. Table 1.16 shows the external memory areas specified using the chip select signal.

Chip select control register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: CSR | Address: $0008_{16}$ | When reset: $01_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| CS0 | $\overline{CS0}$ output enable bit | 0 : Chip select output disabled (Normal port pin)<br>1 : Chip select output enabled | O | O |
| CS1 | $\overline{CS1}$ output enable bit | | O | O |
| CS2 | $\overline{CS2}$ output enable bit | | O | O |
| CS3 | $\overline{CS3}$ output enable bit | | O | O |
| CS0W | $\overline{CS0}$wait bit | 0 : Wait state inserted<br>1 : No wait state | O | O |
| CS1W | $\overline{CS1}$wait bit | | O | O |
| CS2W | $\overline{CS2}$wait bit | | O | O |
| CS3W | $\overline{CS3}$wait bit | | O | O |

Chip select expansion register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: CSE | Address: $001B_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| CSE0W | $\overline{CS0}$wait expansion bit | 0 0 : 1 Wait state<br>0 1 : 2 Wait states<br>1 0 : 3 Wait states<br>1 1 : Inhibited | O | O |
| CSE1W | $\overline{CS1}$wait expansion bit | | O | O |
| CSE2W | $\overline{CS2}$wait expansion bit | | O | O |
| CSE3W | $\overline{CS3}$wait expansion bit | | O | O |

Note :Set CSEiW bits (i = 0 to 3) after setting the corresponding CSiW bit (i = 0 to 3) of the CSR register to "0". When CSiW bits are set to "1", CSEiW bits must be returned to "$00_2$".

**Figure 1.13. Chip-select control register**



Downloaded from Arrow.com.

**Table 1.16. External areas specified by the chip select signals**

| Processor mode | Chip-select signal | | | |
|---|---|---|---|---|
| | $\overline{CS0}$ | $\overline{CS1}$ | $\overline{CS2}$ | $\overline{CS3}$ |
| Memory expansion mode | $30000_{16}$ to $CFFFF_{16}$ (640 Kbytes) | $28000_{16}$ to $2FFFF_{16}$ (32 Kbytes) | $08000_{16}$ to $27FFF_{16}$ (128 Kbytes) | $04000_{16}$ to $07FFF_{16}$ (16 Kbytes) |
| Microprocessor mode | $30000_{16}$ to $FFFFF_{16}$ (832 Kbytes) | | | |

**Read/write signals**

With a 16-bit data bus (BYTE pin ="L"), bit 2 of the processor mode register 0 (address $0004_{16}$) selects the combinations of $\overline{RD}$, $\overline{BHE}$, and $\overline{WR}$ signals or $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ signals. With an 8-bit data bus (BYTE pin = "H"), use the combination of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals. (Set bit 2 of the processor mode register 0 (address $0004_{16}$) to "0".) Tables 1.17 and 1.18 show the operation of these signals.

After a reset, the combination of $\overline{RD}$, $\overline{WR}$R, and $\overline{BHE}$ signals is automatically selected.

When switching to the $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ combination, do not write to external memory until bit 2 of the processor mode register 0 (address $0004_{16}$) has been set . Before attempting to change the contents of the processor mode register 0, set bit 1 of the protect register (address $000A_{16}$) to "1".

**Table 1.17. Operation of $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ signals**

| Data bus width | $\overline{RD}$ | $\overline{WRL}$ | $\overline{WRH}$ | External data bus status |
|---|---|---|---|---|
| 16-bit (BYTE = "L") | L | H | H | Read data |
| | H | L | H | Write 1 byte of data to even address |
| | H | H | L | Write 1 byte of data to odd address |
| | H | L | L | Write data to both even and odd addresses |

**Table 1.18. Operation of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals**

| Data bus width | $\overline{RD}$ | $\overline{WRL}$ | $\overline{BHE}$ | A0 | External data bus status |
|---|---|---|---|---|---|
| 16-bit (BYTE = “L”) | H | L | L | H | Write 1 byte of data to odd address |
| | L | H | L | H | Read 1 byte of data from odd address |
| | H | L | H | L | Write 1 byte of data to even address |
| | L | H | H | L | Read 1 byte of data from even address |
| | H | L | L | L | Write data from both even and odd addresses |
| | L | H | L | L | Read data from both even and odd addresses |
| 8-bit (BYTE = “H”) | H | L | Not used | H/L | Write 1 byte of data |
| | L | H | Not used | H/L | Read 1 byte of data |



Downloaded from Arrow.com.

### The ALE signal

The ALE signal can be used by an external device to latch the address from the address bus. This signal indicates when the address on the bus is valid. Latch the address when the ALE signal falls.

### The $\overline{RDY}$ signal

RDY is a signal that facilitates access to an external device that requires long access time. As shown in Figure 1.14, if an "L" is input to the $\overline{RDY}$ at the BCLK falling edge, the bus turns to the wait state. If an "H" is being input to the $\overline{RDY}$ pin at the BCLK falling edge, the bus cancels the wait state. Table 1.19 shows the state of the microcomputer with the bus in the wait state. Figure 1.15 is an example of the $\overline{RD}$ signal prolonged by the $\overline{RDY}$ signal.

The $\overline{RDY}$ signal is valid when accessing the external area during the bus cycle in which bits 4 to 7 of the chip select control register (address $0008_{16}$) are set to "0". The $\overline{RDY}$ signal is invalid when setting "1" to all bits 4 to 7 of the chip select control register (address $0008_{16}$), but the $\overline{RDY}$ pin should still be connected properly as it is when not used.

**Table 1.19. Microcomputer status in ready state (Note)**

| Item | Status |
|---|---|
| Oscillation | On |
| R/$\overline{W}$ signal, address bus, data bus, $\overline{CS}$<br>ALE signal, $\overline{HLDA}$, programmable I/O ports | Maintain status when $\overline{RDY}$ signal received |
| Internal peripheral circuits | On |

Note: The $\overline{RDY}$ signal cannot be received immediately before a software wait.

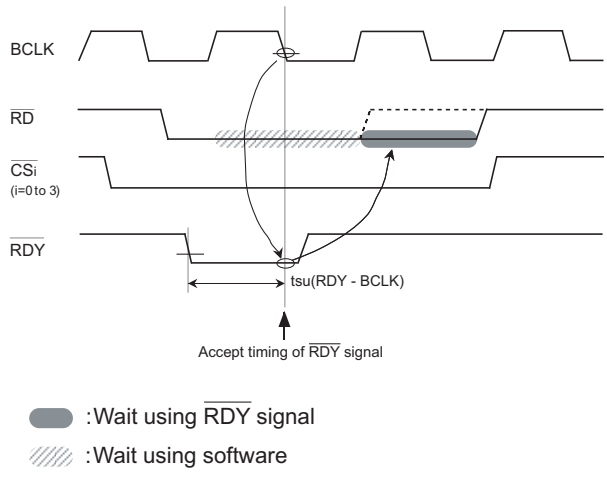



**Figure 1.14. Example of $\overline{RD}$ signal extended by $\overline{RDY}$ signal**



Downloaded from Arrow.com.

### HOLD signal

The hold signal is used to transfer the bus privileges from the CPU to the external circuits. Inputting "L" to the $\overline{\text{HOLD}}$ pin places the microcomputer in the hold state at the end of the current bus access. This status is maintained and "L" is output from the $\overline{\text{HLDA}}$ pin as long as "L" is input to the $\overline{\text{HOLD}}$ pin. Table 1.20 shows the microcomputer status in the hold state.

Bus priorities listed in descending order are: $\overline{\text{HOLD}}$, DMAC, and CPU.

**Table 1.20. Microcomputer status in HOLD state**

| Item | | Status |
|---|---|---|
| Oscillation | | On |
| R/$\overline{\text{W}}$ signal, address bus, data bus, $\overline{\text{CS}}$, $\overline{\text{BHE}}$ | | Floating |
| Programmable I/O ports | P0, P1, P2, P3, P4, P5 | Floating |
| | P6, P7, P8, P9, P10 | Maintains status when hold signal is received |
| $\overline{\text{HLDA}}$ | | Output "L" |
| Internal peripheral circuits | | On (Watchdog timer is stopped) |
| ALE signal | | Undefined |

### External bus status when the internal area is accessed

Table 1.21 shows the external bus status when the internal area is accessed.

**Table 1.21. External bus status when the internal area is accessed**

| Item | | SFR accessed | Internal ROM/RAM accessed |
|---|---|---|---|
| Address bus | | Address output | Maintain status before accessing address of external area |
| Data | When read | Floating | Floating |
| | When write | Output data | Undefined |
| $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ | | $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ output | Output "H" |
| $\overline{\text{BHE}}$ | | $\overline{\text{BHE}}$ output | Maintain status before accessing status of external area |
| $\overline{\text{CS}}$ | | Output "H" | Output "H" |
| ALE | | Output "L" | Output "L" |

### BCLK output

The user can choose to output BCLK on $P5_3$ by use of bit 7 of processor mode register 0 ($0004_{16}$) (Note). When set to "1", the output is left floating.

Note: Before attempting to change the contents of the processor mode register 0, set bit 1 of the protect register (address $000A_{16}$) to "1".



Downloaded from Arrow.com.

### Software wait

A software wait of one to three BCLK cycles can be inserted by setting bits 4 to 7 of the chip select control register (address 0008$_{16}$) and the bits in the chip select expansion register (address 001B$_{16}$).

Software waits can be set independently for each of the 4 chip select memory areas. Bits 4 to 7 of the chip select control register correspond to chip selects $\overline{CS0}$ to $\overline{CS3}$. When one of these bits is set to "1", the read bus cycle is executed in one BCLK cycle and the write bus cycle is executed in two BCLK cycles. When set to "0", the read and write bus cycles are executed in two, three or four BCLK cycles, depending on the settings in the chip select expansion register. The bits in the chip select expansion register are only valid when the corresponding bit in the chip select control register is set to "0". When the bits in the chip select control register are set to "1", the corresponding bits in the chip select expansion register must be set to "00$_2$". The bits in the chip select control register and chip select expansion register default to "0" after the microcomputer has been reset.

When the user is using the $\overline{RDY}$ signal, the relevant bit in the chip select control register's bits 4 to 7 must be set to "0".

The SFR area is always accessed in two BCLK cycles regardless of the setting of these control bits.

Table 1.22 shows the software waits and bus cycles. Figures 1.15 and 1.16 show example bus timing when using software waits.

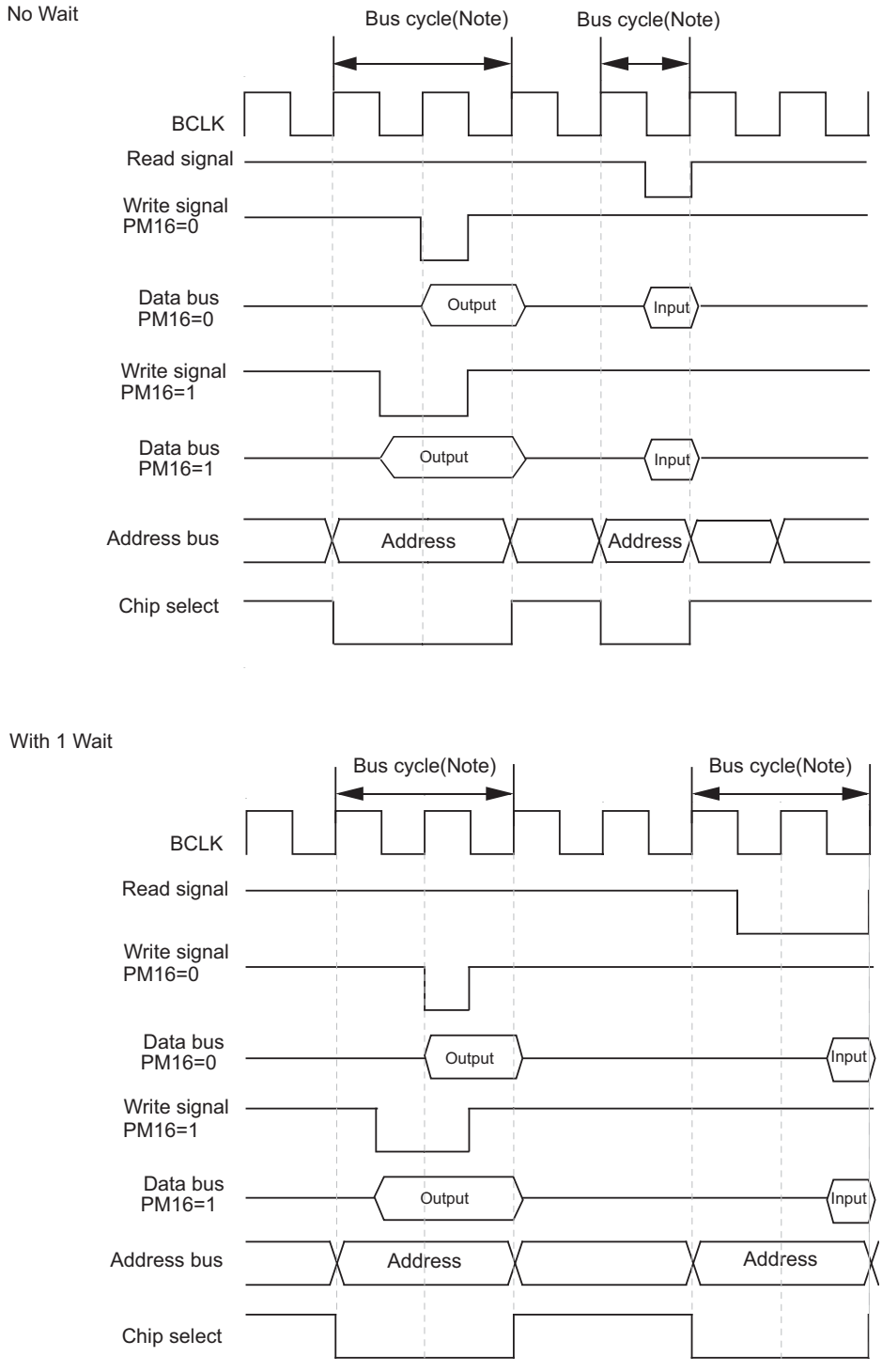
**Table 1.22. Software waits and bus cycles**

| Area | CSxW (Note 1) | CSExW (Note 2) | Bus Cycles | |
|---|---|---|---|---|
| | | | Read | Write |
| SFR | Invalid | Invalid | 2 BCLK cycles | 2 BCLK cycles |
| Internal ROM/RAM | Invalid | Invalid | 1 BCLK cycle | 1 BCLK cycle |
| External memory area | 0 | 00 | 2 BCLK cycles | 2 BCLK cycles |
| | 0 | 01 | 3 BCLK cycles | 3 BCLK cycles |
| | 0 | 10 | 4 BCLK cycles | 4 BCLK cycles |
| | 0 | 11 | Inhibited | Inhibited |
| | 1 | 00 | 1 BCLK cycle | 2 BCLK cycles |

Note 1: When using the $\overline{RDY}$ signal, always set this bit to "0".
Note 2: Set the CSxW bit to 0 before setting these bits. Also, when setting the CSxW bit to 1, be sure to reset these bits to '00$_2$' first.



Downloaded from Arrow.com.

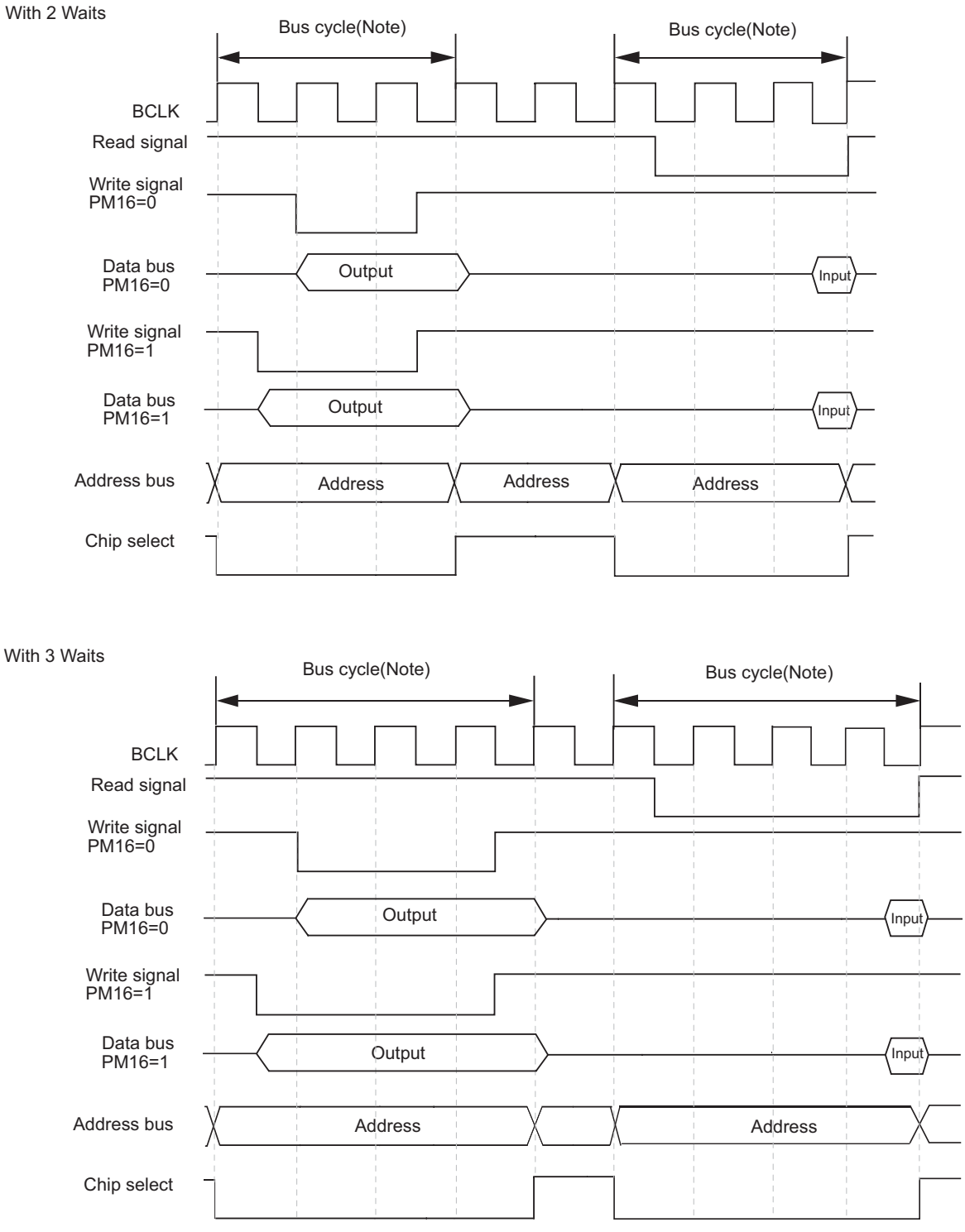



Note : These example timing charts indicate bus cycle length.
After this bus cycle sometimes come read and write cycles in succession.

**Figure 1.15. Typical bus timings using software wait (1)**



Downloaded from Arrow.com.




Note : These example timing charts indicate bus cycle length.
After this bus cycle sometimes come read and write cycles in succession.

**Figure 1.16. Typical bus timings using software wait (2)**



Downloaded from Arrow.com.

### Protection

The protection function is provided so that the values in important registers cannot be changed in the event that the program runs out of control. Figure 1.17 shows the protect register. The values in the processor mode register 0 (address $0004_{16}$), processor mode register 1 (address $0005_{16}$), system clock control register 0 (address $0006_{16}$) and system clock control register 1 (address $0007_{16}$) can only be changed when the respective bit in the protect register is set to "1". Setting the respective bits in the protect register to "0" will write protect these registers and not allow them to be changed.

Protect register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | 0 | |

Symbol: PRCR
Address: $000A_{16}$
When reset: $XXXXX000_2$

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PRC0 | Enables writing to system clock control registers 0 and 1 (addresses $0006_{16}$ and $0007_{16}$) and frequency synthesizer registers (addresses $03DB_{16}$ to $03DF_{16}$) | 0 : Write-inhibited<br>1 : Write-enabled | O | O |
| PRC1 | Enables writing to processor mode registers 0 and 1 (addresses $0004_{16}$ and $0005_{16}$) | 0 : Write-inhibited<br>1 : Write-enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits.<br>The values are indeterminate when read. | | | – | – |

**Figure 1.17. Protect register**



Downloaded from Arrow.com.

## System Clock

### Clock-generating Circuit

The clock generating circuit contains two oscillator circuits that supply the operating clock sources to the CPU and internal peripheral units. Figure 1.18 shows the block diagram of the clock-generating circuit. Table 1.23 lists the main clock-generating circuits.

**Table 1.23. Main clock-generating circuits**




Note: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
When the oscillation drive capacity is set to low, check that oscillation is stable.

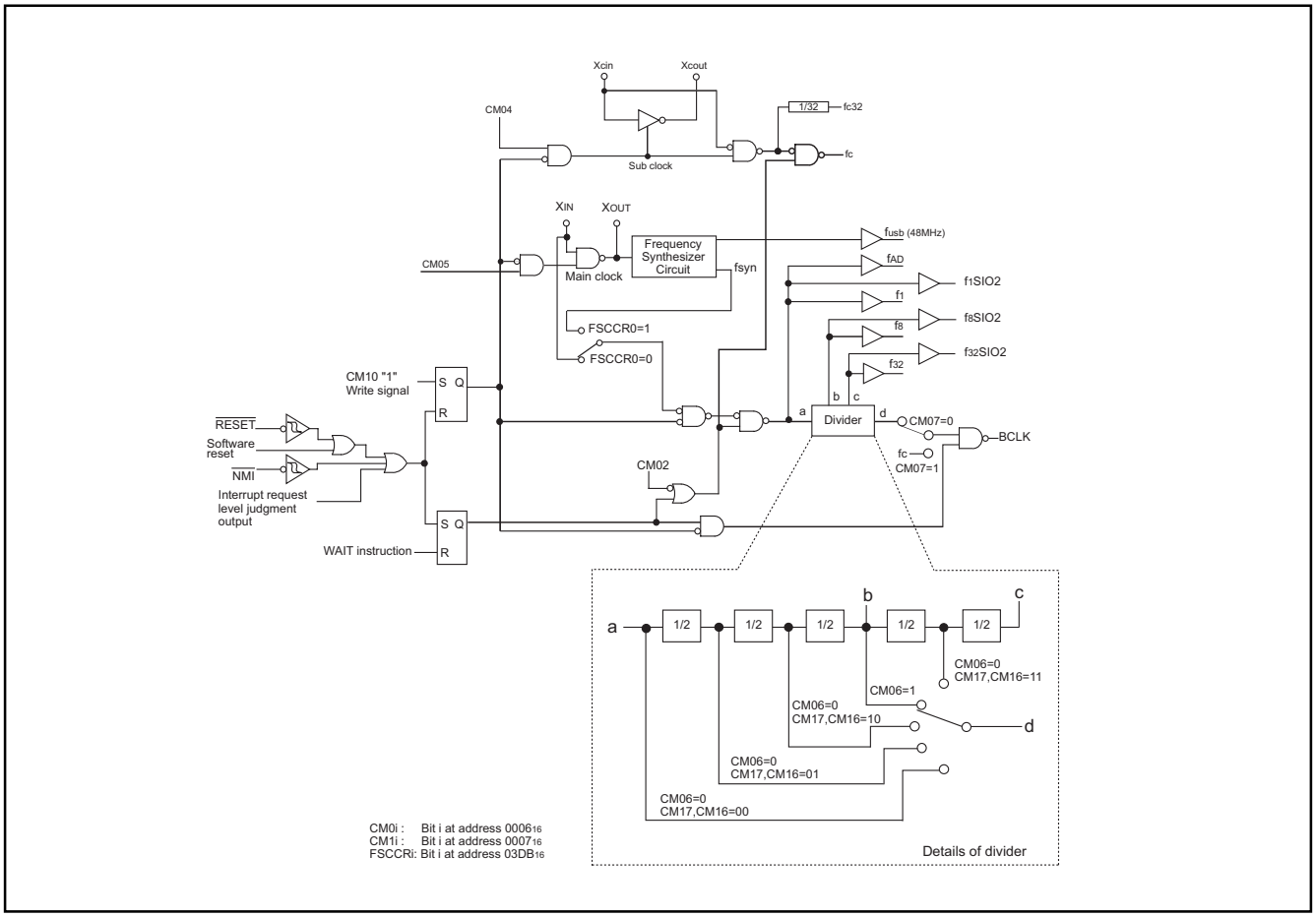



**Figure 1.18. Block diagram of the clock-generating circuit**



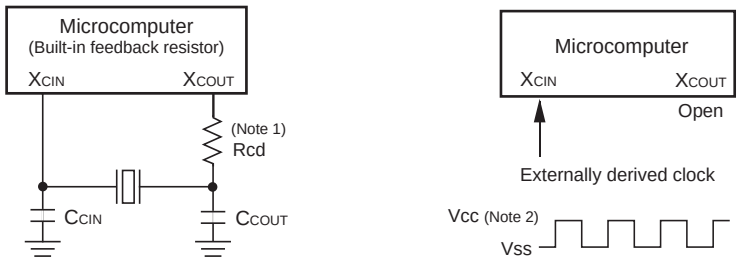
Downloaded from Arrow.com.

Figure 1.19 shows some examples of the main clock circuit, one using an oscillator connected to the circuit, and the other one using an externally derived clock for input. Figure 1.20 shows some examples of subclock circuits, one using an oscillator connected to the circuit, and the other one using an externally derived clock for input. Circuit constants in Figure 1.19 and Figure 1.20 vary with each oscillator used. Use the values recommended by the manufacturer of your oscillator.




Note: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
When the oscillation drive capacity is set to low, check that oscillation is stable.

**Figure 1.19. Examples of main clock**




Note 1: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
When the oscillation drive capacity is set to low, check that oscillation is stable. Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between XCIN and XCOUT following their instructions.

Note 2: Reference XCIN to Vcc supply.

**Figure 1.20. Examples of sub-clock**



Downloaded from Arrow.com.

## Clock Control

### Main clock

The main clock is generated by the main clock oscillation circuit. After a reset, this clock is divided by 8 to produce the BCLK. The clock can be stopped using the main clock stop bit (bit 5 at address $0006_{16}$). Stopping the clock, after switching the operating clock source of CPU to the subclock, reduces the power dissipation.

After the oscillation of the main clock oscillation circuit has stabilized, the drive capacity of the main clock oscillation circuit can be reduced using the XIN-XOUT drive capacity select bit (bit 5 at address $0007_{16}$). Reducing the drive capacity of the main clock oscillation circuit reduces power dissipation. This bit changes to "1" when shifting from high-speed/medium-speed mode to stop mode and at a reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

### Subclock

The subclock is generated by the subclock oscillation circuit. No subclock is generated after a reset. After oscillation is started using the port Xc select bit (bit 4 at address $0006_{16}$), the subclock can be selected as the BCLK by using the system clock select bit (bit 7 at address $0006_{16}$). However, be sure that the subclock oscillation has fully stabilized before switching.

After the oscillation of the subclock oscillation circuit has stabilized, the drive capacity of the subclock oscillation circuit can be reduced using the XCIN-XCOUT drive capacity select bit (bit 3 at address $0006_{16}$). Reducing the drive capacity of the subclock oscillation circuit reduces the power dissipation. This bit changes to "1" when changing to stop mode and at a reset.

### BCLK

The BCLK is the clock that drives the CPU, and is equal to fc or the clock that is derived by dividing the main clock by 1, 2, 4, 8, or 16. The BCLK is derived by dividing the main clock by 8 after a reset. The BCLK signal can be output from the BCLK pin (P5$_3$) by use of the BCLK output disable bit (bit 7 at address $0004_{16}$) in the memory expansion and the microprocessor modes.

The main clock division select bit 0 (bit 6 at address $0006_{16}$) changes to "1" when shifting from high-speed/medium-speed to stop mode and at reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

### Peripheral function clock ($f_1$, $f_8$, $f_{32}$, $f_{1SIO2}$, $f_{8SIO2}$, $f_{32SIO2}$, $f_{AD}$)

The clock for the peripheral devices is derived from the main clock or by dividing it by 1, 8, or 32. The peripheral function clock is stopped by stopping the main clock or by setting the WAIT peripheral function clock stop bit (bit 2 at $0006_{16}$) to "1" and then executing a WAIT instruction.

#### $f_{C32}$

This clock is derived by dividing the subclock by 32. It is used for the Timer A counts.

#### fc

This clock has the same frequency as the subclock. It is used for the BCLK and for the watchdog timer.

#### $f_{USB}$

This clock provides a 48 MHz signal required for USB operation. It is derived from the Frequency Synthesizer circuit.



Downloaded from Arrow.com.

### System clock control registers

Figure 1.21 shows the system clock control registers 0 and 1.

System clock control register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | 0 | 0 |

Symbol: CM0 — Address: $0006_{16}$ — When reset: $48_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| Reserved bit | | Always set to "0" | O | O |
| CM02 | WAIT peripheral function clock stop bit | 0 : Do not stop in wait mode<br>1 : Stop in wait mode (Note 8) | O | O |
| CM03 | XCIN-XCOUT drive capacity select bit (Note 2) | 0 : LOW<br>1 : HIGH | O | O |
| CM04 | Port Xc select bit | 0 : I/O port<br>1 : XCIN-XCOUT generation | O | O |
| CM05 | Main clock (XIN-XOUT) stop bit (Note 3, 4, 5) | 0 : On<br>1 : Off | O | O |
| CM06 | Main clock division select bit 0 (Note 6) | 0 : CM16 and CM17 valid<br>1 : Divide-by-8 mode | O | O |
| CM07 | System clock select bit (Note 7) | 0 : XIN, XOUT<br>1 : XCIN, XCOUT | O | O |

Note 1: Set bit 0 of the protect register (address $000A_{16}$) to "1" before writing to this register.
Note 2: Changes to "1" when changing to Stop mode and Reset.
Note 3: When entering power saving mode, main clock is stopped using this bit. When returning from stop mode and operating in XIN, set this bit to "0". When main clock oscillation is operating by itself, set system clock select bit (CM07) to "1" before setting this bit to "1".
Note 4: When inputting external clock, only clock oscillation buffer is stopped and clock input is acceptable.
Note 5: If this bit is set to "1", XOUT becomes "H". The built-in feedback resistor remains connected, so XIN becomes pulled up to XOUT ("H") using the feedback resistor.
Note 6: This bit changes to "1" when changing from high-speed/medium mode to stop mode and at reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.
Note 7: Set Port Xc select bit (CM04) to "1" and stabilize the sub clock oscillating before setting to this bit from "0" to "1". Do not write to both bits at the same time. Also, set the main clock stop bit (CM05) to "0" and stabilize the main clock oscillating before setting this bit from "1" to "0".
Note 8: fc32 is not included. Do not set to "1" when using low-speed or low power dissipation mode.

System clock control register 1 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | 0 | 0 | |

Symbol: CM1 — Address: $0007_{16}$ — When reset: $20_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| CM10 | All clock stop control bit (Note 4) | 0 : Clock on<br>1 : All clocks off (stop mode) | O | O |
| Reserved bit | | Always set to "0" | O | O |
| CM15 | XIN-XOUT drive capacity select bit (Note 2) | 0 : LOW<br>1 : HIGH | O | O |
| CM16 | Main clock division select bit 1 (Note 3) | b7 b6<br>0 0 : No division mode<br>0 1 : Divide-by-2 mode<br>1 0 : Divide-by-4 mode<br>1 1 : Divide-by-16 mode | O | O |
| CM17 | | | O | O |

Note 1: Set bit "0" of the protect register (address $000A_{16}$) to "1" before writing to this register.
Note 2: This bit changes to "1" when shifting from high-speed/medium speed mode to stop mode an at reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.
Note 3: Can be selected when bit 6 of the system clock control register 0 (address $0006_{16}$) is "0". If "1", divide mode is fixed at 8.
Note 4: If this bit is set to "1", XOUT becomes "H" and the built-in feedback resistoris cut off. XCIN and XCOUT become high impedance state.

**Figure 1.21. Clock control registers 0 and 1**



Downloaded from Arrow.com.

### Stop Mode

Writing "1" to the all-clock stop control bit (bit 0 at address $0007_{16}$) stops all oscillation and the microcomputer enters stop mode. In stop mode, the content of the internal RAM is retained provided that Vcc remains above 2V.

Because the oscillation of BCLK, $f_1$ to $f_{32}$, $f_{1SIO2}$ to $f_{32SIO2}$, fc, $fc_{32}$ and $f_{AD}$ stops in stop mode, peripheral functions such as the A/D converter and watchdog timer do not function. However, timer A operates, provided that the event counter mode is set to an external pulse, and UARTi (i = 0 to 3) functions provided an external clock is selected. Table 1.24 shows the status of the ports in stop mode.

Stop mode is cancelled by a hardware reset or interrupt. If an interrupt is to be used to cancel stop mode, that interrupt must first be enabled and the interrupt priority of any interrupts not used to cancel stop mode should be set to "0". The I flag must also be set prior to stopping for an interrupt to cancel it. When returning by an interrupt, that interrupt routine is executed. If only a hardware reset or an $\overline{NMI}$ interrupt is used to cancel stop mode, change the priority level of all interrupts to "0", then change to stop mode.

After coming out of stop mode, it is recommended that four "NOP" instructions be executed to clear the instruction queue.

When changing from high-speed/medium speed mode to stop mode and at reset, the main clock division select bit 0 (bit 6 at $0006_{16}$) is set to "1". When changing from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

**Table 1.24. Port status during stop mode**

| Pin | Memory expansion mode<br>Microprocessor mode | Single-chip mode |
|---|---|---|
| Address bus, data bus, $\overline{CS0}$ to $\overline{CS3}$, $\overline{BHE}$ | Retains status before stop mode | |
| $\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$ | "H" | |
| $\overline{HLDA}$, BCLK | "H" | |
| ALE | "H" | |
| Port | Retains status before stop mode | Retains status before stop mode |



Downloaded from Arrow.com.

### Wait Mode

When a WAIT instruction is executed, the BCLK stops and the microcomputer enters the wait mode. In this mode, oscillation continues but the BCLK and watchdog timer stop. Writing "1" to the WAIT peripheral function clock stop bit and executing a WAIT instruction stops the clock being supplied to the internal peripheral functions, allowing power dissipation to be reduced. However, the peripherial function clock $f_{C32}$ does not stop during wait mode and thus does not contribute to any power savings. When the MCU is running in low-speed or low power dissipiation mode, do not enter WAIT mode with this bit set to "1". Table 1.25 shows the status of the ports in wait mode.

Wait mode is cancelled by a hardware reset or interrupt. If an interrupt is used to cancel wait mode, that interrupt must first be enabled and the interrupt priority levels of all other interrupts that are not used to cancel wait mode must be set to "0". When returning from an interrupt, the microcomputer restarts using as BCLK the clock that had been selected when the WAIT instruction was executed, and the program continues from the interrupt routine. If only a hardware reset or $\overline{\text{NMI}}$ interrupt is used to cancel wait mode, change the priority level of all interrupts to "0", then shift to wait mode.

**Table 1.25. Port status during wait mode**

| Pin | Memory expansion mode Microprocessor mode | Single-chip mode |
|---|---|---|
| Address bus, data bus, $\overline{\text{CS0}}$ to $\overline{\text{CS3}}$ | Retains status before wait mode | |
| $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{BHE}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ | "H" | |
| $\overline{\text{HLDA}}$, BCLK | "H" | |
| ALE | "H" | |
| Port | Retains status before wait mode | Retains status before wait mode |

### BCLK Status transition

Power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 1.26 shows the operating modes corresponding to the settings of system clock control registers 0 and 1.

When reset, the device starts in division by 8 mode. The main clock division select bit 0 (bit 6 at address $0006_{16}$) changes to "1" when shifting from high-speed/medium-speed to stop mode and at a reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained. The following shows the operational modes of BCLK.



Downloaded from Arrow.com.

**Table 1.26. System clock control registers 0 and 1 operating mode settings**

| CM17 | CM16 | CM07 | CM06 | CM05 | CM04 | BCLK operating mode |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | Invalid | Divide-by-2 mode |
| 1 | 0 | 0 | 0 | 0 | Invalid | Divide-by-4 mode |
| Invalid | Invalid | 0 | 1 | 0 | Invalid | Divide-by-8 mode |
| 1 | 1 | 0 | 0 | 0 | Invalid | Divide-by-16 mode |
| 0 | 0 | 0 | 0 | 0 | Invalid | No division mode |
| Invalid | Invalid | 1 | Invalid | 0 | 1 | Low-speed mode |
| Invalid | Invalid | 1 | Invalid | 1 | 1 | Low power dissipation mode |

- **Divide by 2 mode**

  The main clock is divided by 2 to obtain the BCLK.

- **Divide by 4 mode**

  The main clock is divided by 4 to obtain the BCLK.

- **Divide by 8 mode**

  The main clock is divided by 8 to obtain the BCLK. When reset, the device starts operating from this mode. Before the user can go from this mode to no division mode, division by 2 mode, or division by 4 mode, the main clock oscillator must be stable. When going to low-speed or lower power consumption mode, make sure the subclock's oscillator is stable.

- **Divide by 16 mode**

  The main clock is divided by 16 to obtain the BCLK.

- **No-division mode**

  The main clock is divided by 1 to obtain the BCLK.

- **Low-speed mode**

  fc is used as the BCLK.

  Note: Oscillation of both the main and sub-clocks must have stabilized before transferring from this mode to another or vice versa. At least 2 to 3 seconds are required after the subclock starts. Therefore, write the program to wait until this clock has stabilized after powering up and after returning from stop mode.

- **Low power dissipation mode**

  fc is the BCLK and the main clock is stopped.

  Note: Before the count source for BCLK can be changed from XIN to XCIN or vice versa, the new count source's oscillator must first be stable. Allow some wait time in software for the oscillation to stabilize before switching the clock over.



Downloaded from Arrow.com.

## Power control

The following is a description of the three available power control modes. Figure 1.22 shows the state transition diagram for these modes.

**Normal operation mode**

- **High-speed mode**

  Divide-by-1 frequency of the main clock becomes the BCLK. The CPU operates with the internal clock selected. Each peripheral function operates according to its assigned clock.

- **Medium-speed mode**

  Divide-by-2, divide-by-4, divide-by-8, or divide-by-16 frequency of the main clock becomes the BCLK. The CPU operates according to the internal clock selected. Each peripheral function operates according to its assigned clock.

- **Low-speed mode**

  fc becomes the BCLK. The CPU operates according to the fc clock. The fc clock is supplied by the secondary clock. Each peripheral function operates according to its assigned clock.

- **Low power consumption mode**

  The main clock operating in low-speed mode is stopped. The CPU operates according to the fc clock. The fc clock is supplied by the secondary clock. The only peripheral functions that operate are those with the subclock selected as the count source.

**Wait mode**

The CPU operation is stopped. The oscillators do not stop.

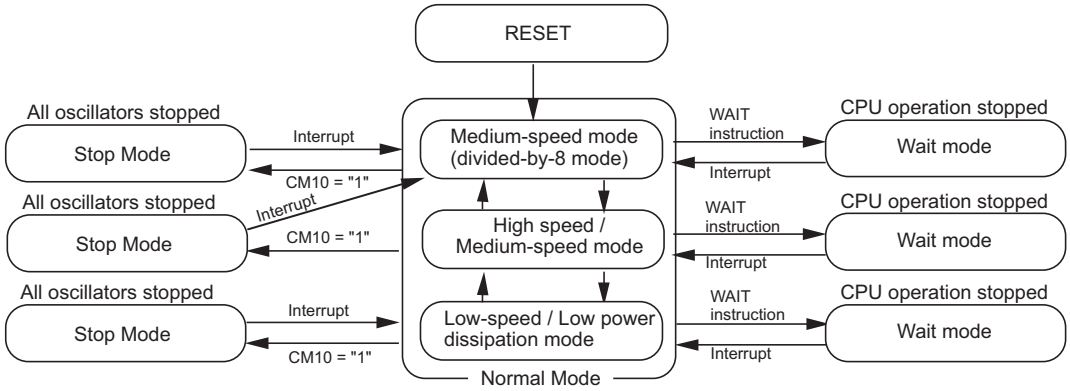
**Stop mode**

All oscillators stop. The CPU and all built-in peripheral functions stop. Of the three modes discusses, the stop mode is the most effective in decreasing power consumption.

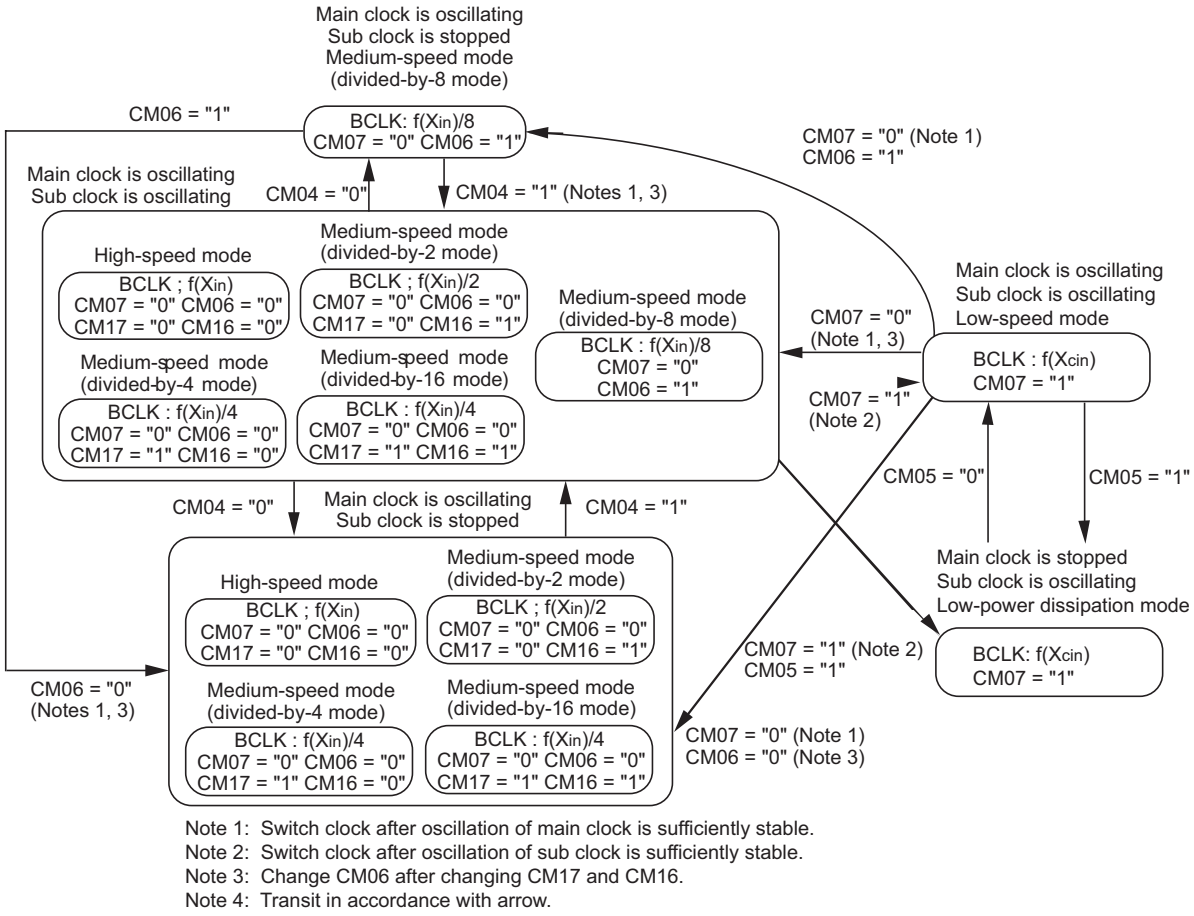


Downloaded from Arrow.com.

**State Transitions for Stop and Wait modes**




**State Transitions for normal mode**




Note 1: Switch clock after oscillation of main clock is sufficiently stable.
Note 2: Switch clock after oscillation of sub clock is sufficiently stable.
Note 3: Change CM06 after changing CM17 and CM16.
Note 4: Transit in accordance with arrow.

**Figure 1.22. Power control mode state transition diagram**



Downloaded from Arrow.com.

## Frequency synthesizer circuit

The frequency synthesizer circuit generates a 48MHz clock ($f_{USB}$) needed by the USB block and a clock $f_{SYN}$ that are a multiple of the external input reference clock f(XIN). A block diagram of the circuit is shown in Figure 1.23.

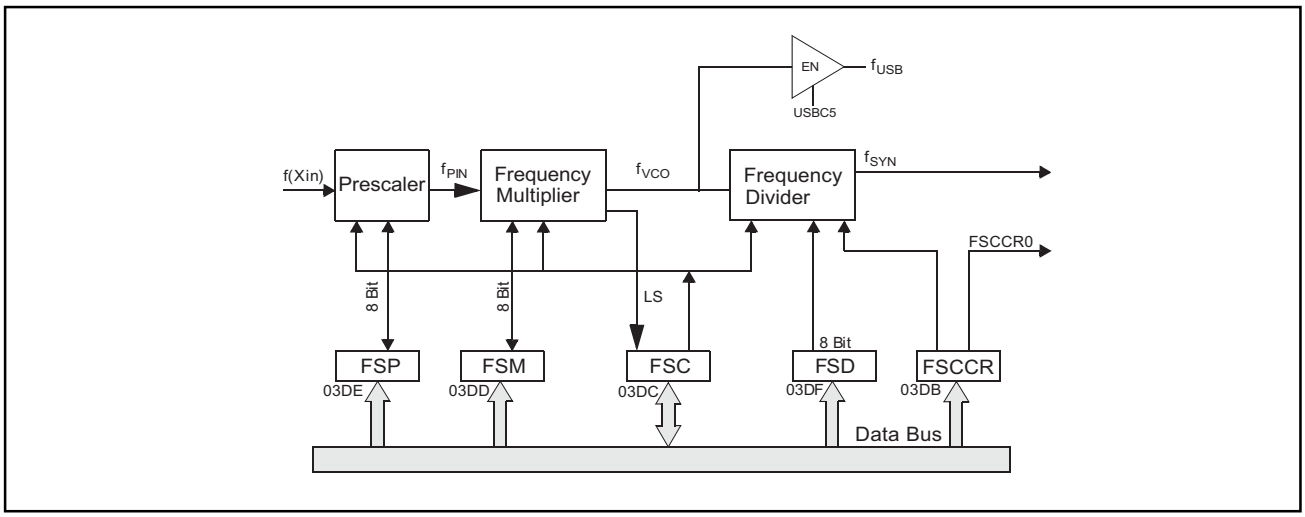



**Figure 1.23. Frequency Synthesizer Circuit**

The frequency synthesizer consists of a prescaler, frequency multiplier, a frequency divider, and five registers: FSP; FSM; FSC; FSD; and FSCCR. Clock f(XIN) is prescaled down using FSP to generate $f_{PIN}$. $f_{PIN}$ is multiplied by FSM to generate an $f_{VCO}$ clock, which is then divided by FSD to produce the clock $f_{SYN}$. The $f_{VCO}$ clock is optimized for 48 MHz operation and is buffered and sent out of the frequency synthesizer block as signal fUSB. This signal is used by the USB block.

The FSC0 bit in the FSC Control Register enables the frequency synthesizer block. When disabled (FSC0 = "0"), $f_{VCO}$ is held at either a high or low state. When the frequency synthesizer control bit is active (FSC0 = "1"), a lock status (LS ="1") indicates that $f_{SYN}$ and $f_{VCO}$ are the correct frequency. The LS and FSCO control bits in the FSC Control register are shown in Figure 1.24.

When using the frequency synthesizer, a low-pass filter must be connected to the LPF pin. Once the frequency synthesizer is enabled, a delay of 2-5ms is recommended before the output of the frequency synthesizer is used. This is done to allow the output to stabilize. It is also recommended that none of the registers be modified once the frequency synthesizer is enabled as it will cause the output to be temporarily (2-5ms) unstable.

The MCU clock source is selected via the Frequency Synthesizer Clock Control register (FSCCR). See Figure 1.25.

Note: None of the registers must be written to once the frequency synthesizer is enabled and used as the system clock source (FSCCR register, address $03DB_{16}$, bit '0' set to '1') because it will cause the output of the PLL to freeze. Switch system back to f(XIN) and disable before modifying PLL registers.



Downloaded from Arrow.com.

Frequency Synthesizer Control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | | | |

Symbol: FSC | Address: $03DC_{16}$ | When reset: $60_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| FSE | Frequency Synthesizer enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| VCO0 | VCO Gain Control | b2 b1<br>0 0 : Lowest gain<br>0 1 : Low gain<br>1 0 : High gain (Note )<br>1 1 : Highest gain | O | O |
| VC01 | | | O | O |
| Reserved bit | | Must always be set to "0" | O | O |
| CHG0 | LPF Current Control | b6 b5<br>0 0 : Disabled<br>0 1 : Low current<br>1 0 : Medium current (Note)<br>1 1 : High current | O | O |
| CHG1 | | | O | O |
| LS | Frequency Synthesizer Lock Status | 0 : Unlocked<br>1 : Locked | O | O |

Note: Recommended

**Figure 1.24. Frequency Synthesizer Control register (FSC)**

Frequency Synthesizer Clock Control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | 0 | 0 | 0 | |

Symbol: FSCCR | Address: $03DB_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| FSCCR0 | Clock source selection | 0 :Xin<br>1 :$f_{SYN}$ | O | O |
| Reserved | | Must always be set to "0" | O | O |
| FCCR4 | Divide-by-3 option | 0 : Normal<br>1 : Divide-by-3 | O | O |
| Reserved | | Must always be set to "0" | O | O |

**Figure 1.25. Frequency Synthesizer Clock Control register (FSCCR)**

## Prescaler

Clock fPIN is a divided down version of clock f(XIN) (see Figure 1.26). The relationship between fPIN and the clock f(XIN) input to the prescaler is as follows:

• fPIN = f(XIN) / 2(n+1) where n is the decimal equivalent loaded into the FSP.

• Setting FSP to 255 disables the prescaler and fPIN = f(XIN).

Note: fPIN frequency below 1 MHz is not recommended.

| MSB 7 | | | | | | | LSB 0 |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

Address: $03DE_{16}$
Access: R/W
Reset: $FF_{16}$

| $f_{PIN}$ | FSP | | f(XIN) |
|---|---|---|---|
| | Dec(n) | Hex(n) | |
| 12 MHz | 255 | FF | 12.00 MHz |
| 1 MHz | 7 | 07 | 16.00 MHz |
| 1 MHz | 5 | 05 | 12.00 MHz |
| 2 MHz | 3 | 03 | 16.00 MHz |
| 2 MHz | 2 | 02 | 12.00 MHz |
| 3 MHz | 1 | 01 | 12.00 MHz |
| 6 MHz | 0 | 00 | 12.00 MHz |

$f_{PIN} = f(X_{IN})/2(n+1)$

**Figure 1.26. Frequency Synthesizer Prescaler register (FSP)**



Downloaded from Arrow.com.

### Multiplier

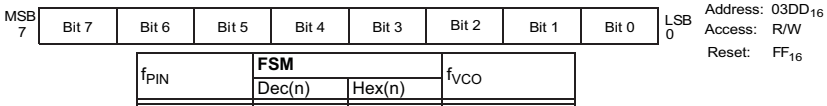
Clock $f_{VCO}$ is a multiplied up version of clock $f_{PIN}$ (See Figure 1.27). The relationship between $f_{VCO}$ and the clock ($f_{PIN}$) input to the multiplier from the prescaler is as follows:

• $f_{VCO} = f_{PIN} \times 2(n+1)$ where n is the decimal equivalent of the value loaded in FSM.

• Setting FSM to 255 disables the multiplier and $f_{VCO} = f_{PIN}$.

Note 1: n must be chosen such that $f_{VCO}$ equals 48 MHz.

Note 2: Minimum $f_{PIN}$ is 1 MHz. Maximum $f_{PIN}$ is 12 MHz.




| $f_{PIN}$ | FSM | | $f_{VCO}$ |
|---|---|---|---|
| | Dec(n) | Hex(n) | |
| 1 MHz | 23 | 17 | 48.00 MHz |
| 2 MHz | 11 | 0B | 48.00 MHz |
| 4 MHz | 5 | 05 | 48.00 MHz |
| 6 MHz | 3 | 03 | 48.00 MHz |
| 8 MHz | 2 | 02 | 48.00 MHz |
| 12 MHz | 1 | 01 | 48.00 MHz |

$f_{VCO} = f_{PIN} \times 2(n+1)$

**Figure 1.27. Frequency Synthesizer Multiply register (FSM)**

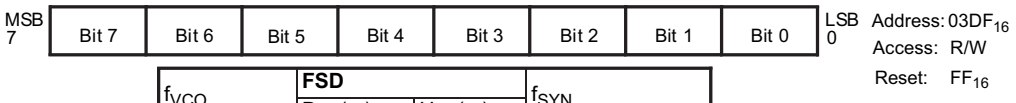
### Divider

Clock $f_{SYN}$ is a divided down version of clock $f_{VCO}$ (See Figure 1.28). The relationship between $f_{SYN}$ and the clock ($f_{VCO}$) input to the divider from the multiplier is as follows:

• $f_{SYN} = f_{VCO} / 2(m+1)$ where m is the decimal equivalent of the value loaded in FSD.

NOTE: $f_{SYN} = f_{VCO} / (m+1)$ when the divide by 3 option (bit 4 at address $03DB_{16}$) is set and when m = 2.

• Setting FSD to 255 disables the divider and $f_{SYN} = f_{VCO}$.




| $f_{VCO}$ | FSD | | $f_{SYN}$ |
|---|---|---|---|
| | Dec(m) | Hex(m) | |
| 48.00 MHz | 1 | 01 | 12.00 MHz |
| 48.00 MHz | 2 | 02 | 8.00 MHz |
| 48.00 MHz | 2 | 02 | 16.00 MHz (Note) |
| 48.00 MHz | 3 | 03 | 6.00 MHz |
| 48.00 MHz | 127 | 7F | 187.50 KHz |

$f_{SYN} = f_{VCO} / 2(m+1)$

Note: $f_{SYN} = f_{VCO} / (m+1)$ if FSCCR4 = 1 and m = 2

**Figure 1.28. Frequency Synthesizer Divide register (FSD)**



Downloaded from Arrow.com.

## Interrupts

Figure 1.29 lists the types of interrupts.

- **Maskable:** An interrupt that can be enabled or disabled by the interrupt enable flag (I flag) or can have its interrupt priority changed by the priority level.
- **Non-maskable:** An interrupt that cannot be enabled or disabled by the interrupt enable flag (I flag) or cannot have its interrupt priority changed by the priority level.

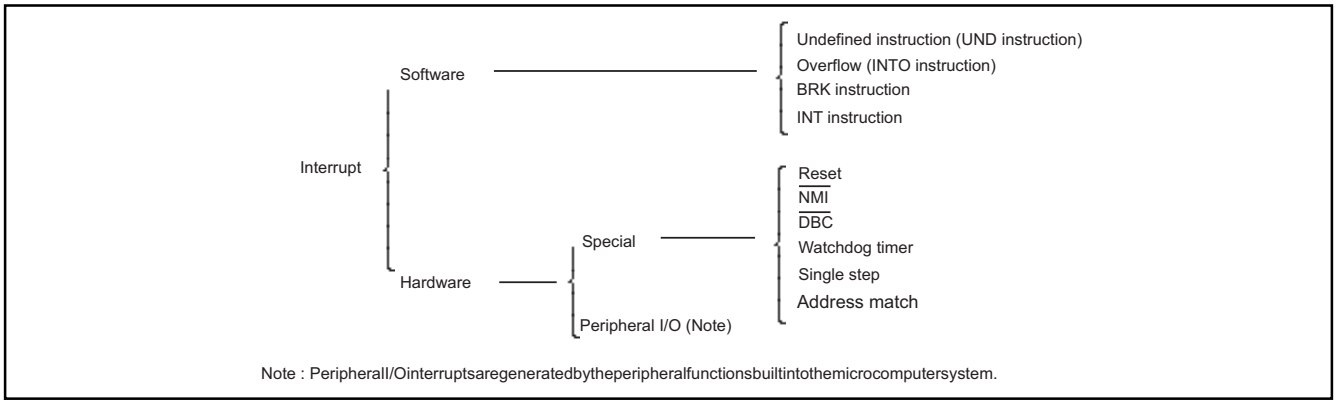



**Figure 1.29. Interrupt classification**

### Software Interrupts

A software interrupt occurs when executing certain instructions. Software interrupts are non-maskable interrupts.

**• Undefined instruction interrupt**

An undefined instruction interrupt occurs when executing the UND instruction.

**• Overflow interrupt**

An overflow interrupt occurs when an executing arithmetic instruction overflows. The instructions that set an O flag when an overflow occurs are: ABS, ADC, ADCF, ADD, CMP, DIV, DIVU, DIVX, NEG, RMPA, SBB, SHA, SUB

**• BRK interrupt**

A BRK interrupt occurs when executing the BRK instruction.

**• INT interrupt**

An INT interrupt occurs when specifying one of the software interrupt numbers 0 through 63 and executing the INT instruction. Software interrupt numbers 0 through 31 are assigned to peripheral I/O interrupts, so executing the INT instruction executes the same interrupt routine as the peripheral I/O interrupt.
The stack pointer (SP), used for the INT interrupt, is dependent on which software interrupt number is selected.
As far as software interrupt numbers 0 through 31 are concerned, the microcomputer saves the stack pointer assignment flag (U flag) when it accepts an interrupt request. The U flag is set to "0"" selecting the interrupt stack pointer then the interrupt sequence is executed. When returning from the interrupt routine, the U flag is returned to its previous state before accepting the interrupt request.
As far as software numbers 32 through 63 are concerned, the stack pointer does not change.



Downloaded from Arrow.com.

## Hardware Interrupts

Hardware interrupts are classified into two types - special interrupts and peripheral I/O interrupts.

**Special interrupts**

Special interrupts are non-maskable interrupts.

**• Reset**

Reset occurs if an "L" is input to the $\overline{\text{RESET}}$ pin.

**• $\overline{\text{NMI}}$ interrupt**

An $\overline{\text{NMI}}$ interrupt occurs if an "L" is input to the $\overline{\text{NMI}}$ pin.

**• $\overline{\text{DBC}}$ interrupt**

This interrupt is exclusively for the debugger, do not use it in other circumstances.

**• Watchdog timer interrupt**

Generated by the watchdog timer.

**• Single-step interrupt**

This interrupt is exclusively for the debugger, do not use it in other circumstances. With the debug flag (D flag) set to "1", a single-step interrupt occurs after one instruction is executed.

**• Address match interrupt**

An address match interrupt occurs immediately before the instruction held in the address indicated by the address match interrupt register is executed with the address match interrupt enable bit set to "1". If an address other than the first address of the instruction in the address match interrupt register is set,

**Peripheral I/O interrupts**

A peripheral I/O interrupt is generated by one of the built-in peripheral functions. Built-in peripheral functions are dependent on classes of products, so the interrupt factors are also dependent on classes of products. The interrupt vector table is the same as the one for software interrupt numbers 0 through 31 the INT instruction uses. Peripheral I/O interrupts are maskable interrupts.

**• Bus collision detection interrupt**

This is an interrupt that the serial I/O bus collision detection generates.

**• DMA0 through DMA3 interrupt**

These are interrupts the DMA generates.

**• Key-input interrupt**

A key-input interrupt occurs if an "L" is input to any of the $\overline{\text{KI}_1}$ to $\overline{\text{KI}_7}$ pins.

**• A/D conversion interrupt**

This is an interrupt that the A/D converter generates.

**• UART0, UART1, UART2, UART3 transmit / NACK / SSI0, SSI1 transmit interrupt**

These are interrupts that the serial I/O, $I^2C$, and SSI generate.

**• UART0, UART1, UART2, UART3 receive / ACK / SSI0, SSI1 receive interrupt**

These are interrupts that the serial I/O, $I^2C$, and SSI generate.

**• Timer A0 interrupt through Timer A4 interrupt**

These are interrupts that Timer A generates

**• INT0 through INT2 interrupt**

An INT interrupt occurs if either a rising edge or a falling edge or both edges are input to one of the INT pins.

**• USB interrupts (EP0, Suspend, Resume, SOF, Reset, USB Function)**

These are interrupts that are generated from USB.

**• VBus Detect interrupt**

This interrupt is generated from the USB VBus detection circuitry.



Downloaded from Arrow.com.

## Interrupt Routine

### Interrupt vector tables

1If an interrupt request is accepted, program execution branches to the interrupt routine set in the interrupt vector table. Set the first address of the interrupt routine in each vector table. Figure 1.30 shows the format for specifying the address. Two types of interrupt vector tables are available - fixed vector table in which addresses are fixed and variable vector table in which addresses can be varied by the setting.

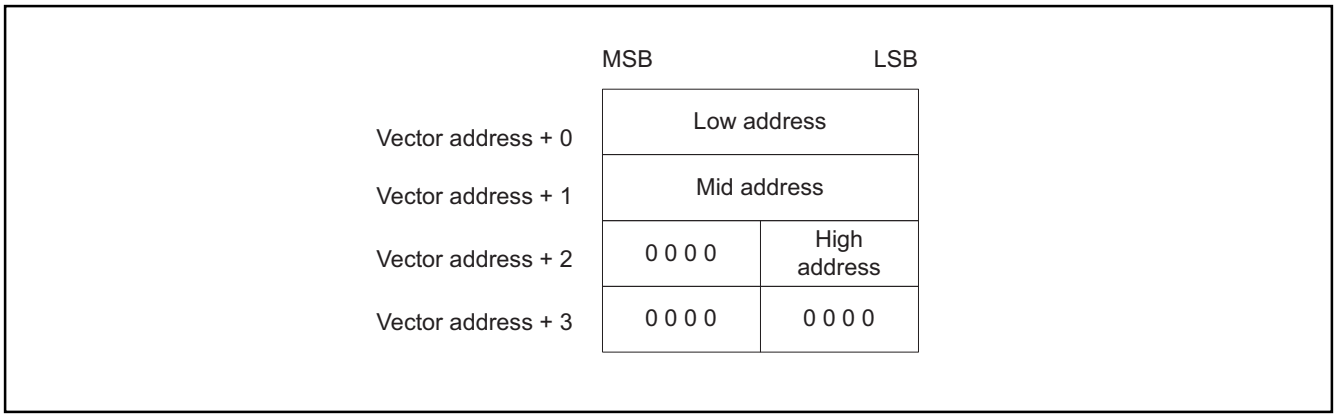



**Figure 1.30. Format for specifying interrupt vector addresses**

### Fixed vector tables

The fixed vector table is a table in which addresses are fixed. The vector tables are located in an area extending from FFFDC16 to FFFFF16. One vector table comprises four bytes. Set the first address of interrupt routine in each vector table. Table 1.27 shows the interrupts assigned to the fixed vector tables and addresses of vector tables.

**Table 1.27. Interrupt vectors with fixed addresses**

| Interrupt source | Vector table addresses Address(L) to Address(H) | Remarks |
|---|---|---|
| Undefined instruction | $FFFDC_{16}$ to $FFFDF_{16}$ | Interrupt on UND instruction |
| Overflow | $FFFE0_{16}$ to $FFFE3_{16}$ | Interrupt on INTO instruction |
| BRK instruction | $FFFE4_{16}$ to $FFFE7_{16}$ | If the vector is filled with $FF_{16}$, program execution starts from the address shown by the vector in the variable vector table |
| Address Match | $FFFE8_{16}$ to $FFFEB_{16}$ | There is an address-matching interrupt enable bit |
| Single Step (Note) | $FFFEC_{16}$ to $FFFEF_{16}$ | Do not use |
| Watchdog timer | $FFFF0_{16}$ to $FFFF3_{16}$ | |
| $\overline{DBC}$ (Note) | $FFFF4_{16}$ to $FFFF7_{16}$ | Do not use |
| $\overline{NMI}$ | $FFFF8_{16}$ to $FFFFB_{16}$ | External interrupt by $\overline{NMI}$ pin |
| Reset | $FFFFC_{16}$ to $FFFFF_{16}$ | |

Note: Interrupts used for debugging purposes only.



Downloaded from Arrow.com.

### Variable vector tables

The addresses in the variable vector table can be modified, according to the user's settings. Before enabling interrupts, the user must load the INTB register with the address of the first entry in the table. The 256-byte area subsequent to the address the INTB indicates becomes the area for the variable vector tables. One vector table comprises four bytes. Set the first address of the interrupt routine in each vector table.

**Table 1.28. Interrupt vectors with variable addresses**

| Software interrupt number | Vector table addresses Address(L) to Address(H) | Interrupt source | Remarks |
|---|---|---|---|
| 0 | +0 to +3 (Note 1) | BRK instruction | Cannot be masked by I flag |
| 1 | +4 to +7 | Key input | |
| 2 | +8 to +11 | UART2 receive / ACK | (Note2) |
| 3 | +12 to +15 | UART1/UART3 Bus collision, Start/stop condition | (Note2) |
| 4 | +16 to +19 | $\overline{\text{INT1}}$ | |
| 5 | +20 to +23 | Timer A1 | |
| 6 | +24 to +27 | USB EP0 | |
| 7 | +28 to +31 | Timer A2 | |
| 8 | +32 to +35 | UART1 receive / ACK / SSI1 receive | (Note2) |
| 9 | +36 to +39 | UART0/UART2 Bus collision, Start/stop condition | (Note2) |
| 10 | +40 to +43 | UART0 receive / ACK / SSI0 receive | (Note2) |
| 11 | +44 to +47 | A/D | |
| 12 | +48 to +51 | DMA0 | |
| 13 | +52 to +55 | UART3 transmit / NACK | (Note2) |
| 14 | +56 to +59 | DMA1 | |
| 15 | +60 to +63 | UART2 transmit / NACK | (Note2) |
| 16 | +64 to +67 | DMA2 | |
| 17 | +68 to +71 | UART1 transmit / NACK / SSI1 transmit | (Note2) |
| 18 | +72 to +75 | DMA3 | |
| 19 | +76 to +79 | UART0 transmit / NACK / SSI0 transmit | (Note2) |
| 20 | +80 to +83 | Timer A0 | |
| 21 | +84 to +87 | UART3 receive / ACK | (Note2) |
| 22 | +88 to +91 | USB suspend | |
| 23 | +92 to +95 | Timer A3 | |
| 24 | +96 to +99 | USB resume | |
| 25 | +100 to +103 | Timer A4 | |
| 26 | +104 to +107 | USB Reset | |
| 27 | +108 to +111 | USB SOF | |
| 28 | +112 to +115 | USB Vbus Detect | |
| 29 | +116 to +119 | USB Function | |
| 30 | +120 to +123 | $\overline{\text{INT2}}$ | |
| 31 | +124 to +127 | $\overline{\text{INT0}}$ | |
| 32 to 63 | +252 to +255 | Software interrupt | Cannot be masked by I flag |

Note 1: Address relative to address in interrupt table base address register (INTB).
Note 2: When $I^2C$ mode is selected, NACK/ACK, start/stop condition detection interrupts are selected.



Downloaded from Arrow.com.

### Interrupt control

The interrupt request bit is set by hardware to "0" when an interrupt request is received. The interrupt request bit can also be set by software to "0". (Do not set to "1".)

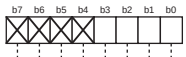
INT0, INT1, and INT2 are triggered by the edges of external inputs. The edge polarity is selected using the polarity select bit. (Other interrupts are described elsewhere.)

An interrupt must first be enabled before it can be used to cancel stop mode.

Peripheral I/O interrupts have their own interrupt control registers. Figure 1.31 shows the interrupt control registers. Table 1.29 shows the addresses of the interrupt control registers.

Interrupt control register

| Symbol | Address | When reset | Symbol | Address | When reset |
|---|---|---|---|---|---|
| KUPIC | $0041_{16}$ | $XXXXX000_2$ | S1TIC | $0051_{16}$ | $XXXXX000_2$ |
| S2RIC | $0042_{16}$ | $XXXXX000_2$ | DM3IC | $0052_{16}$ | $XXXXX000_2$ |
| S13BCNIC | $0043_{16}$ | $XXXXX000_2$ | S0TIC | $0053_{16}$ | $XXXXX000_2$ |
| TA1IC | $0045_{16}$ | $XXXXX000_2$ | TA0IC | $0054_{16}$ | $XXXXX000_2$ |
| EP0IC | $0046_{16}$ | $XXXXX000_2$ | S3RIC | $0055_{16}$ | $XXXXX000_2$ |
| TA2IC | $0047_{16}$ | $XXXXX000_2$ | SUSPIC | $0056_{16}$ | $XXXXX000_2$ |
| S0RIC | $004A_{16}$ | $XXXXX000_2$ | TA3IC | $0057_{16}$ | $XXXXX000_2$ |
| ADIC | $004B_{16}$ | $XXXXX000_2$ | RSMIC | $0058_{16}$ | $XXXXX000_2$ |
| DMOIC | $004C_{16}$ | $XXXXX000_2$ | TA4IC | $0059_{16}$ | $XXXXX000_2$ |
| S3TIC | $004D_{16}$ | $XXXXX000_2$ | RSTIC | $005A_{16}$ | $XXXXX000_2$ |
| DM1IC | $004E_{16}$ | $XXXXX000_2$ | SOFIC | $005B_{16}$ | $XXXXX000_2$ |
| S2TIC | $004F_{16}$ | $XXXXX000_2$ | VBDIC | $005C_{16}$ | $XXXXX000_2$ |
| DM2IC | $0050_{16}$ | $XXXXX000_2$ | USBFIC | $005D_{16}$ | $XXXXX000_2$ |




| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| ILVL0 | Interrupt priority level select bit | b2 b1 b0<br>0 0 0 : Level 0 (interrupt disabled)<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 | ○ | ○ |
| ILVL1 | | | ○ | ○ |
| ILVL2 | | | ○ | ○ |
| IR | Interrupt request bit | 0 : Interrupt not requested<br>1 : Interrupt requested | ○ | ○ (Note) |
| Nothing is assigned. These bits can neither be set nor reset. When read, their contents are indeterminate. | | | – | – |

Note: This bit can only be reset (= 0), but cannot be set ( = 1).

| Symbol | Address | When reset |
|---|---|---|
| INT1IC | $0044_{16}$ | $XX00X000_2$ |
| S1RIC | $0048_{16}$ | $XX00X000_2$ |
| S02BCNIC | $0049_{16}$ | $XX00X000_2$ |
| INT2IC | $005E_{16}$ | $XX00X000_2$ |
| INT0IC | $005F_{16}$ | $XX00X000_2$ |




| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| ILVL0 | Interrupt priority level select bit | b2 b1 b0<br>0 0 0 : Level 0 (interrupt disabled)<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 | ○ | ○ |
| ILVL1 | | | ○ | ○ |
| ILVL2 | | | ○ | ○ |
| IR | Interrupt request bit | 0: Interrupt not requested<br>1: Interrupt requested | ○ | ○ (Note 1) |
| POL | Polarity select bit (Note 2) | 0 : Selects falling edge<br>1 : Selects rising edge | ○ | ○ |
| Reserved bit | | Always set to "0" | ○ | ○ |
| Nothing is assigned. These bits can neither be set nor reset. When read, their contents are indeterminate. | | | – | – |

Note 1: This bit can only be reset (=0), but cannot be set (=1).
Note 2: For S1RIC( $0048_{16}$) and S02BCNIC ( $0049_{16}$), "0" should always be written.

**Figure 1.31. Interrupt control registers control registers**



Downloaded from Arrow.com.

**Table 1.29. Addresses in interrupt control register**

| Interrupt control register | Symbol name | Address | Interrupt control register | Symbol name | Address |
|---|---|---|---|---|---|
| Key input | KUPIC | $0041_{16}$ | DMA2 | DM2IC | $0050_{16}$ |
| UART2 receive / ACK | S2RIC | $0042_{16}$ | UART1 transmit / NACK / Serial Sound Interface 1 transmit | SITIC | $0051_{16}$ |
| UART1 / UART3 Bus collision | S13BCNIC | $0043_{16}$ | DMA3 | DM3IC | $0052_{16}$ |
| INT1 | INT1IC | $0044_{16}$ | UART0 transmit / NACK / Serial Sound Interface 0 transmit | S0TIC | $0053_{16}$ |
| Timer A1 | TA1IC | $0045_{16}$ | Timer A0 | TA0IC | $0054_{16}$ |
| USB EP0 | EP0IC | $0046_{16}$ | UART3 receive / ACK | S3RIC | $0055_{16}$ |
| Timer A2 | TA2IC | $0047_{16}$ | USB Suspend | SUSPIC | $0056_{16}$ |
| UART1 receive / ACK / Serial Sound Interface 1 receive | S1RIC | $0048_{16}$ | Timer A3 | TA3IC | $0057_{16}$ |
| UART0 / UART2 Bus collision | S02BCNIC | $0049_{16}$ | USB Resume | RSMIC | $0058_{16}$ |
| UART0 receive / ACK / Serial Sound Interface 0 receive | S0RIC | $004A_{16}$ | Timer A4 | TA4IC | $0059_{16}$ |
| A/D | ADIC | $004B_{16}$ | USB Reset | RSTIC | $005A_{16}$ |
| DMA0 | DM0IC | $004C_{16}$ | USB SOF | SOFIC | $005B_{16}$ |
| UART3 transmit / NACK | S3TIC | $004D_{16}$ | USB Vbus Detect | VBDIC | $005C_{16}$ |
| DMA1 | DM1IC | $004E_{16}$ | USB Function | USBFIC | $005D_{16}$ |
| UART2 transmit / NACK | S2TIC | $004F_{16}$ | INT2 | INT2IC | $005E_{16}$ |
| | | | INT0 | INT0IC | $005F_{16}$ |

**Rewrite the Interrupt Control Register**

(a) The interrupt control register for any interrupt should be modified in places where no requests for that interrupt may occur. Otherwise, disable the interrupt before rewriting the interrupt control register.

(b) To rewrite the interrupt control register for any interrupt after disabling that interrupt, be careful with the instruction to be used.

- Changing any bit other than the IR bit

- Changing the IR bit
  Depending on the instruction used, the IR bit may not always be cleared to “0” (interrupt not requested). Therefore, be sure to use the MOV instruction to clear the IR bit.

(c) When using the I flag to disable an interrupt, refer to the sample program fragments shown below as you set the I flag. (Refer to (b) for details about rewrite the interrupt control registers in the sample program fragments.)

Examples 1 through 3 show how to prevent the I flag from being set to “1” (interrupts enabled) before the interrupt control register is rewrited, owing to the effects of the internal bus and the instruction queue buffer.



Downloaded from Arrow.com.

**Example 1:Using the NOP instruction to keep the program waiting until the interrupt control register is modified**

```
INT_SWITCH1:
  FCLR    I              ; Disable interrupts.
  AND.B   #00h, 0055h    ; Set the TA0IC register to “00h”.
  NOP                    ;
  NOP
  FSET    I              ; Enable interrupts.
```

The number of NOP instruction is as follows.

PM20=1(1 wait) : 2, PM20=0(2 wait) : 3, when using HOLD function : 4.

**Example 2:Using the dummy read to keep the FSET instruction waiting**

```
INT_SWITCH2:
  FCLR    I              ; Disable interrupts.
  AND.B   #00h, 0055h    ; Set the TA0IC register to “00h”.
  MOV.W   MEM, R0        ; Dummy read.
  FSET    I              ; Enable interrupts.
```

**Example 3:Using the POPC instruction to changing the I flag**

```
INT_SWITCH3:
  PUSHC   FLG
  FCLR    I              ; Disable interrupts.
  AND.B   #00h, 0055h    ; Set the TA0IC register to “00h”.
  POPC    FLG            ; Enable interrupts.
```

**Interrupt Enable Flag (I flag)**

The interrupt enable flag (I flag) controls the enabling and disabling of maskable interrupts. Setting this flag to "1" enables all maskable interrupts; setting it to "0" disables all maskable interrupts. This flag is set to "0" after reset.

**Interrupt Request Bit**

The interrupt request bit is set to "1" by hardware when an interrupt is requested. After the interrupt is accepted and jumps to the corresponding interrupt vector, the request bit is set to "0" by hardware. The interrupt request bit can also be set to "0" by software. (Do not set this bit to "1").

**Interrupt Sequence**

The interrupt sequence, described below, is performed during the period from when an interrupt is accepted to when the interrupt routine is executed.

If an interrupt occurs during execution of an instruction, the processor determines its priority when the execution of the instruction is completed, and transfers control to the interrupt sequence from the next cycle. If an interrupt occurs during execution of either the SMOVB, SMOVF, SSTR or RMPA instruction, the processor temporarily suspends the instruction being executed, and transfers control to the interrupt sequence.

The processor carries out the following in sequence after an interrupt request:

(1) CPU gets the interrupt information (the interrupt number and interrupt request level) by reading address $00000_{16}$.
(2) Saves the contents of the flag register (FLG) as it was immediately before the start of interrupt sequence in the temporary register (Note) within the CPU.
(3) Sets the interrupt enable flag (I flag), the debug flag (D flag), and the stack pointer select flag (U flag) to "0" (the U flag, however does not change if the INT instruction, in software interrupt numbers 32 through 63, is executed).
(4) Saves the contents of the temporary register (Note) within the CPU in the stack area.
(5) Saves the contents of the program counter (PC) in the stack area.
(6) Sets the interrupt priority level of the accepted instruction in the IPL.

After the interrupt sequence is completed, the processor resumes executing instructions from the first address of the interrupt routine.

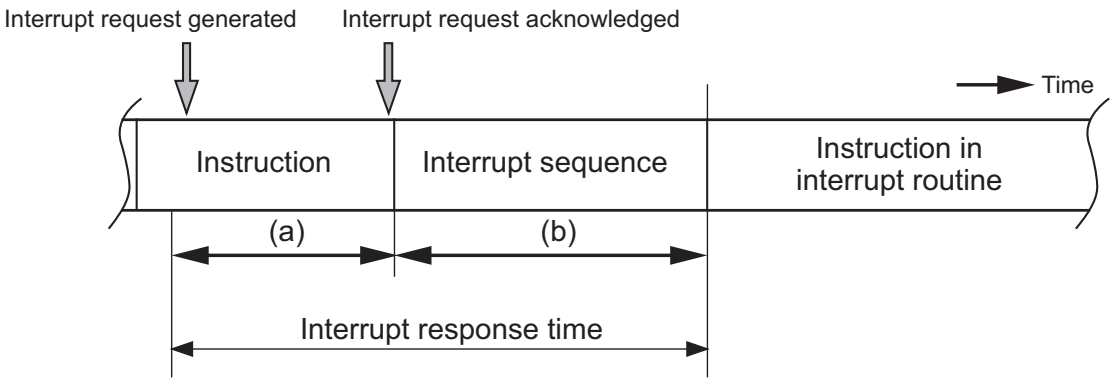
Note: This register cannot be utilized by the user.



Downloaded from Arrow.com.

### Interrupt Response Time

'interrupt response time' is the period between when an interrupt occurs and when the first instruction within the interrupt routine has been executed. This time comprises the period from the occurrence of an interrupt to the completion of the instruction under execution at that moment (a) and the time required for executing the interrupt sequence (b). Figure 1.33 shows the interrupt response time.




(a) Time from interrupt request is generated to when the instruction then under execution is completed.
(b) Time in which the instruction sequence is executed.

**Figure 1.33. Interrupt response time**

Time (a) is dependent on the instruction under execution. Thirty cycles is the maximum required for the DIVX instruction (without wait).
Time (b) is as shown in Table 1.30 . Figure 1.34 shows the time required for executing the interrupt sequence.

**Table 1.30. Time required for executing the interrupt sequence**

| Interrupt vector address | Stack pointer (SP) value | 16-bit bus, without wait | 8-bit bus, without wait |
|---|---|---|---|
| Even | Even | 18 cycles (Note 1) | 20 cycles (Note 1) |
| Even | Odd | 19 cycles (Note 1) | 20 cycles (Note 1) |
| Odd (Note 2) | Even | 19 cycles (Note 1) | 20 cycles (Note 1) |
| Odd (Note 2) | Odd | 20 cycles (Note 1) | 20 cycles (Note 1) |

Note 1: Add 2 cycles for DBC interrupt.
Add 1 cycle for either an address match interrupt or a single-chip interrupt.
Note 2: Locate an interrupt vector address in an even address if possible.



Downloaded from Arrow.com.

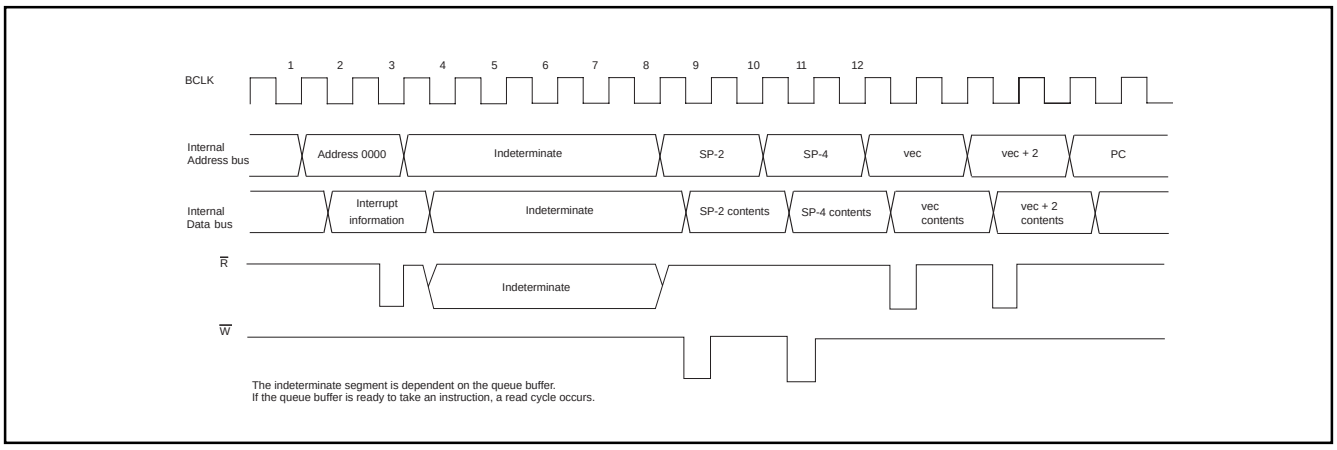



**Figure 1.34. Interrupt sequence timing**

### Returning from an Interrupt Routine

Executing the REIT instruction at the end of an interrupt routine restores the contents of the flag register (FLG) as it was immediately before the start of the interrupt sequence and the contents of the program counter (PC), both of which were saved in the stack area. Then control returns to the program that was being executed before the acceptance of the interrupt request, so that the suspended process resumes.

Return the other registers that were saved by software within the interrupt routine using the POPM instruction or a similar instruction before executing the REIT instruction.

### Interrupt priority

The order of priority when two or more interrupts are generated simultaneously is determined by both hardware and software.

The interrupt priority levels determined by hardware are:

**$\overline{\text{RESET}}$ > $\overline{\text{NMI}}$ > DBC > Watchdog Timer > Peripheral I/O > Single step > Address match**

The interrupt priority levels determined by software are set in the interrupt control registers.

When two or more interrupts are generated simultaneously, the interrupt with the higher software priority is selected. However, if the interrupts have the same software priority level, the interrupt is selected according to the hardware priority set in the circuit.

The selected interrupt is accepted only when the priority level is higher than the processor interrupt priority level (IPL) in the flag register (FLG) and the interrupt enable flag (I flag) is "1" Note that the reset, $\overline{\text{NMI}}$, $\overline{\text{DBC}}$, watchdog timer, single-step, address-match, BRK instruction, overflow, and undefined instruction interrupts are accepted regardless of the interrupt enable flag (I flag).

### Interrupt Priority Level Select Bit and Processor Interrupt Priority Level (IPL)

Set the interrupt priority level using the interrupt priority level select bits, which consists of three interrupt control register bits. When an interrupt request occurs, the interrupt priority level is compared with the IPL of the CPU flag register. The interrupt is enabled only when the priority level of the interrupt is higher than the IPL. Therefore, setting the interrupt priority level to "0" disables the interrupt.



Downloaded from Arrow.com.

Table 1.31 shows the settings of interrupt priority levels and Table 1.32 shows the interrupt levels enabled, according to the contents of the IPL.

The following are conditions under which an interrupt is accepted:

- interrupt enable flag (I flag) = 1
- interrupt request bit = 1 (set by hardware)
- interrupt priority level > IPL

The interrupt enable flag (I flag), the interrupt request bit, the interrupt priority select bit, and the IPL are independent, and they are not affected by one another.

**Table 1.31. Interrupt priority level settings**

| Interrupt priority level select bit | Interrupt priority level | Priority order |
|---|---|---|
| b2 b1 b0<br>0 0 0 | Level 0 (interrupt disabled) | ——— |
| 0 0 1 | Level 1 | Low |
| 0 1 0 | Level 2 | |
| 0 1 1 | Level 3 | |
| 1 0 0 | Level 4 | |
| 1 0 1 | Level 5 | |
| 1 1 0 | Level 6 | |
| 1 1 1 | Level 7 | High |

**Table 1.32. Interrupt levels enabled according to the contents of the IPL**

| IPL | Enabled interrupt priority levels |
|---|---|
| IPL2 IPL1 IPL0<br>0 0 0 | Interrupt levels 1 and above are enabled |
| 0 0 1 | Interrupt levels 2 and above are enabled |
| 0 1 0 | Interrupt levels 3 and above are enabled |
| 0 1 1 | Interrupt levels 4 and above are enabled |
| 1 0 0 | Interrupt levels 5 and above are enabled |
| 1 0 1 | Interrupt levels 6 and above are enabled |
| 1 1 0 | Interrupt levels 7 and above are enabled |
| 1 1 1 | All maskable interrupts are disabled |



Downloaded from Arrow.com.

### Interrupt resolution circuit

When two or more interrupts are generated simultaneously, this circuit selects the interrupt with the highest priority. Figure 1.35 shows the circuit that judges the interrupt priority.

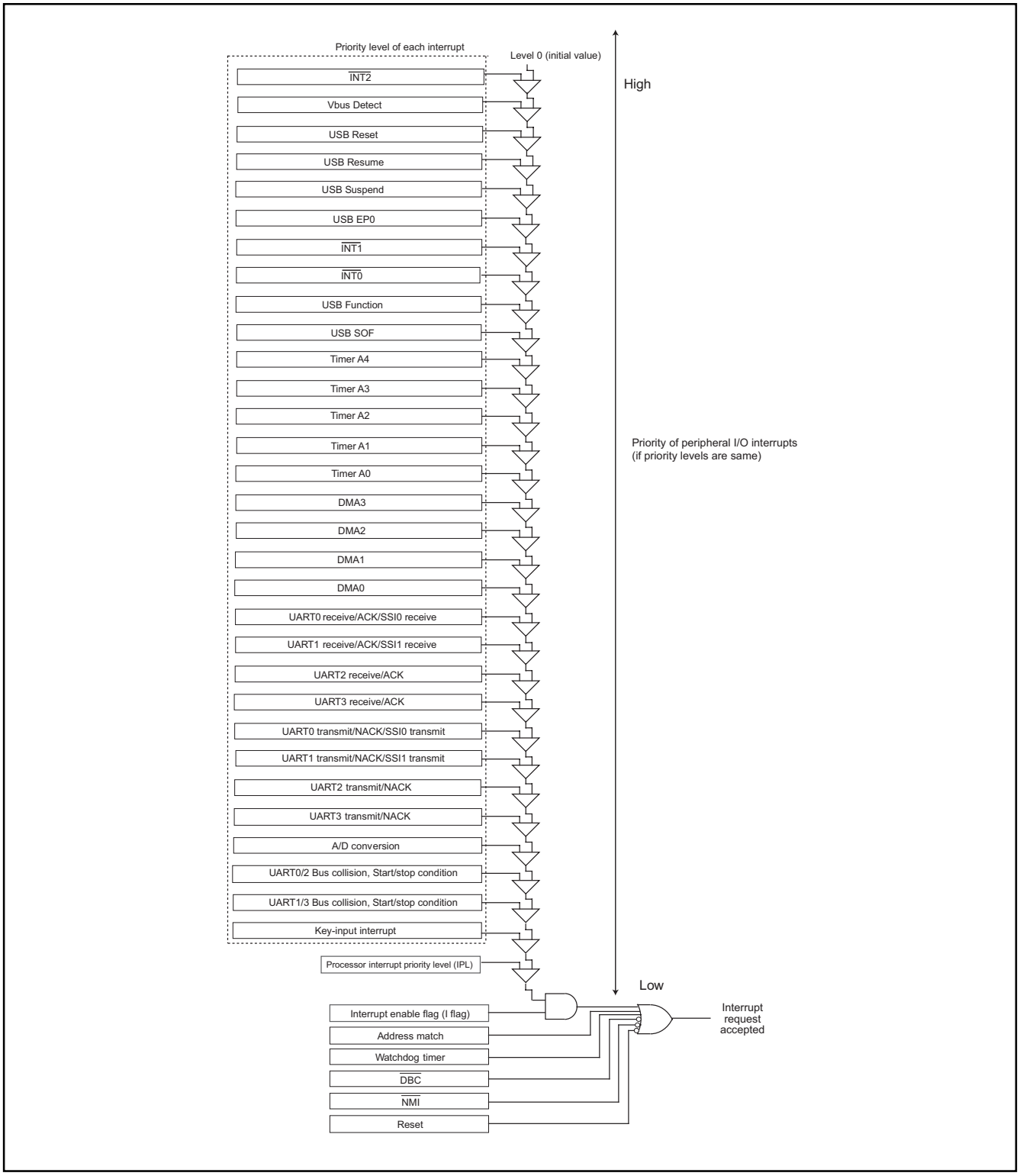



**Figure 1.35. Interrupt resolution circuit**



Downloaded from Arrow.com.

### Flag changes

When an interrupt request is received, the stack pointer select flag (U flag) changes to "0" and the flag register (FLG) and program counter (PC) are saved to the stack area indicated by the interrupt stack pointer (ISP). Thereafter, the interrupt enable flag (I flag) and debug flag (D flag) change to "0" and the processor interrupt priority level (IPL) at the flag register (FLG) is replaced by the priority level of the received interrupt. However, when interrupt requests are received for software interrupts 32 to 63, the flag register (FLG) and program counter (PC) are saved to the stack shown by the stack pointer select flag (U flag) at the time the interrupt was received. The stack pointer select flag (U flag) does not change. The value of the processor interrupt priority level (IPL) in the flag register (FLG) differs in the case of reset, $\overline{\text{NMI}}$, $\overline{\text{DBC}}$, watchdog timer, single-step, address-match, BRK instruction, overflow, and undefined instruction interrupts. Table 1.34 shows how the IPL changes when interrupt requests are received.

**Table 1.34. Change of IPL state when interrupt request are accepted**

| Interrupt | Change of IPL |
|---|---|
| Reset | Level 0 ($000_2$), is set |
| $\overline{\text{NMI}}$ | Level 7 ($111_2$), is set |
| $\overline{\text{DBC}}$ | Does not change |
| Watchdog timer | Level 7 ($111_2$), is set |
| Single step | Does not change |
| Address match | Does not change |
| Software interrupt | Does not change |



Downloaded from Arrow.com.

**INT interrupt**

$\overline{INT0}$ to $\overline{INT2}$ are triggered by the edges of external inputs. The edge polarity is selected using the polarity select bit in the interrupt control register ($0044_{16}$, $005E_{16}$, $005F_{16}$) and the polarity switching bit in the interrupt request cause select register ($035F_{16}$).

For an external interrupt input, an interrupt can be generated both at the rising edge and at the falling edge by setting "1" in the INTi interrupt polarity switching bit of the interrupt request cause select register ($035F_{16}$). To select one edge, set the polarity switching bit of the corresponding interrupt request cause select register to "one edge" ("0"), and set the polarity select bit in the interrupt control register to rising edge or falling edge.

Interrupt request cause select register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: IFSR | Address: $035F_{16}$ | When reset: $00_{16}$

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| IFSR0 | $\overline{INT0}$ interrupt polarity swiching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR1 | $\overline{INT1}$ interrupt polarity swiching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR2 | $\overline{INT2}$ interrupt polarity swiching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate when read. | | | — | — |
| IFSR6 | Bus collision interrupt request cause select bit 0 | 0 : UART0 Bus collision<br>1 : UART2 Bus collision | ○ | ○ |
| IFSR7 | Bus collision interrupt request cause select bit 1 | 0 : UART1 Bus collision<br>1 : UART3 Bus collision | ○ | ○ |

**Figure 1.36 shows the interrupt request cause select register.**

**$\overline{NMI}$ Interrupt**

An $\overline{NMI}$ interrupt is generated when the input to the $P8_5/\overline{NMI}$ pin changes from "H" to "L". The $\overline{NMI}$ interrupt is a non-maskable external interrupt. The pin level can be checked in the Port $P8_5$ register (bit 5 at address $03F0_{16}$). This pin cannot be used as a normal port input.

Notes: When not intending to use the $\overline{NMI}$ function, be sure to connect the $\overline{NMI}$ pin to Vcc. Because the $\overline{NMI}$ interrupt is non-maskable, it cannot be disabled.

When the $\overline{NMI}$ pin input is "L", do not set the microcomputer in stop mode or wait mode. The $\overline{NMI}$ interrupt is triggered by the falling edge, so the "L" level does not need to be maintained longer than necessary.

**Key-Input Interrupt**

A Key input interrupt can be generated by a falling edge, rising edge or both edges input to any Port 10 pin. It can also be used as a Key-on wake up function for canceling the wait mode or stop mode. Figure 1.37 shows the block diagram of the Key-input interrupt.

Figure 1.38 shows the Key-input mode register. It is possible to select both edges or the falling edge of the Key input interrupt for P10 with bits 0 and 1 of this register. This register is also used to enable or disable Port 10 pins that are to be used for Key-input interrupts. Port 10 can be configured with pull-up resistors using the pull-up control resistor.



Downloaded from Arrow.com.

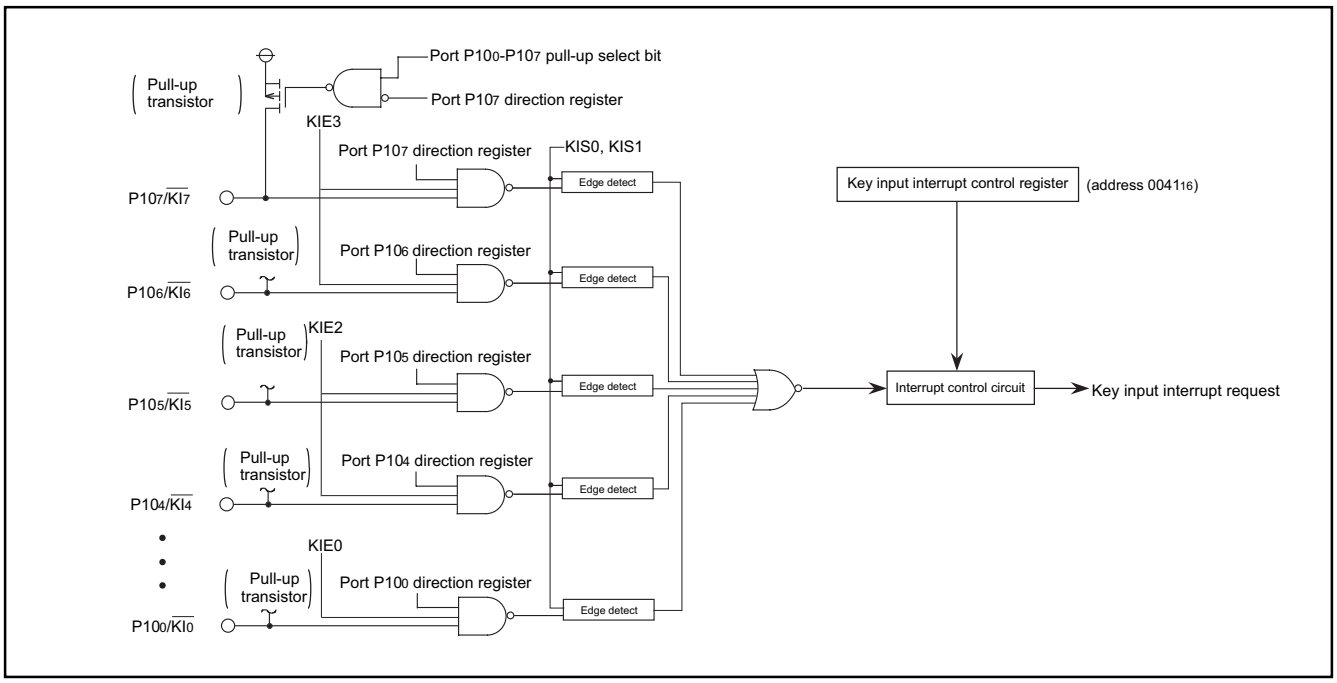



**Figure 1.37. Block diagram of Key-input interrupt**

Key-input mode register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: KUPM | Address: $0126_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| KIS0 | P10 Key-input edge select 0 | b1 b0<br>0 0 : Falling edge<br>0 1 : Rising edge<br>1 0 : Two edges<br>1 1 : Reserved | O | O |
| KIS1 | P10 Key-input edge select 1 | | O | O |
| KIE0 | $P10_0$ and $P10_1$ Key-input enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| KIE1 | $P10_2$ and $P10_3$ Key-input enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| KIE2 | $P10_4$ and $P10_5$ Key-input enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| KIE3 | $P10_6$ and $P10_7$ Key-input enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" if read. | | | – | – |

**Figure 1.38. Key-input mode register**

### Enable/Disable

The key-input interrupts can be enabled and disabled in pairs using the Key-input mode register ($03F9_{16}$) and Key-input interrupt register ($0041_{16}$). The key-input interrupt is affected by the interrupt priority level (IPL) and the interrupt enable flag (I flag). The input signal edge that triggers the Key-input interrupt can be selected by setting the P10 Key-input edge select bits (bits 0 and 1 at $03F9_{16}$). Also, make sure to set the port direction for the enabled Key-input interrupt pins to input.



Downloaded from Arrow.com.

### Occurrence timing of the key-input interrupt

With the Key-input interrupt enabled, Port 10 pins that are enabled in the Key-input mode register are set to input mode and become Key-input interrupt pins ($KI_0$ through $KI_7$). A Key-input interrupt occurs when the selected edge is input to a Key-input interrupt pin. At this moment, the level of other key-input interrupt pins must be "H" No interrupt occurs when the level of any other key-input interrupt pins is "L".

### Determining a key-input interrupt

A key-input interrupt occurs when the selected edge is input to one of 8 pins, (if they are all enabled in the Key-input mode register) but each pin has the same vector address. Therefore, read the input level of Port P10 in the key-input interrupt routine to determine the interrupted pin.

### Related registers

Figure 1.39 shows the memory map of key-input interrupt-related registers.

| Address | Register name | Acronym |
|---|---|---|
| $0040_{16}$ | | |
| $0041_{16}$ | Key input interrupt register | KUPIC |
| | | |
| $03F0_{16}$ | | |
| $03F1_{16}$ | | |
| $03F2_{16}$ | | |
| $03F3_{16}$ | | |
| $03F4_{16}$ | Port 10 | P10 |
| $03F5_{16}$ | | |
| $03F6_{16}$ | Port 10 direction register | PD10 |
| $03F7_{16}$ | | |
| $03F8_{16}$ | | |
| $03F9_{16}$ | Key-input mode register | KUPM |
| $03FA_{16}$ | | |
| $03FB_{16}$ | | |
| $03FC_{16}$ | | |
| $03FD_{16}$ | | |
| $03FE_{16}$ | Pull-up control register 2 | PUR2 |
| $03FF_{16}$ | | |

**Figure 1.39. Memory map of Key-input interrupt-related registers**



Downloaded from Arrow.com.

### Address-Match Interrupt

An address-match interrupt is generated when the address-match interrupt address register contents match the program counter value. Two address-match interrupts can be set, each of which can be enabled and disabled by an address-match interrupt enable bit. The interrupt enable flag (I flag) does not affect address-match interrupts and processor interrupt priority level (IPL).

Note: When the external data bus width is set to 8 bits, the address match interrupt cannot be used for external areas.

Figure 1.40 shows the address-match interrupt-related registers.

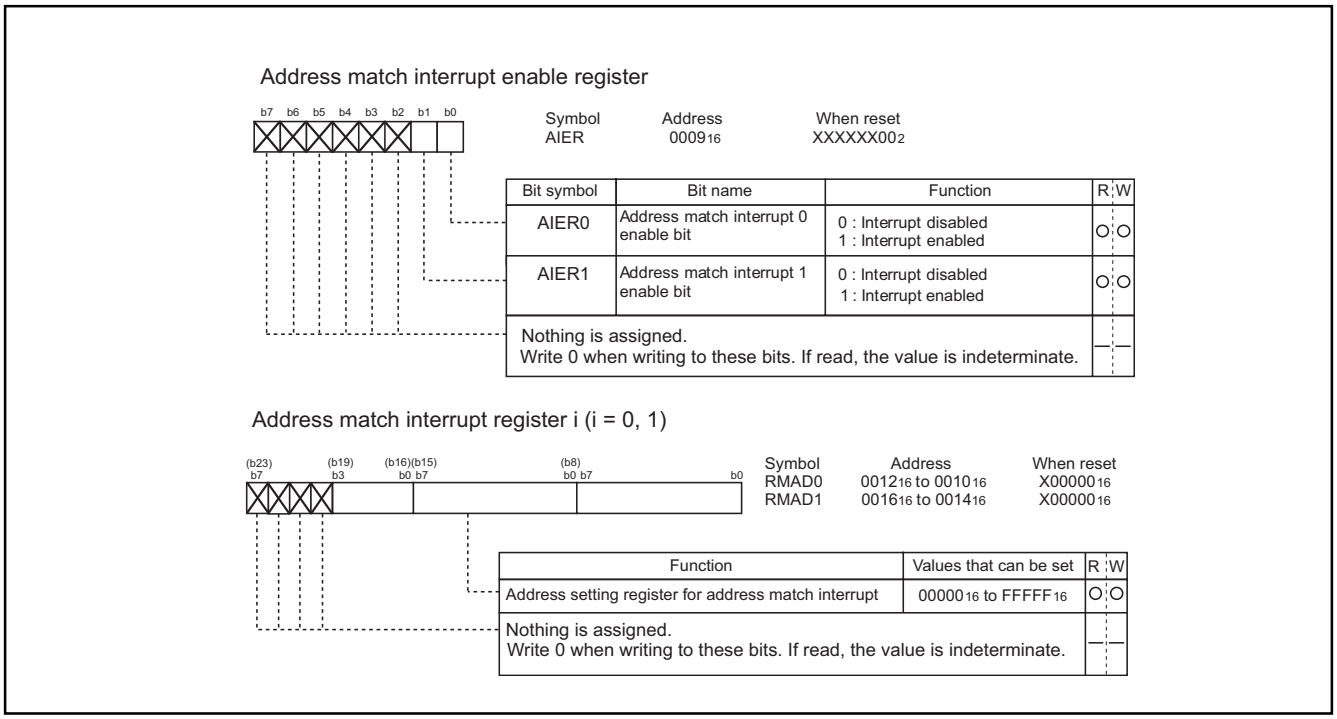



**Figure 1.40. Address-match interrupt-related register Interrupt precautions**

## Precautions

### Reading address 0000016

When maskable interrupt occurs, the CPU reads the interrupt information (the interrupt number and interrupt request level) in the interrupt sequence. The interrupt request bit of the interrupt written in address 0000016 will then be set to "0". Do not read address 0000016 by software. Reading address 0000016 by software sets enabled highest priority interrupt source request bit to "0". Though the interrupt is generated, the interrupt routine may not be executed.

### Setting the stack pointer

The value of the stack pointer immediately after reset is initialized to 000016. Accepting an interrupt before setting a value in the stack pointer may cause program runaway. Be sure to set a value in the stack pointer before accepting an interrupt.

When using the $\overline{\text{NMI}}$ interrupt, initialize the stack pointer at the beginning of a program. Generating any interrupts including the $\overline{\text{NMI}}$ interrupt is prohibited for the first instruction immediately after reset.



Downloaded from Arrow.com.

**The $\overline{\text{NMI}}$ interrupt**

The $\overline{\text{NMI}}$ interrupt can not be disabled. Be sure to connect $\overline{\text{NMI}}$ pin to Vcc with a pull-up resistor if unused. Do not go into stop mode when the $\overline{\text{NMI}}$ pin set to "L".

The $\overline{\text{NMI}}$ pin also serves as $P8_5$, which is exclusively an input. Reading the contents of the P8 register allows the pin value to be read. Reading this pin is only to be used for establishing the pin level when the $\overline{\text{NMI}}$ interrupt is input.

Do not reset the CPU with the input to the $\overline{\text{NMI}}$ pin in the "L" state.

Do not attempt to go into stop mode when the input to the $\overline{\text{NMI}}$ pin is in "L" state. When the input to the $\overline{\text{NMI}}$ is in "L" state, CM10 is fixed to "0" thereby refusing to go into stop mode.

Do not attempt to go into wait mode when the input to the $\overline{\text{NMI}}$ pin is in "L" state. When the input to the $\overline{\text{NMI}}$ pin is in"L" state, the CPU stops but the oscillation does not. This action does not save power. When this occurs, the CPU is returned to the normal state by a later interrupt.

Signals input to the $\overline{\text{NMI}}$ pin require an "L" level of (2 clocks + 300nS) or more from the operation clock of the CPU.

**External interrupt**

Either an "H" or "L" level of at least 250 ns width is necessary for the signal input to pins $\overline{\text{INT0}}$ to $\overline{\text{INT2}}$ regardless of the CPU operation clock.

When the polarity of the $\overline{\text{INT0}}$ to $\overline{\text{INT2}}$ pins is changed, the interrupt request bit is sometimes set to "1". After changing the polarity, reset the interrupt request bit to "0". Figure 1.41 shows the procedure for changing the INT interrupt generate factor.

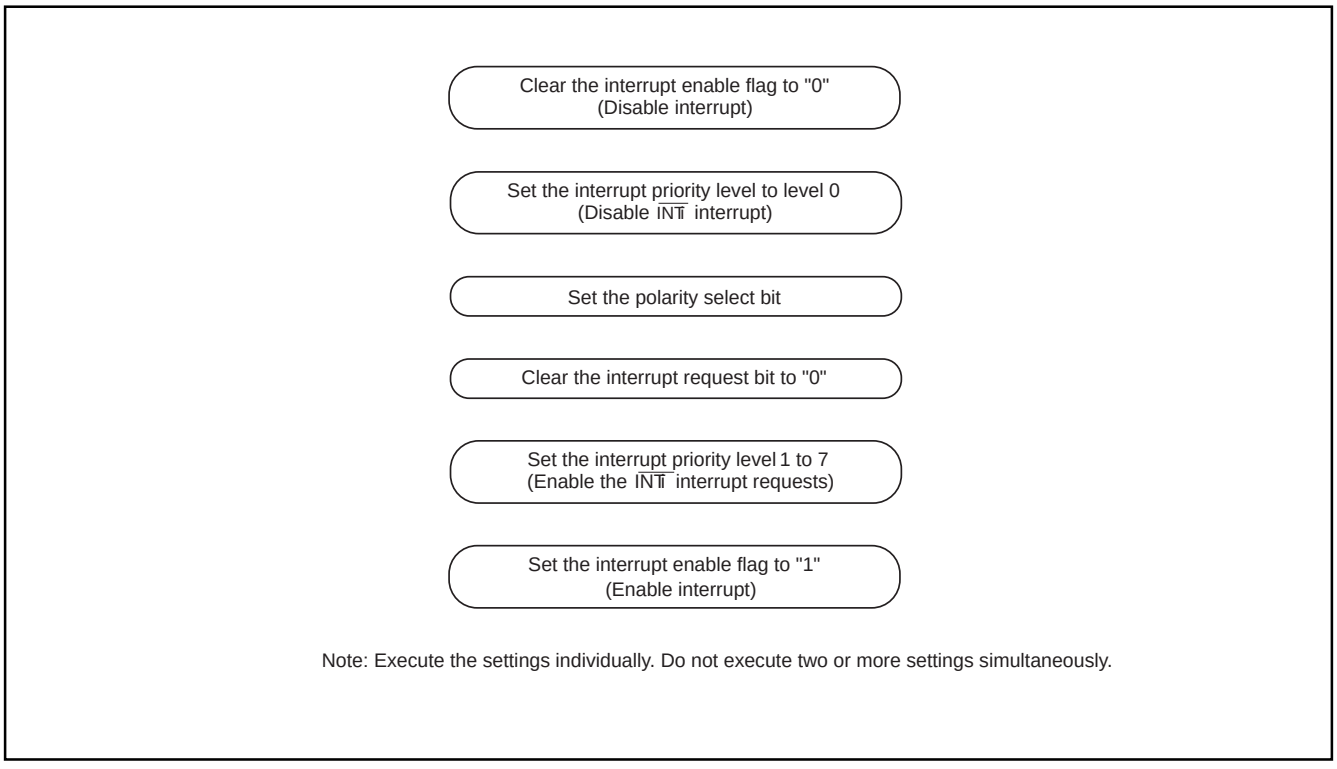



**Figure 1.41. Switching condition of INT interrupt request**



Downloaded from Arrow.com.

**Clearing the Interrupt request bit**

Even when the IR bit (bit 3 of the interrupt control register) is cleared to "0" (interrupt not requested), it may not actually get cleared to "0" depending on the instruction used to clear it. Therefore, use the MOV instruction to clear the IR bit.

Rewriting the interrupt control register

Rewrite the interrupt control register so that it does not generate an interrupt request for that register. If an interrupt request occurs, rewrite the interrupt control register after the interrupt is disabled. Some program examples are described below.

When an instruction to rewrite the interrupt control register is executed but the interrupt is disabled, the interrupt request bit is not always set even if the interrupt request for that register has been generated. This will depend on the instruction. If this creates problems, use the instructions below to change the register.

Instructions: AND, OR, BCLR, BSET

Examples 1 through 3 show how to prevent the I flag from being set to "1" (interrupts enabled) before the interrupt control register is rewritting, due to the effects of the internal bus and the instruction queue buffer.

**Example 1:**

```
INT_SWITCH1:
        FCLR    I           :Disable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.
        NOP                 ;Four NOP instructions are required when using the HOLD function.
        NOP
        FSET    I           ;Enable interrupts.
```

**Example 2:**

```
INT_SWITCH2:
        FCLR    I           :Disable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.
        MOV.W   MEM, R0     ;Dummy read.
        FSET    I           ;Enable interrupts.
```

**Example 3:**

```
INT_SWITCH3:
        PUSHC   FLG         ;Push Flag register onto stack
        FCLR    I           ;Diable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.‘
        POPC    FLG         ;Enable interrupts.
```

**The reason why two NOP instructions (four using the HOLD function) or a dummy read is inserted before "FSET I" in Examples 1 and 2, is to prevent the interrupt enable flag I from being set before the interrupt control register is rewritten due to the effects of the instruction queue.**



Downloaded from Arrow.com.

## Watchdog Timer

The watchdog timer can detect a runaway program. It is a 15-bit counter that decrements using the clock derived from dividing the BCLK by the prescaler. A watchdog timer interrupt is generated when an underflow occurs in the watchdog timer. The watchdog timer interrupt is a non-maskable interrupt.

When XIN is selected for BCLK, bit 7 (WDC7) of the watchdog timer control register (address $000F_{16}$) selects the prescaler divide ratio to be either 16 or 128. When XCIN is selected for BCLK, the prescaler divide ratio is set to 2 regardless of WDC7. The watchdog timer cycle can be calculated as follows:

When XIN chosen for BCLK:

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (16 or 128)} \times \text{Watchdog timer count (32768)}}{\text{BCLK}}$$

When XCIN chosen for BCLK:

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (2)} \times \text{Watchdog timer count (32768)}}{\text{BCLK}}$$

Example:

When BCLK is 12 MHz and the prescaler divide ratio is set to 16, the monitor timer cycle is approximately 43.69 ms.

The watchdog timer is initialized by writing to the watchdog timer start register (address $000E_{16}$), and when a watchdog timer interrupt request is generated.

The prescaler is initialized only when the microcomputer is reset. After a reset, the watchdog timer and prescaler are both stopped. The count is started by writing to the watchdog timer start register (address $000E_{16}$).

The watchdog timer and the prescaler stop in stop mode, wait mode, and hold state. After exiting these modes, counting starts from the remaining value. Figure 1.42 shows the block diagram of the watchdog timer. Figure 1.43 shows the watchdog timer-related registers.



Downloaded from Arrow.com.

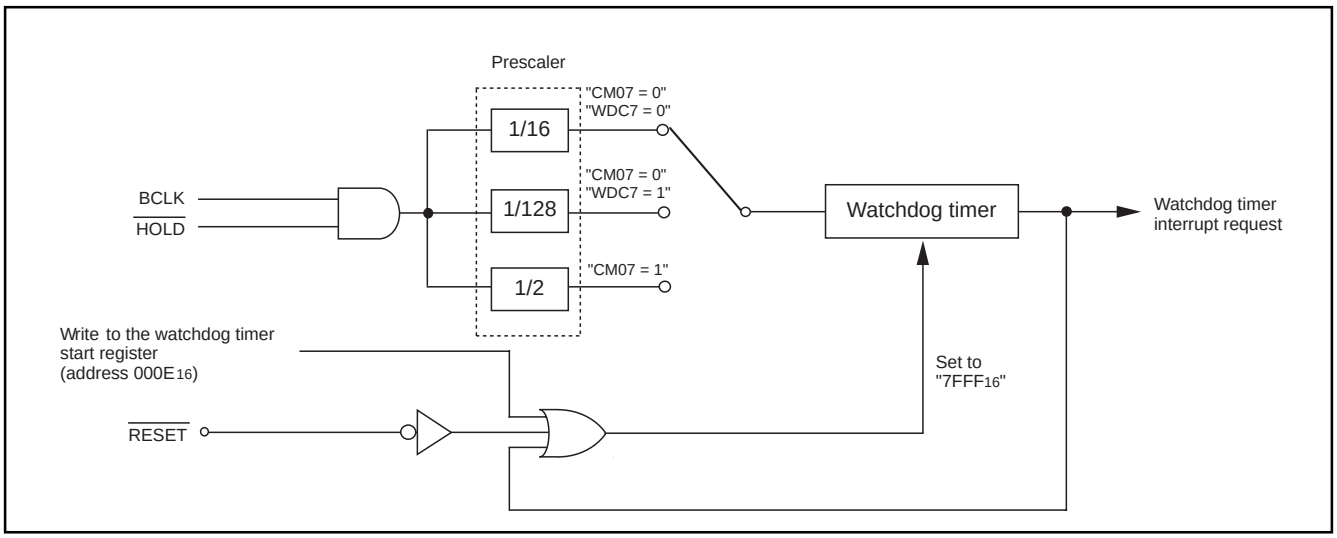



**Figure 1.42. Block diagram of Watchdog timer**

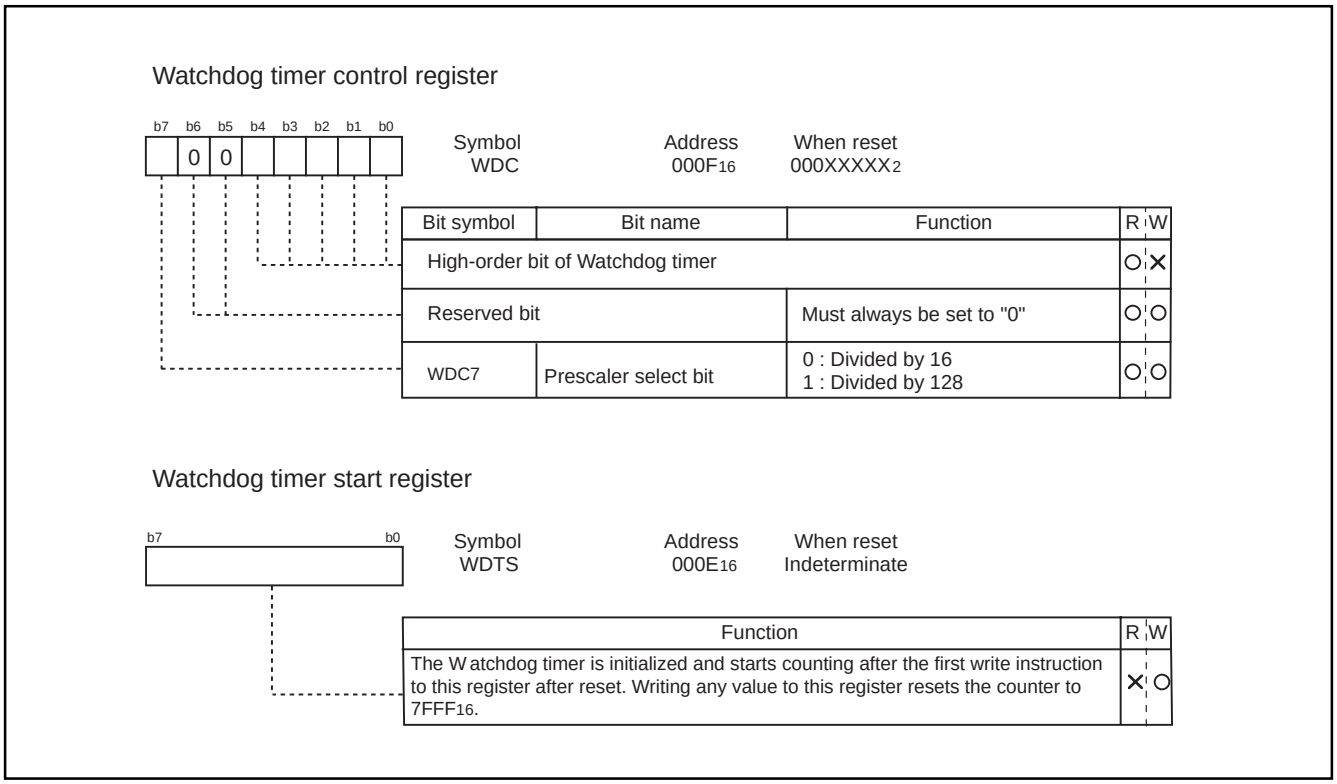



**Figure 1.43. Watchdog timer control and start registers**



Downloaded from Arrow.com.

## Universal Serial Bus

### Features

- USB Specification Revision 2.0 compliant
- Support of full-speed operation (12 Mbps)
- Support of all USB transfer types: ..... Isochronous
  ..... Bulk
  ..... Control
  ..... Interrupt
- Built-in 3.25 Kbyte FIFO as endpoint buffer: ..... Endpoint 0: IN/OUT 128-byte/128-byte
  ..... Endpoint 1: IN/OUT Programmable
  ..... Endpoint 2: IN/OUT Programmable
  ..... Endpoint 3: IN/OUT Programmable
  ..... Endpoint 4: IN/OUT Programmable
- 9 endpoints - control endpoint (EP0 - bidirectional) plus four IN and four OUT endpoints
- Control endpoint (EP0) continuous transfer mode
- Programmability of transfer type and buffer size for 8 of the 9 endpoints (EP1 - EP4 IN & OUT)
  - Transfer type: ..... Bulk, Isochronous or Interrupt

  Single or double buffer selectable
  - Buffer size: ..... Maximum 1K bytes

  When in double buffer mode, effective maximum buffer size up to 2 x 1K bytes
- Bulk endpoints continuous transfer mode
- SOF output and interrupt generation with artificial SOF capability (in the event of corrupt SOF packet)
- 8- or 16-bit CPU access to the FIFO and registers

### USB Interrupts

There are six USB interrupts in this device:

- EP0 Interrupt (multiple-trigger events)
- USB Function Interrupt (multiple sources)
- USB Reset Interrupt
- USB Resume Interrupt
- USB Suspend Interrupt
- USB Start-of-Frame (SOF) Interrupt

The first five interrupts are used to control the data flow and USB power consumption. The SOF interrupt is used to monitor the transfer of isochronous (ISO) data. Setting the corresponding bit in the Interrupt Control Register for each interrupt enables each of the six USB interrupts.



Downloaded from Arrow.com.

The USB Function Interrupt has multiple interrupt sources that can be enabled within the USB Function Interrupt Enable Register (USBIE).

**EP0 Interrupt**

The EP0 interrupt is generated when one of the following events occur:

- A data set is successfully received
- A data set is successfully sent
- EP0CSR3 (DATA_END) flag is cleared. This event is maskable and the default is masked.
- A control transfer ends prematurely (i.e., the USB FCU sets the SETUP_END bit).

**USB Function Interrupt**

The USB Function interrupt can be triggered by:

- The interrupts from eight endpoints (EP1-EP4 IN/OUT). The interrupts indicate if a data set was either sent or received.
- A data flow error from any of the nine endpoints (including EP0)
- The enabling of any IN endpoint (EP1-EP4 IN).
- The corruption of the final ACK of a Control Read transfer's Data Stage.

Each endpoint interrupt is enabled by setting the corresponding bit in the USB Interrupt Enable register (USBIE).

Interrupt status flags associated with each source are contained in USB Interrupt Status register (USBIS).

**USB Reset Interrupt**

A USB Reset Interrupt is generated when the USB Function Control Unit (USB FCU) sees a SE0 present on D+/D- for at least 2.5us. When a reset signal is detected by the USB FCU, an internal reset pulse is also generated to reset all USB internal registers to the default values.
When the CPU recognizes a USB Reset Interrupt, it re-initializes the USB FCU to ensure that the USB operation functions properly.
The USB Reset Interrupt Control register (RSTIC) contains the USB Reset Interrupt request bit and interrupt priority select bits used to enable the interrupt and set the software priority level.

**USB Resume Interrupt**

A USB Resume Interrupt is generated when the USB FCU is in the suspend state and detects non-idle signaling on the D+/D-.
The USB Resume Interrupt Control register (RSMIC) contains the USB Resume Interrupt request bit and interrupt priority select bits used to enable the interrupt and set its software priority level.

**USB SOF Interrupt**

The USB SOF (Start-Of-Frame) Interrupt is used to control the transfer of isochronous data. The USB FCU generates a USB SOF Interrupt request when a start-of-frame packet is received.
Because the start-of-frame packet could be corrupted, a new frame might start without successful reception of the SOF packet. For this reason, an artificial SOF is provided. The frame timer signals a time out when a SOF packet is not received within the allotted time. The device generates an SOF interrupt once every frame. Setting bit 2 of the USB ISO Control Register to a "1" enables the artificial SOF function.
Register SOFIC contains the USB SOF Interrupt's request bit and interrupt priority select bits that are used to enable the interrupt and set its software priority level.



Downloaded from Arrow.com.

### USB Suspend Interrupt

A USB Suspend Interrupt is generated when the USB FCU does not detect any bus activity on D+/D- (in J-state) for at least 3ms.
The USB Suspend Interrupt Control register (SUSPIC) contains the USB Suspend Interrupt request bit and interrupt priority select bits that are used to enable the interrupt and set its software priority level.

## USB Endpoint FIFOs

The USB FCU has a built-in 3.25 K bytes FIFO as an endpoint buffer. The EP0 (control endpoint) FIFO occupies a fixed location (from 3K - 3.25K) with fixed buffer sizes (128 bytes each) for its IN and OUT data transfers. The other 8 endpoints (EP1 to EP4 IN and OUT) share a 3K bytes buffer. Each endpoint's FIFO size and starting location (64 bytes) are programmable by the user. The sum of the 8 endpoint FIFOs can not exceed 3K bytes (3072 bytes).
Note: Throughout the USB Block specification, "data packet" is generally used when continuous mode is disabled; "data set" (one or more data packets) is generally used when continuous mode is enabled. If a description applies for both noncontinuous mode and continuous mode, "data set" is used.
Throughout the whole USB Block Specification, "FIFO" and "Buffer" are generally interchangeable terms.

### EP0 FIFO Operation

The CPU writes data to the EP0 IN FIFO Data Register. The write pointer automatically increments by 2 in word accessing mode or by 1 in byte accessing mode after a write. The CPU must only write data to the EP0 IN FIFO Data Register and "1" to the SET_IN_BUF_RDY bit of the EP0 CSR when the IN_BUF_RDY flag is a "0". When a NULL packet is required to complete a control read request, the CPU must write "1" to the SET_IN_BUF_RDY bit of EP0_CSR without writing data to the EP0 IN FIFO Data Register.
Continuous transfer modes are available for EP0 Control Transfers.

#### EP0 IN FIFO with control read continuous transfer mode disabled

The CPU writes "1" to the SET_IN_BUF_RDY bit of the EP0 CSR after the CPU finishes writing a data packet to the FIFO, this updates the IN_BUF_RDY flag to "1". The USB FCU updates the IN_BUF_RDY flag to "0" after the packet has been successfully transmitted to the host.

#### EP0 IN FIFO with control read continuous transfer mode enabled

The CPU writes "1" to the SET_IN_BUF_RDY bit of the EP0 CSR after the CPU finishes writing a data set (up to 128 bytes) to the FIFO. This updates the IN_BUF_RDY flag to "1". The USB FCU sends out data packets equal to the EP0 MAXP size one at a time, except for the last packet if the data set in the FIFO is not a multiple of EP0 MAXP. In this case the USB FCU sends a short packet. The USB FCU updates the IN_BUF_RDY flag to "0" after the data set has been successfully transmitted to the host.
The CPU reads data from EP0 OUT FIFO Data Register. The read pointer automatically increments by 2 in word accessing mode or by 1 in byte accessing mode after a read. The CPU must only read data from the EP0 OUT FIFO when the OUT_BUF_RDY flag of the EP0_CSR is "1".

When a SETUP packet is received, an EP0 interrupt is generated (both OUT_BUF_RDY and SETUP flags are set) regardless of the continuous transfer mode bit setting.

#### EP0 OUT FIFO with control write continuous transfer mode disabled

The USB FCU updates the OUT_BUF_RDY flag to "1" after it has successfully received a data packet from the host. The CPU writes "1" to CLR_OUT_BUF_RDY after the data packet has been unloaded from the FIFO by the CPU (updates the OUT_BUF_RDY flag to a "0").



Downloaded from Arrow.com.

**EP0 OUT FIFO with control write continuous transfer mode enabled**

The USB FCU updates the OUT_BUF_RDY flag to "1" after:

• It has successfully received a data set equal to 128 bytes or a short packet from the host

OR

• A control write status phase has started but there are pending OUT data packets in the buffer.

The CPU writes "1" to CLR_OUT_BUF_RDY after the data set has been unloaded from the FIFO by the CPU (updates the OUT_BUF_RDY flag to "0").

**Special note when using continuous transfer mode in control read request**

In continuous transfer mode, the CPU can write multiple data packets to the data buffer before setting the SET_IN_BUF_RDY to "1". The CPU must write the last data packet separately to the data buffer and sets the SET_DATA_END bit. For example, if the buffer size=128 bytes, MAXP= 8 bytes, and the CPU sends 64 bytes of data to the host, the CPU does the following:

• Writes 7x8=56 bytes to the buffer;

• Sets SET_IN_BUF_RDY=1;

• After the 7 packets are successfully sent to host, the IN_BUF_RDY flag changes from "1" to "0";

• Writes the last 8 bytes of data to the buffer;

• Sets SET_IN_BUF_RDY="1" and SET_DATA_END to "1";

*The CPU should not write all 64 bytes of data, and set the SET_IN_BUF_RDY and SET_DATA_END bits to "1" at the same time.*

**Special note when using continuous transfer mode in control write request**

Because the buffer can hold multiple data packets before generating an interrupt, two special cases should be taken into consideration:

1. The SETUP_END flag usually indicates a premature completion of a control transfer. However, if the data field of a control write is a multiple of MAXP but not a multiple of the buffer size, the SETUP_END flag may be set without causing a premature completion of transfer. For example, if MAXP =8, buffer size = 128, wLength = 192 (a multiple of MAXP but not the buffer size), the following occurs in continuous mode:

After receiving 16 8-byte packets, (128 bytes) from the host, an EP0 interrupt is generated to indicate to the CPU that data unloading can start.

When the host completes sending the remainder of the data field (eight 8-byte packets) an EP0 interrupt is not generated because the buffer is not full and there is no short packet.

When the status phase starts (the host sends an IN token), OUT_BUF_RDY and SETUP_END are set. The SETUP_END is set because the CPU is unaware of the end of the data phase, thus DATA_END is not set. Whenever DATA_END is not set and the status stage starts, the protocol state machine will treat it as a premature completion (data field is less than wLength) and sets the SETUP_END bit.

It is the users responsibility to determine the difference between a premature completion and a normal completion (data field equals the wLength) when the CPU acknowledges a SETUP_END flag in continuous mode.

2. The device usually returns a stall handshake when the host sends more data than specified in the wLength field. However, if a host sends more data than specified in wLength in the middle of a continuous transfer burst, the USB FCU returns ACK to every packet it receives if there are no errors. In this case, when the firmware detects this kind of protocol error, it must set CLR_OUT_PKT_RDY to "1" and set SEND_STALL to "1" so that the USB FCU returns STALL in the subsequent data or status phase. For example, if MAXP = 8, buffer size = 128, wLength = 26, the following may occur:



Downloaded from Arrow.com.

**CASE 1**: The host sends three 8-byte packets and one 2-byte packet. When the core receives the last 2-byte packet, the OUT_BUF_RDY flag is set (because of a short packet) indicating the CPU can unload the data. At the end of unloading, the CPU should clear the OUT_BUF_RDY flag and set the DATA_END. If the host sends more data after this point, the core returns STALL automatically.

**CASE 2:** The host sends 6 8-byte packets (anything greater than 3 for this example) and one 2-byte packet (host may erroneously send 50 bytes instead of 26). The host ACKs each received packet however, it does not automatically return a STALL because the DATA_END flag is not set when the excessive packets are received. When the CPU retrieves the data and detects that the data field is greater than the wLength, it sets the SEND_STALL bit for the core to return STALL.

**EP1-4 IN (Transmit) FIFO Operation**

The CPU writes data to the endpoint's FIFO Data Register. The write pointer automatically increments by 2 in word accessing mode or increments by 1 in byte accessing mode after a write. The CPU must only write data to the FIFO Data Register when the IN_BUF_STS1 flag of the corresponding EPx IN CSR is "0". The IN_BUF_STS0 & IN_BUF_STS1 flags are both "1" after a hardware reset or a USB reset, and become "0" when the corresponding endpoint is first enabled (Endpoints 1-4 IN & OUT are disabled at reset).
The user can program the buffer size and starting location of each IN Endpoint. Users can assign a buffer size up to 1024 bytes in units of 64 bytes to an endpoint. If double buffer mode is selected, the effective buffer size is 2 x buffer size specified.
Continuous transfer mode is available for IN EP1-4 for Bulk Transfers only. When the continuous transfer mode is enabled, it is the user's responsibility to ensure the buffer size is a multiple of the MAXP value.
AUTO_SET function is available for IN EP1-4 for both noncontinuous and continuous modes. When this function is enabled, if a short packet or a less than buffer size data set is to be transmitted to the host, the CPU must write a "1" to the SET_IN_BUF_RDY bit to signify the packet (data set) is ready to send.

**AUTO_SET and continuous transfer mode are disabled:**

**Single Buffer Mode:**

The CPU writes a "1" to the SET_IN_BUF_RDY bit of the corresponding EPx IN CSR after the CPU finishes writing a data packet to the buffer (updates the IN_BUF_STS1 & IN_BUF_STS0 flags from $00_2$ to $11_2$). The USB FCU updates the buffer status flags from $11_2$ to $00_2$ after the data packet has been successfully transmitted to the host.

**Double Buffer Mode:**

The CPU writes "1" to the SET_IN_BUF_RDY bit of the corresponding EPx IN CSR after the CPU finishes writing a data packet to the buffer (updates the IN_BUF_STS1 & IN_BUF_STS0 flags).

- If the buffer is immediately available to accept another data packet, the buffer status flags transition from $00_2$ to $01_2$.
- If the buffer is not available to accept another data packet, the buffer status flags transition from $01_2$ to $11_2$.

The USB FCU updates the buffers status flags after a data packet has been successfully transmitted to the host.

- If the buffer has one more data packet in it, the buffer status flags transition from $11_2$ to $01_2$.
- If the buffer has no more data packet in it, the buffer status flags transition from $01_2$ to $00_2$.

**AUTO_SET is disabled and continuous transfer mode enabled:**

**Single Buffer Mode:**

The CPU writes a "1" to the SET_IN_BUF_RDY bit of the corresponding EPx IN CSR after the CPU finishes writing a data set up to its buffer size to the buffer (updates the IN_BUF_STS1 & IN_BUF_STS0 flags from $00_2$ to $11_2$). The USB FCU sends out data packets equal to the MAXP size one at a time, except for the last packet, if the data set in the buffer is not a multiple of the MAXP, the USB FCU sends a short packet.
The USB FCU updates the buffer status flags from $11_2$ to $00_2$ after the data set has been successfully transmitted to the host.



Downloaded from Arrow.com.

**Double Buffer Mode:**

The CPU writes a "1" to the SET_IN_BUF_RDY bit of the corresponding EPx IN CSR after the CPU finishes writing a data set up to its buffer size to the buffer (updates the IN_BUF_STS1 & IN_BUF_STS0 flags). The USB FCU sends out data packets equal to the MAXP size one at a time, except for the last packet if the data in the buffer is not a multiple of the MAXP, the USB FCU sends a short packet.

• If the buffer is immediately available to accept another data set, the buffer status flags transition from $00_2$ to $01_2$.

• If the buffer is not available to accept another data set, the buffer status flags transition from $01_2$ to $11_2$.

The USB FCU updates the buffers status flags after a data set has been successfully transmitted to the host.

• If the buffer has one more data set in it, the buffer status flags transition from $11_2$ to $01_2$.

• If the buffer has no more data set in it, the buffer status flags transition from $01_2$ to $00_2$.

**AUTO_SET is enabled and continuous transfer mode disabled:**

**Single Buffer Mode:**

After the CPU writes a data packet equal to the MAXP size to the buffer, the USB FCU updates the corresponding EPx IN CSR's IN_BUF_STS1 & IN_BUF_STS0 flags from $00_2$ to $11_2$ automatically without the CPU writing "1" to the SET_IN_BUF_RDY bit. The USB FCU updates the buffer status flags from $11_2$ to $00_2$ after the data packet has been successfully transmitted to the host.
If the data packet is less than the MAXP size, the CPU must write "1" to the SET_IN_BUF_RDY bit to signify the data packet is ready to send.

**Double Buffer Mode:**

After the CPU writes a data packet equal to its MAXP size to the buffer, the USB FCU updates the corresponding EPx IN CSR's IN_BUF_STS1 & IN_BUF_STS0 flags.

• If the buffer is immediately available to accept another data packet, the buffer status flags transition from $00_2$ to $01_2$.

• If the buffer is not available to accept another data packet, the buffer status flags transition from $01_2$ to $11_2$.

The USB FCU updates the buffers status flags after a data packet has been successfully transmitted to the host.

• If the buffer has one more data packet in it, the buffer status flags transition from $11_2$ to $01_2$.

• If the buffer has no more data packet in it, the buffer status flags transition from $01_2$ to $00_2$.

• If the data packet is less than the MAXP size, the CPU must write "1" to the SET_IN_BUF_RDY bit to signify the data packet is ready to send.

**AUTO_SET and continuous transfer mode are enabled:**

**Single Buffer Mode:**

After the CPU writes a data set equal to the buffer size to the buffer, the USB FCU updates the corresponding EPx IN CSR's IN_BUF_STS1 & IN_BUF_STS0 flags from $00_2$ to $11_2$ automatically without the CPU writing "1" to the SET_IN_BUF_RDY bit. The USB FCU sends out data packets equal to the MAXP size one at a time, except for the last packet if the data set in the buffer is not a multiple of its MAXP, the USB FCU sends a short packet. The USB FCU updates the buffer status flags from $11_2$ to $00_2$ after the data set has been successfully transmitted to the host. If the data set is less than the buffer size, the CPU must write "1" to the SET_IN_BUF_RDY bit to signify the data set is ready to send.



Downloaded from Arrow.com.

**Double Buffer Mode:**

After the CPU writes a data set equal to its buffer size to the buffer, the USB FCU updates the IN_BUF_STS1 & IN_BUF_STS0 flags.

• If the buffer is immediately available to accept another data set, the buffer status flags transition from $00_2$ to $01_2$.

• If the buffer is not available to accept another data set, the buffer status flags transition from $01_2$ to $11_2$.

The USB FCU sends out data packets equal to the MAXP size one at a time, except for the last packet.

• If the data in the buffer is not a multiple of the MAXP, the USB FCU sends a short packet.

The USB FCU updates the buffers status flags after a data set has been successfully transmitted to the host.

• If the buffer has one more data set in it, the buffer status flags transition from $11_2$ to $01_2$.

• If the buffer has no more data set in it, the buffer status flags transition from $01_2$ to $00_2$.

• If the data set is less than the buffer size, the CPU must write "1" to the SET_IN_BUF_RDY bit to signify the data set is ready to send.

**IN Endpoint FIFO Flush**

A software or a hardware flush causes the USB FCU to act (in both continuous and noncontinuous transfer modes) as if a data set has been successfully transmitted out to the host. When there is one data set in the buffer, a flush causes the buffer to be empty. When there are two data sets in the buffer, a flush causes the older data set to be flushed out from the buffer. A flush also updates the buffer status flags of the corresponding EPx IN CSR.
The Endpoint 1-4 IN buffer status can be obtained from the two status bits of the EPx IN CSR of the corresponding endpoint as shown in Table 1.35.

**Table 1.35. Endpoint 1-4 IN buffer status**

| IN_BUF_STS1 | IN_BUF_STS0 | Buffer Status | |
|---|---|---|---|
| 0 | 0 | No data set in the IN buffer | |
| 0 | 1 | Single buffer mode: | N/A |
| | | Double buffer mode: | One data set in the IN Buffer |
| 1 | 0 | Single buffer mode: | N/A |
| | | Double buffer mode: | N/A |
| 1 | 1 | Single buffer mode: | One data set in the IN buffer |
| | | Double buffer mode: | Two data sets in the IN buffer |



Downloaded from Arrow.com.

**EP1-4 OUT (Receive) FIFOs**

The CPU reads data from the endpoint's FIFO Data Register. The read pointer automatically increments by 2 in word accessing mode or by 1 in byte accessing mode after a read. The CPU must only read data from the FIFO Data Register when the OUT_BUF_STS1 flag of the corresponding EPx OUT CSR is a "1".
The user can program each OUT endpoint's buffer size and starting location, and assign a buffer size up to 1024 bytes in units of 64 bytes to an endpoint. If double buffer mode is selected, the effective buffer size is 2 x buffer size specified.
Continuous transfer mode is available for OUT EP1-4 bulk transfers only. When the continuous transfer mode is enabled, the user is responsible for ensuring that the buffer size is a multiple of the MAXP value. Also, the user must ensure that the last data set from the host either contains a short packet or is equal to the buffer size, otherwise there is no interrupt or status that will signify that the last data set was received.
AUTO_CLR function is available for OUT EP1-4.

**AUTO_CLR and continuous transfer mode are disabled:**

**Single Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags from $00_2$ to $11_2$ after it has successfully received a data packet from the host.
The CPU writes "1" to the CLR_OUT_BUF_RDY bit after the data packet has been unloaded from the buffer by the CPU (updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags from $11_2$ to $00_2$).

**Double Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags after it has successfully received a data packet from the host.

- If the buffer has only one data packet, the buffer status flags transition from $00_2$ to $10_2$.
- If the buffer has two data packets, the buffer status flags transition from $10_2$ to $11_2$.

The CPU writes "1" to the CLR_OUT_BUF_RDY bit after a data packet has been unloaded from the buffer by the CPU (updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags).

- If the buffer has one more data packet in it, the buffer status flags transition from $11_2$ to $10_2$.
- If the buffer has no more data packet in it, the buffer status flags transition from $10_2$ to $00_2$.

**AUTO_CLR is disabled and continuous transfer mode enabled:**

**Single Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags from $00_2$ to $11_2$ after it has successfully received from the host a data set equal to the buffer size, or a short packet.
The CPU writes "1" to the CLR_OUT_BUF_RDY bit after the data set has been unloaded from the buffer by the CPU (updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags from $11_2$ to $00_2$ ).

**Double Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags after it has successfully received a data set equal to its buffer size or a short packet from the host..

- If the buffer has only one data set, the buffer status flags transition from $00_2$ to $10_2$ .
- If the buffer has two data sets, the buffer status flags transition from $10_2$ to $11_2$ .



Downloaded from Arrow.com.

The CPU writes a "1" to the CLR_OUT_BUF_RDY bit after a data set has been unloaded from the buffer by the CPU (updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags).

• If the buffer has one more data set in it, the buffer status flags transition from $11_2$ to $10_2$ .

• If the buffer has no more data set in it, the buffer status flags transition from $10_2$ to $00_2$ .

**AUTO_CLR is enabled and continuous transfer mode disabled:**

**Single Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags from $00_2$ to $11_2$ after it has successfully received a data packet from the host.
The USB FCU updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags from $11_2$ to $00_2$ automatically when the data packet has been unloaded from the buffer by the CPU without the CPU writing a "1" to the CLR_OUT_BUF_RDY bit.

**Double Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags after it has successfully received a data packet from the host.

• If the buffer has only one data packet, the buffer status flags transition from $00_2$ to $10_2$ .

• If the buffer has two data packets, the buffer status flags transition from $10_2$ to $11_2$ .

The USB FCU updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags automatically when a data packet has been unloaded from the buffer by the CPU without the CPU writing a "1" to the CLR_OUT_BUF_RDY bit.

• If the buffer has one more data packet in it, the buffer status flags transition from $11_2$ to $10_2$ .

**AUTO_CLR is enable and continuous transfer mode enabled:**

**Single Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags from $00_2$ to $11_2$ after it has successfully received a data set equal to its buffer size or a short packet from the host.
The USB FCU updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags from $11_2$ to $00_2$ automatically when the data set has been unloaded from the buffer by the CPU without the CPU writing a "1" to the CLR_OUT_BUF_RDY bit.

**Double Buffer Mode:**

The USB FCU updates the corresponding EPx OUT CSR's OUT_BUF_STS1 & OUT_BUF_STS0 flags after it has successfully received a data set equal to its buffer size or a short packet from the host.

• If the buffer has only one data set, the buffer status flags transition from $00_2$ to $10_2$ .

• If the buffer has two data sets, the buffer status flags transition from $10_2$ to $11_2$ .

The USB FCU updates the OUT_BUF_STS1 & OUT_BUF_STS0 flags automatically when a data set has been unloaded from the buffer by the CPU without the CPU writing a "1" to the CLR_OUT_BUF_RDY bit.

• If the buffer has one more data set in it, the buffer status flags transition from $11_2$ to $10_2$.

• If the buffer has no more data set in it, the buffer status flags transition from $10_2$ to $00_2$.



Downloaded from Arrow.com.

### OUT Endpoint FIFO Flush

A software flush causes the USB FCU to act as if a data set has been unloaded from the buffer. The user must only set the flush bit when OUT_BUF_STS1 = 1, which indicates that one or two data sets have been received. When there is one data set in the buffer, a flush causes the buffer to empty. When there are two data sets in the buffer, a flush causes the older data set to be flushed out from the buffer. A flush also updates the buffer status flags of the corresponding EPx OUT CSR.

The status of Endpoint 1-4 OUT buffers can be obtained from the two status bits of the EPx OUT CSR of the corresponding endpoint as shown in Table 1.36.

**Table 1.36. Endpoints 1-4 OUT buffer status**

| OUT_BUF_STS1 | OUT_BUF_STS0 | Buffer Status | |
|---|---|---|---|
| 0 | 0 | No data set in the OUT buffer | |
| 0 | 1 | Single buffer mode: | N/A |
| | | Double buffer mode: | N/A |
| 1 | 0 | Single buffer mode: | N/A |
| | | Double buffer mode: | One data set in the OUT buffer |
| 1 | 1 | Single buffer mode: | One data set in the OUT buffer |
| | | Double buffer mode: | Two data sets in the OUT buffer |

### Interrupt Endpoints

Any endpoint can be used for interrupt transfers. For normal interrupt transfers, the interrupt transactions behave the same as bulk transactions, i.e., no special setting is required.
The IN endpoints may be used to communicate rate feedback information for certain types of isochronous functions. Setting the INTPT bit in the IN CSR register of the corresponding IN CSR enables this function. When the INTPT bit is set, the data toggle bit changes after each packet is sent regardless of the presence or type of handshake that is returned from the host.

The operation sequence for an IN endpoint used to communicate rate feedback information is listed in the following steps.

1. Set single buffer mode for the endpoint in use;

2. Set INTPT bit of the IN CSR;

3. Load interrupt status information and set SET_IN_BUF_RDY bit in the IN CSR;

4. Repeat step 3 for all subsequent interrupt status updates.

When an interrupt endpoint is used for rate feedback, the device always has data to send back to the host, even if the data conveys that everything is ‘fine’. Therefore, the device never NAKs an IN token from the host. The device always sends out the data in the FIFO in response to an IN token regardless of the IN buffer status bits.



Downloaded from Arrow.com.

### USB Special Function Registers

The MCU controls USB operation through the use of special function registers. Some USB-related special function registers have a mix of read/write, read only, and write only register bits. Additionally, the bits may be configured to allow the user to write only "0" or "1" to individual bits.

• When accessing these registers, writing "0" to a register that can only be set to "1" by the CPU has no effect on that register bit.

• Writing "1" to a register that can only be set to "0" by the CPU has no effect on that register bit.

All USB SFRs, with the exceptions of Endpoint FIFO data registers, USBAD, and USBC can be accessed by word or by byte at an even or odd address. Endpoint FIFO Data Registers can be accessed by either word or by byte at even addresses only.
The contents of all USB Special Functions Registers, including USB Attach/Detach and USB Control, are preserved after a software reset.

### USB Attach/Detach Register

The USB Attach / Detach Register is shown in Figure 1.44. The register is used to attach and detach the USB function from a USB host without physically disconnecting the USB cable. This functionality is enabled by setting P9$_0$_SECOND to a "1". Doing this forces P9$_0$ to operate as a pull-up for D+ (through an external 1.5k ohm resistor). The port driver is tri-stated and a "1" is always read from the port bit in this mode. When the ATTACH/DETACH bit is a "1" (and P9$_0$_SECOND is a "1"), P9$_0$ is driven with the voltage on UVcc, causing D+ to be pulled up and the host to detect an attach. When the ATTACH/DETACH bit is a "0" (and P9$_0$_SECOND is a "1"), P9$_0$ is tri-stated, causing D+ to be pulled down (through the cable and 15k ohm resistor on the host/hub side) and a detach to be registered by the host. A 1.5k ohm pull-up resistor must be connected externally from P9$_0$ to D+ when this functionality is used. When it is not used, the 1.5k ohm resistor should be placed between UVcc and D+.
See "Vbus Detect" for information on the vbus detect enable bit.

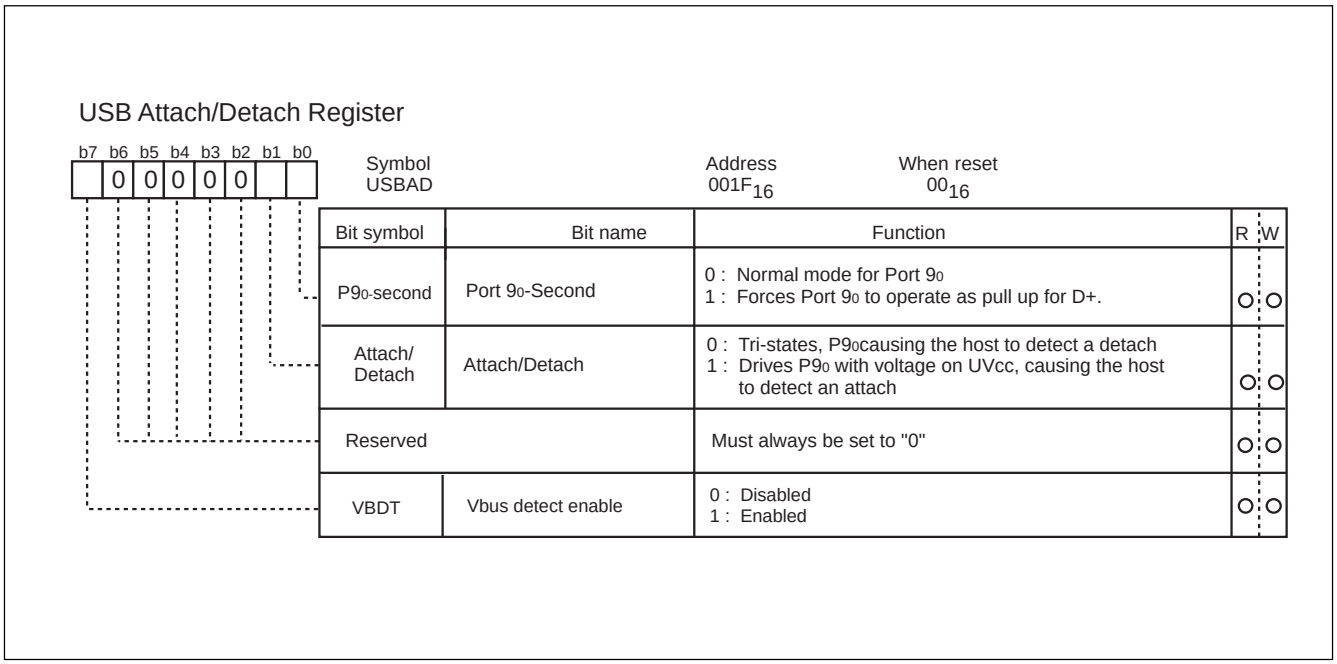



**Figure 1.44. USB Attach/Detach register (USBAD)**



Downloaded from Arrow.com.

### USB Control Register

The USB Control Register, shown in Figure 1.45, is used to control the USB FCU. This register is not reset by USB reset signaling. After the USB is enabled (USBC7 set to "1"), a minimum delay of 250ns (three 12 MHz clock periods) is needed before performing any other USB register read/write operations.

**• USBC5 (USB Clock Enable):**

The USB clock enable bit is used to enable or disable the USB clock (fUSB). This clock is derived from the Frequency Synthesizer and is required for USB operation.

**• USBC6 (SOF port select):**

The SOF port select bit enables or disables outputting a SOF signal on the P92/SOF pin. When this bit is set to "1", an active low pulse is output each time a start of frame packet is detected on the USB. The output pulse width is 166ns (two 12MHz USB clock cycles).

**• USBC7 (USB Enable):**

The USB enable bit is used to enable or disable the USB block. Make sure the USB clock is enabled before setting this bit to "1".

USB Control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | 0 | 0 | 0 |

Symbol: USBC | Address: $000C_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| Reserved | | Must always be set to "0" | O | O |
| USBC5 | USB clock enable bit | 0 : Disable<br>1 : Enable | O | O |
| USBC6 | USB SOF port select bit | 0 : Disable (Note 1)<br>1 : Enable | O | O |
| USBC7 | USB enable bit | 0 : Disable (Note 2)<br>1 : Enable | O | O |

Note 1: $P9_2$ is used as GPI/O pin.
Note 2: All USB internal registers are held at their default values.

**Figure 1.45. USB Control register (USBC)**

### USB Function Address Register

The USB Function Address Register, shown in Figure 1.46, maintains the 7-bit USB address assigned by the host. The USB FCU uses this register value to decode USB token packet addresses. At reset, when the device is not yet configured, the value is 0016. (For the procedures on how to update this register, refer to Application Notes USB Consecutive Set Address)

USB Function Address register

| (b15) b7 | | | | | | | b0 | (b8) b7 | | | | | | | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | |

Symbol: USBA | Address: $0280_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| FUNAD6-0 | Function address | 7-bit programmable function address | O | O |
| Reserved | | Must always be "0" | O | O |

**Figure 1.46. USB Function Address register (USBA)**



Downloaded from Arrow.com.

### Power Management Register

The USB Power Management Register, shown in Figure 1.47, is used for power management in the USB FCU.

**SUSPEND State Flag:**

When the USB FCU does not detect any bus activity on D+/D- (in the J-state) for at least 3ms, it updates the Suspend State Flag and generates an interrupt. This flag is cleared when active signaling from the host is detected on D+/D- (The USB FCU generates a resume interrupt), or the CPU sets the Remote Wake-up Bit while in suspend state and it is subsequently cleared by the CPU. If the USB clock was disabled during the suspend state, the SUSPEND state flag is not cleared until after the USB clock is re-enabled.

**WAKEUP Control Bit:**

The CPU writes a "1" to the WAKEUP Control Bit for remote wake-up. While this bit is set and the USB FCU is in suspend mode, resume signaling is sent to the host. The CPU must keep this bit set for a minimum of 1ms and a maximum of 15ms before writing a "0" to this bit.

USB Power Management register

(b15) b7 (b8) b0 b7 b0

0 0 0 0 0 0 0 0 0 0 0 0 0 0

Symbol: USBPM | Address: $0282_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| SUSPEND | Suspend state flag | 0 : Not in suspend state<br>1 : In suspend state | O | O Note |
| WAKEUP | Remote wakeup | 0 : End remote wakeup signal<br>1 : Remote wakeup signaling if SUSPEND="1" | O | O |
| Reserved | | Must always be "0" | O | O |

Note: Read only

**Figure 1.47. USB Power Management register (USBPM)**

### USB Function Interrupt Status Register

USB Function Interrupt Status register, shown in Figure 1.48, is used to indicate the condition that caused a USB function interrupt to the CPU. A "1" indicates the corresponding condition caused an interrupt.

**INTST0, INITST2, INTST4 or INTST6** is set to "1" by the USB FCU when:

• The endpoint is enabled from a disabled state;

• A data set is successfully sent;

• A hardware autoflush takes place or the CPU writes "1" to INxCSR6 (FLUSH) if there are one or two data sets in the buffer. This causes the EP1-4 IN buffer status flag to change states.

**INTST1, INTST3, INTST5 or INTST7** is set to "1" by the USB FCU when:

• A data set is successfully received.

**INTST8** is an Error Interrupt Status flag, which indicates that an error has been encountered at any endpoint. This flag is set to "1" by the USB FCU when:

• EP0CSR4 (FORCE_STALL) flag is set;

• EP0CSR5 (SETUP_END) flag is set;

• INxCSR2 (UNDER_RUN) flag is set on any EP1-4 IN endpoint;

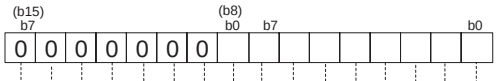
• OUTxCSR2 (OVER_RUN) flag is set on any EP1-4 OUT endpoint;

• OUTxCSR3 (FORCE_STALL) flag is set on any EP1-4OUT endpoint;



Downloaded from Arrow.com.

USB Interrupt Status register




| Symbol | Address | When reset |
|---|---|---|
| USBIS | $0284_{16}$ | $0000_{16}$ |

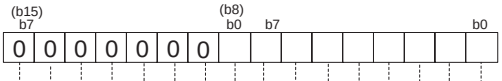
| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| INTST0 | EP1 IN interrupt status flag | 0 : No interrupt request<br>1 : Interrupt request issued | O | X |
| INTST1 | EP1 OUT interrupt status flag | | | |
| INTST2 | EP2 IN interrupt status flag | | | |
| INTST3 | EP2 OUT interrupt status flag | | | |
| INTST4 | EP3 IN interrupt status flag | | | |
| INTST5 | EP3 OUT interrupt status flag | | | |
| INTST6 | EP4 IN interrupt status flag | | | |
| INTST7 | EP4 OUT interrupt status flag | | | |
| INTST8 | Error interrupt status flag | | | |
| Reserved | | Must always be "0" | O | X |

**Figure 1.48. USB Interrupt Status register (USBIS)**

### USB Function Interrupt Clear Register

The USB Function Interrupt Clear register, shown in Figure 1.49, is used by the CPU to clear the USB Function Interrupt Status bits. The CPU writes a "1" to clear a corresponding USB Function Interrupt Status flag.

USB Interrupt Clear register




| Symbol | Address | When reset |
|---|---|---|
| USBIC | $0286_{16}$ | $0000_{16}$ |

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| INTCL0 | Clear EP1 IN interrupt status flag | 0 : No action<br>1 : Clear interrupt status flag | X<br>Note | O |
| INTCL1 | Clear EP1 OUT interrupt status flag | | | |
| INTCL2 | Clear EP2 IN interrupt status flag | | | |
| INTCL3 | Clear EP2 OUT interrupt status flag | | | |
| INTCL4 | Clear EP3 IN interrupt status flag | | | |
| INTCL5 | Clear EP3 OUT interrupt status flag | | | |
| INTCL6 | Clear EP4 IN interrupt status flag | | | |
| INTCL7 | Clear EP4 OUT interrupt status flag | | | |
| INTCL8 | Clear error interrupt status flag | | | |
| Reserved | | Must always be "0" | X | O |

Note: Always read "0"

**Figure 1.49. USB Interrupt Clear register (USBIC)**

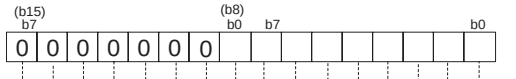


Downloaded from Arrow.com.

### USB Function Interrupt Enable Register

The USB Function Interrupt Enable register, shown in Figure 1.50 is used to enable the corresponding interrupt status conditions that can generate a USB Function interrupt. When the bit of a corresponding interrupt condition is "0", it does not generate a USB function interrupt. When the bit is a "1", it can generate a USB Function interrupt.

USB Interrupt Enable register




Symbol: USBIE | Address: $0288_{16}$ | When reset: $01FF_{16}$

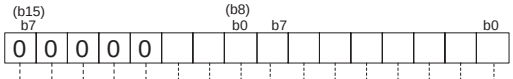
| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| INTEN0 | EP1 IN interrupt enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| INTEN1 | EP1 OUT interrupt enable bit | | | |
| INTEN2 | EP2 IN interrupt enable bit | | | |
| INTEN3 | EP2 OUT interrupt enable bit | | | |
| INTEN4 | EP3 IN interrupt enable bit | | | |
| INTEN5 | EP3 OUT interrupt enable bit | | | |
| INTEN6 | EP4 IN interrupt enable bit | | | |
| INTEN7 | EP4 OUT interrupt enable bit | | | |
| INTEN8 | Error interrupt enable bit | | | |
| Reserved | | Must always be "0" | O | O |

**Figure 1.50. USB Interrupt Enable (USBIE)**

### USB Frame Number Register

The USB Frame Number Register, shown in Figure 1.51, contains the 11-bit frame number received from the host.

USB Frame Number register




Symbol: USBFN | Address: $028A_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| FN10-0 | Frame number bit 0-10 | 11-bit frame number issued with an SOF packet | O | X |
| Reserved | | Must always be "0" | O | X |

**Figure 1.51. USB Frame Number register (USBFN)**



Downloaded from Arrow.com.

### USB ISO Control Register

The USB ISO Control Register, shown in Figure 1.52, contains the isochronous data transfer control and status information.

• ISO_UPD

The ISO_UPD bit is a global bit for endpoints 1-4 and works with IN isochronous pipes only.

When ISO_UPD = "0", a data packet in an endpoints IN buffer is always 'ready to transmit' when it receives the next IN token from the host (with matched address and endpoint number), if the CPU writes "1" to the corresponding endpoint's SET_IN_BUF_RDY bit, or in AUTO_SET case, a data packet equal to EPx's MAXP value has been written to the FIFO. When ISO_UPD = "1" and the ISO bit of the corresponding endpoint's IN CSR is set, the internal 'ready to transmit' signal to the transmit control logic is not activated when the CPU writes "1" to the corresponding endpoint's SET_IN_BUF_RDY bit, or in the AUTO_SET case, a data packet equal to EPx's MAXP value has been written to the FIFO. Instead it is activated when the next SOF is received, thus the data loaded in frame n is transmitted out in frame n+1.

**•AUTO_FL**

When AUTO_FL = "1", ISO_UPD = "1", IN endpoint's ISO bit is set, and the IN endpoint's IN_BUF_STS1 & IN_BUF_STS0 are "1"s at the time the USB FCU detects a SOF (from the host or from artificial SOF), it automatically flushes the oldest packet from the IN buffer. In this case, IN_BUF_STS1 & IN_BUF_STS0 are "1"s and indicate that two data packets are in the IN buffer. Double buffering is required for ISO transfer.

**•ART_SOF_ENA**

An artificial SOF function enable bit.

**•ART_SOF_SET**

Artificial SOF function status flag. When this flag is "1", it indicates that an artificial SOF will be generated by the device because of a missing or corrupt SOF packet (when the SOF enable bit is set to "1"). A corrupt SOF packet is any SOF having an error in its 8-bit Pakcet ID (PID) field.

**•CLR_ART_SOF**

The CPU writes "1" to this bit to clear the ART_SOF_SET flag.

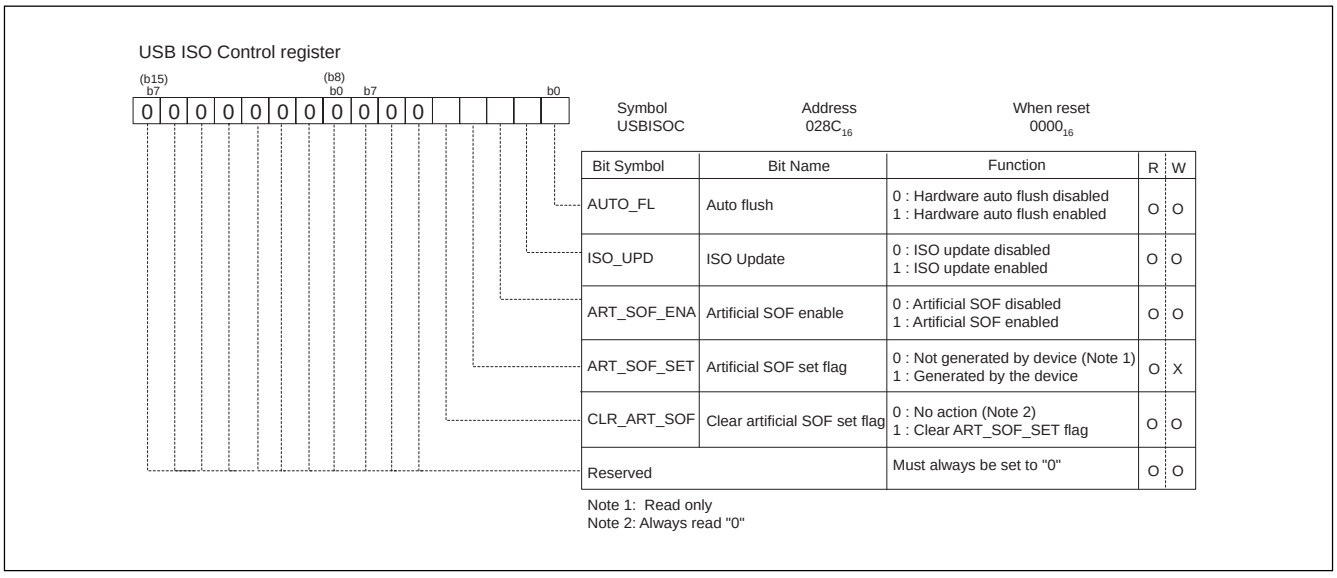



**Figure 1.52. USB ISO Control register (USBISOC)**



Downloaded from Arrow.com.

### USB Endpoint Enable Register

The USB Endpoint Enable Register, shown in Figure 1.53, is used to enable/disable an individual endpoint. EP0 is always enabled and cannot be disabled by firmware. All endpoints are disabled after reset.

USB Endpoint Enable register

(b15) b7 | (b8) b0 | b7 | b0

0 0 0 0 0 0 0 0

Symbol: USBEPEN | Address: $028E_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| EP1_OUT | EP1 OUT enable | 0 : Disabled<br>1 : Enabled | O | O |
| EP1_IN | EP1 IN enable | | | |
| EP2_OUT | EP2 OUT enable | | | |
| EP2_IN | EP2 IN enable | | | |
| EP3_OUT | EP3 OUT enable | | | |
| EP3_IN | EP3 IN enable | | | |
| EP4_OUT | EP4 OUT enable | | | |
| EP4_IN | EP4 IN enable | | | |
| Reserved | | Must always be "0" | O | O |

**Figure 1.53. USB Endpoint Enable register (USBEPEN)**

### USB DMAx Request Register (x = 0 to 3)

The USB DMAx Request Register, shown in Figure 1.54, selects which USB EPx FIFO read/write requests are used as the DMAC channels 0 to 3 request source. Each USB DMAx Request Registers should have only one bit set at any given time. When multiple bits are set, no request is selected.

USB DMAx Request registers

(b15) b7 | (b8) b0 | b7 | b0

0 0 0 0 0 0

Symbol: USBDMAx (x=0 to 3) | Address: $0290_{16}$, $0292_{16}$, $0294_{16}$, $0296_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DMAxRO | EP0 IN FIFO write request select bit | 0 : Not selected<br>1 : Selected | O | O |
| DMAxR1 | EP1 IN FIFO write request select bit | | | |
| DMAxR2 | EP2 IN FIFO write request select bit | | | |
| DMAxR3 | EP3 IN FIFO write request select bit | | | |
| DMAxR4 | EP4 IN FIFO write request select bit | | | |
| DMAxR5 | EP0 OUT FIFO read request select bit | | | |
| DMAxR6 | EP1 OUT FIFO read request select bit | | | |
| DMAxR7 | EP2 OUT FIFO read request select bit | | | |
| DMAxR8 | EP3 OUT FIFO read request select bit | | | |
| DMAxR9 | EP4 OUT FIFO read request select bit | | | |
| Reserved | | Must always be "0" | O | O |

**Figure 1.54. USB DMAx Request register (x=0 to 3)**



Downloaded from Arrow.com.

### USB Endpoint 0 CSR

The Endpoint 0 CSR (Control & Status register), shown in Figure 1.55, contains the control and status information for EP0.

**•EP0CSR0(OUT_BUF_RDY):**

A status flag, "1" indicates a SETUP packet or an OUT data set is in the OUT buffer, ready for the CPU to unload.

During the data phase, if noncontinuous mode is set, the OUT_BUF_RDY bit is "1" when:

• A data packet is received from the host

During the data phase, if continuous mode is set, the OUT_BUF_RDY bit is "1" when:

• A data set equal to 128 bytes is received from the host

• A short packet is received from the host

• A control write status phase has started with pending OUT data packets in the buffer.

**•EP0CSR1(IN_BUF_RDY):**

A status flag, "1" indicates a data set is in the IN buffer, ready for transmission. The USB FCU clears this bit after the data set is successfully transmitted to the host, or the EP0CSR5 (SETUP_END) bit is set.

**• EP0CSR2 (SETUP):**

A status flag, "1" indicates a SETUP packet has been received. The SETUP Flag is a subset of the OUT_BUF_RDY flag.

**•EP0CSR3(DATA_END):**

A status flag, "1" indicates the CPU sets the DATA_END bit. The USB FCU clears this flag after the status phase has started or a new SETUP is received. This flag is a maskable flag. If DATA_END Flag Mask is a "1" (default), this DATA_END flag is always a "0" and no EP0 interrupt is caused by the DATA_END flag being cleared.

**•EP0CSR4(FORCE_STALL):**

A status flag, "1" indicates a protocol error when one of the following occurs:

• Host sends an IN token in the absence of a SETUP stage

• Host sends a bad data toggle in the STATUS stage, (i.e. DATA0 is used)

• Host sends a bad data toggle in the SETUP stage, (i.e. DATA1 is used)

• Host requests more data than specified in the SETUP state, (i.e. IN token comes after DATA_END bit is set)

• Host sends more data than specified in the SETUP state, (i.e. OUT token comes after DATA_END bit is set)

• Host sends a larger data packet than the MAXP size

All of the conditions stated (except bad data toggle in the SETUP stage) cause the device to send a STALL handshake for the current IN/OUT transaction. For the bad data toggle in the SETUP stage, the device sends ACK for the SETUP stage and then sends STALL for the next IN/OUT transaction. A STALL handshake caused by the above conditions lasts for one transaction and terminates the ongoing control transfer. Any packet after the STALL handshake will be seen as the beginning of a new control transfer.



Downloaded from Arrow.com.

**•EP0CSR5(SETUP_END):**

A status flag, "1" indicates a premature completion of a control transfer when one of the following events occurs:

• A control transfer ends before the specific length of data is transferred during the data phase (status phase starts before DATA_END bit is set)

• A new SETUP is received before successfully completing the status phase of the previous control transfer.

**•EP0CSR6(CLR_OUT_BUF_RDY):**

The CPU writes a "1" to this bit after unloading a data set from the buffer. Writing a "1" to this bit clears the OUT_BUF_RDY status flag.

**•EP0CSR7(SET_IN_BUF_RDY):**

The CPU writes a "1" to this bit after loading a data set to the buffer. Writing a "1" to this bit sets the IN_BUF_RDY status flag.

**•EP0CSR8(CLR_SETUP):**

The CPU writes a "1" to this bit to clear the SETUP status flag.

**•EP0CSR9(SET_DATA_END):**

The CPU writes a "1" to this bit when it writes (IN data phase) the last data packet to the buffer or reads (OUT data phase) the last data packet from the buffer. The CPU sets this bit at the same time (using the same instruction) as it sets the CLR_OUT_BUF_RDY bit or sets the SET_IN_BUF_RDY bit for the last data set. Writing a "1" to this bit sets the DATA_END status flag.

**•EP0CSR10(CLR_FORCE_STALL):**

The CPU writes a "1" to this bit to clear the FORCE_STALL status flag.

**•EP0CSR11(CLR_SETUP_END):**

The CPU writes a "1" to this bit to clear the SETUP_END status flag.

**•EP0CSR12(SEND_STALL):**

The CPU writes a "1" to this bit when it decodes an invalid or unsupported request from the host. The CPU should only write a "1" to this bit at the same time it writes a "1" to EP0CSR6 (CLR_OUT_BUF_RDY). When this bit is a "1", the USB FCU returns STALL handshakes for all subsequent IN/OUT transactions. The CPU writes a "0" to clear it after it receives a new SETUP packet. It is up to the firmware to decide what SETUP packet should lead the clearing of the SEND_STALL bit.

**•EP0CSR13(DATA_END_MASK):**

This bit is for the CPU to mask or unmask the clearing of DATA_END as an EP0 interrupt source - default is masked (clearing of DATA_END does not cause an EP0 interrupt).



Downloaded from Arrow.com.

USB Endpoint x OUT Control and Status register




Symbol: EPxOCS (x = 1 - 4)

Address: $02B6_{16}$, $02BE_{16}$, $02C6_{16}$, $02CE_{16}$

When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| OUTxCSR0 | OUT_BUF_STS0 flag | These two bits indicate the EPx OUT buffer status:<br>Bit1 Bit0<br>0 0 : No data set in the OUT buffer<br>0 1 : Single buffer mode: N/A<br>Double buffer mode: N/A<br>1 0 : Single buffer mode: N/A<br>Double buffer mode: one data set in the OUT buffer<br>1 1 : Single buffer mode: one data set in the OUT buffer<br>Double buffer mode: two data sets in the OUT buffer | O | X |
| OUTxCSR1 | OUT_BUF_STS1 flag | | O | X |
| OUTxCSR2 | OVER-RUN flag | 0 : No over run detected<br>1 : Over run detected | O | X |
| OUTxCSR3 | FORCE_STALL flag | 0 : No packet size larger than MAXP violation detected<br>1 : Packet size larger than MAXP violation detected | O | X |
| OUTxCSR4 | DATA_ERR flag | 0 : No data error detected<br>1 : Data error detected | O | X |
| OUTxCSR5 | CLR_OUT_BUF_RDY | 0 : No action<br>1 : Data set unloaded from the OUT buffer (updates status flags) | O | O Note |
| OUTxCSR6 | CLR_OVER_RUN | 0 : No action<br>1 : Clears OVER_RUN flag | O | O Note |
| OUTxCSR7 | CLR_FORCE_STALL | 0 : No action<br>1 : Clears FORCE_STALL flag | O | O Note |
| OUTxCSR8 | CLR_DATA_ERR | 0 : No action<br>1 : Clears DATA_ERR flag | O | O Note |
| OUTxCSR9 | TOGGLE_INIT | 0 : No action<br>1 : Initialize the next data PID as a DATA0 for reception | O | O Note |
| OUTxCSR10 | FLUSH | 0 : No action<br>1 : Flush out one data set | O | O Note |
| OUTxCSR11 | ISO | 0 : Select non-isochronous endpoint<br>1 : Select isochronous endpoint | O | O |
| OUTxCSR12 | SEND_STALL | 0 : No STALL by CPU<br>1 : STALL by CPU | O | O |
| OUTxCSR13 | AUTO_CLR | 0 : AUTO_CLR disabled<br>1 : AUTO_CLR enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |

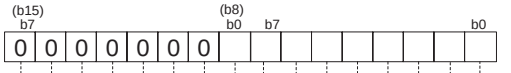
Note: Always read a "0" when writing to this bit

**Figure 1.55. USB Endpoint 0 Control and Status register (EP0CS)**

### USB Endpoint 0 MAXP Register

The USB Endpoint 0 MAXP Register, shown in Figure 1.56, indicates the maximum packet size (MAXP) of an EP0 IN/OUT packet. The default value for EP0 MAXP is 8 bytes. It also contains the enable bits for Control write continuous transfer and control read continuous transfer.

USB Endpoint 0 MAXP register




Symbol: EP0MP

Address: $029A_{16}$

When reset: $0008_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| EP0MP6-0 | Maximum packet size | | O | O |
| WRT_CONT | Control Write continuous transfer mode | 0 : Disabled<br>1 : Enabled | O | O |
| RD_CONT | Control Read continuous transfer mode | 0 : Disabled<br>1 : Enabled | O | O |
| Reserved | | Must always be "0" | O | O |

**Figure 1.56. USB Endpoint 0 MAXP register (EP0MP)**

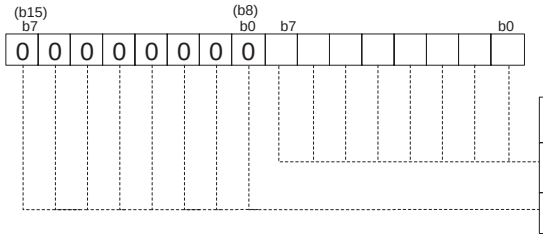


Downloaded from Arrow.com.

### USB Endpoint 0 WRT CNT Register

The USB Endpoint 0 WRT CNT Register, shown in Figure 1.57, contains the number of bytes of the current data set in the OUT buffer. The USB FCU sets the value in the WRT_CNT Register after having successfully received a data set from the host. The CPU reads the register to determine the number of bytes to be read from the buffer. The WRT_CNT value does not decrement upon a CPU read from the FIFO Data Register. The WRT_CNT value is cleared when the CPU writes a "1" to the CLR_OUT_BUF_RDY bit of the EP0 CSR.

USB Endpoint 0 Write Count register




Symbol: EP0WC | Address: $029C_{16}$ | When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| EP0WC7-0 | Receive byte count | | O | X |
| Reserved | | Must always be "0" | O | O |

**Figure 1.57. USB Endpoint 0 write count register (EP0WC)**

### USB Endpoint x IN CSR (x = 1 to 4)

The USB Endpoint x IN control status register, shown in Figure 1.58, contains control and status information of the respective IN EP 1-4.

**• INxCSR0 (IN_BUF_STS0) and INxCSR1 (IN_BUF_STS1):**

Two status flags, indicate the current status of the IN buffer. These two flags are "1"s after reset, and become "0"s when the respective endpoint is enabled from a disabled state. The buffer status flags get updated when one of the following events occurs:

1. The USB FCU successfully sends out a data set to the host.
2. The CPU loads a data set to the buffer (writes a "1" to SET_IN_BUF_RDY).
3. The CPU writes a "1" to the FLUSH bit or a hardware auto flush takes place.

**•INxCSR2(UNDER_RUN):**

A status flag, "1" indicates an under run has occurred in an isochronous data transfer. The USB FCU updates this flag to a "1" at the beginning of an IN token if no data packet is in the buffer.

**•INxCSR3(SET_IN_BUF_RDY):**

The CPU writes a "1" to this bit after loading a data set to the buffer. The CPU can only load data to the buffer and set this bit when INxCSR1 (IN_BUF_STS1) is a "0".

**•INxCSR4(CLR_UNDER_RUN):**

The CPU writes a "1" to this bit to clear the UNDER_RUN status flag.

**•INxCSR5(TOGGLE_INIT):**

The CPU writes a "1" to this bit to initialize the data sequence, force the next packet's data PID to a DATA0 for transmission. Setting the TOGGLE_INT bit also resets the FIFO read/write pointers.

**•INxCSR5(TOGGLE_INIT):**

The CPU writes a "1" to this bit to initialize the data sequence, force the next packet's data PID to a DATA0 for transmission. Setting the TOGGLE_INT bit also resets the FIFO read/write pointers.



Downloaded from Arrow.com.

**• INxCSR6 (FLUSH):**

The CPU writes a "1" to this bit to flush the IN buffer.

• When there is one data set in the IN buffer, a flush causes the IN buffer to be empty.

• When there are two data sets in the IN buffer, a flush causes the older data set to be flushed out from the IN buffer.

The USB FCU updates the buffer status bits the same way as a data set is transmitted to the host when it sees a FLUSH. Setting the FLUSH bit during transmission could produce unpredictable results.

**• INxCSR7 (INTPT):**

The CPU writes a "1" to this bit to initialize the endpoint as a rate feedback interrupt endpoint.

**• INxCSR8 (ISO):**

The CPU writes a "1" to this bit to set the endpoint as an isochronous data transfer endpoint.

**• INxCSR9 (SEND_STALL):**

The CPU writes a "1" to this bit when the endpoint is stalled (transmitter halt). The USB FCU returns STALL handshakes while this bit is set. The CPU writes a "0" to clear this bit, If the STALL condition no longer exists.

**• INxCSR10 (AUTO_SET):**

The CPU writes a "1" to this bit to enable the AUTO_SET function. AUTO_SET takes place only when a data packet that is equal to MAXP (or data set that is equal to BUF_SIZ, in continuous mode) is loaded to the buffer. See "IN (Transmit) FIFO" operation for details.

USB Endpoint x IN Control and Status register

(b15) b7 (b8) b0 b7 b0

0 0 0 0 0

Symbol: EPxICS (x = 1 - 4)

Address: $029E_{16}$, $02A4_{16}$, $02AA_{16}$, $02B0_{16}$

When reset: $0003_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| INxCSR0 | IN_BUF_STS0 flag | These two bits indicate the EPx IN buffer status<br>Bit1 Bit0<br>0 0 : No data set in the IN buffer<br>0 1 : Single buffer mode: N/A<br>Double buffer mode: one data set in the IN buffer<br>1 0 : Single buffer mode: N/A<br>Double buffer mode: N/A<br>1 1 : Single buffer mode: one data set in the IN buffer<br>Double buffer mode: two data sets in the IN buffer | O | X |
| INxCSR1 | IN_BUF_STS1 flag | | O | X |
| INxCSR2 | UNDER-RUN flag | 0 : No underrun detected<br>1 : Underrun detected | O | X |
| INxCSR3 | SET_IN_BUF_RDY | 0 : No action<br>1 : Data set loaded to the IN buffer (updates IN buffer status flags) | O | O Note |
| INxCSR4 | CLR_UNDER_RUN | 0 : No action<br>1 : Clears UNDER_RUN flag | O | O Note |
| INxCSR5 | TOGGLE_INT | 0 : No action<br>1 : Initialize the next data PID as a DATA0 for transmission | O | O Note |
| INxCSR6 | FLUSH | 0 : No action<br>1 : Flush out one data set | O | O Note |
| INxCSR7 | INTPT | 0 : Select non-rate feedback interrupt transfer<br>1 : Select rate feedback interrupt transfer | O | O |
| INxCSR8 | ISO | 0 : Select non-isochronous endpoint<br>1 : Select isochronous endpoint | O | O |
| INxCSR9 | SEND_STALL | 0 : No STALL by CPU<br>1 : STALL by CPU | O | O |
| INxCSR10 | AUTO_SET | 0 : AUTO_SET disabled<br>1 : AUTO_SET enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |

Note: Always read a "0"

**Figure 1.58. USB Endpoint x IN Control & Status register (EPxICS)**

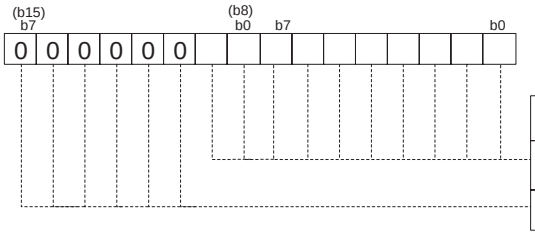


Downloaded from Arrow.com.

### USB Endpoint x IN MAXP Register (x = 1 to 4)

The USB Endpoint x IN MAXP Register, shown in Figure 1.59, indicates the maximum packet size (MAXP) of EPx IN packet. The default values for all EPx IN MAXP are 0 bytes.

USB Endpoint x IN MAXP register




Symbol: EPxIMP ( x = 1 - 4)
Address: $02A0_{16}$, $02A6_{16}$, $02AC_{16}$, $02B2_{16}$
When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| IMAXP9-0 | Maximum packet size | | O | O |
| Reserved | | Must always be "0" | O | O |

**Figure 1.59. USB Endpoint x IN MAXP register (EPxIMP)**

### USB Endpoint x IN FIFO configuration Register (x = 1 to 4)

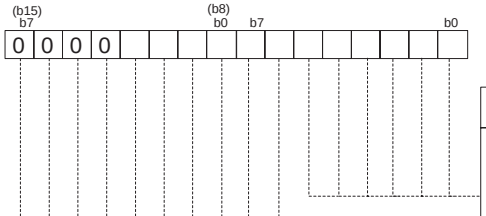
The USB Endpoint x IN FIFO Configuration Register, shown in Figure 1.60, is used to select various FIFO configurations. When the double buffer bit is set, the effective buffer size = 2 x BUF_SIZ. Therefore other EP FIFO buffer's starting locations have to be 2 x BUF_SIZ apart.

The user should ensure:

• Buffer Starting Location + Buffer Size do not exceed the 3K byte boundary.

• Endpoint buffers do not overlap with each other.

USB Endpoint x IN FIFO Configuration register




Symbol: EPxIFC (x = 1 - 4)
Address: $02A2_{16}$, $02A8_{16}$, $02AE_{16}$, $02B4_{16}$
When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| BUF_NUM | FIFO buffer start number | Select the starting number for the EPx IN FIFO (in units of 64 bytes)<br>000000 : buffer starting location = 0<br>000001 : buffer starting location = 64<br>000010 : buffer starting location = 128<br>.....<br>101111 : buffer starting location = 3008 (last starting number) | O | O |
| BUF_SIZ | FIFO buffer size | Select the buffer size for the EPx IN FIFO (in units of 64 bytes)<br>0000 : buffer size = 64<br>0001 : buffer size = 128<br>0010 : buffer size = 192<br>.....<br>1111 : buffer size = 1024 (largest buffer size) | O | O |
| DBL_BUF | Double buffer mode | 0 : Disabled<br>1 : Enabled | O | O |
| CONTINUE | Continuous transfer mode | 0 : Disabled (Note)<br>1 : Enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |

Note: Valid for bulk transfer type only

**Figure 1.60. USB Endpoint x IN FIFO Configuration register (EPxIFC)**



Downloaded from Arrow.com.

**USB Endpoint x OUT CSR (x = 1 to 4)**

The USB Endpoint x OUT CSR (Control and Status Register), shown in Figure 1.61, contains control and status information for the respective OUT EP 1-4.

**• OUTxCSR0 (OUT_BUF_STS0) and OUTxCSR1 (OUT_BUF_STS1):**

Two status flags, indicate the current status of the OUT buffer. The buffer status flags are updated when one of the following events occurs:

1. The USB FCU successfully receives a data set from the host.

2. The CPU unloads a data set from the buffer (writes a "1" to CLR_OUT_BUF_RDY).

3. The CPU writes a "1" to the FLUSH bit.

**• OUTxCSR2 (OVER_RUN):**

A status flag, "1" indicates an over run has occurred in an isochronous data transfer. The USB FCU updates this flag to a "1" at the beginning of an OUT token when two data packets are already present in the buffer.

**• OUTxCSR3(FORCE_STALL):**

A status flag, "1" indicates that the USB FCU detected a Packet size larger than MAXP violation. The USB FCU returns a STALL as a handshake packet for the current transaction.

**• OUTxCSR4(DATA_ERR):**

A status flag, "1" indicates a data error (bit stuffing or CRC error) has occurred in an OUT isochronous data packet.

**• OUTxCSR5(CLR_OUT_BUF_RDY):**

The CPU writes a "1" to this bit after unloading a data set from the buffer. The CPU can only unload data from the buffer and set this bit when OUTxCSR1 (OUT_BUF_STS1) is a "1".

**• OUTxCSR6(CLR_OVER_RUN):**

The CPU writes a "1" to this bit to clear the OVER_RUN status flag.

**• OUTxCSR7(CLR_FORCE_STALL):**

The CPU writes a "1" to this bit to clear the FORCE_STALL status flag.

**• OUTxCSR8(CLR_DATA_ERR):**

The CPU writes a "1" to this bit to clear the DATA_ERR status flag.

**• OUTxCSR9(TOGGLE_INIT):**

The CPU writes a "1" to this bit to initialize the data sequence, and force the next packet's data PID to a DATA0 for reception.

**• OUTxCSR10 (FLUSH):**

The CPU writes a "1" to this bit to flush the OUT buffer. This bit must only be set to a "1" when the OUT_BUF_STS1 flag is a "1".

• When there is one data set in the OUT buffer, a flush causes the OUT buffer to be empty.

• When there are two data sets in the OUT buffer, a flush causes the older packet to be flushed from the OUT buffer.

The USB FCU updates the buffer status flags the same way as a data set is unloaded from the host when it sees a FLUSH. Setting the FLUSH bit during reception could produce unpredictable results.



Downloaded from Arrow.com.

**• OUTxCSR11 (ISO):**

The CPU writes "1" to this bit to set the endpoint as an isochronous data transfer endpoint.

**• OUTxCSR12 (SEND_STALL):**

The CPU writes "1" to this bit when the endpoint is stalled (receiver halt). The USB FCU returns STALL handshakes while this bit is set. The CPU writes "0" to clear this bit, if the STALL condition no longer exists.

**• OUTxCSR13 (AUTO_CLR):**

The CPU writes "1" to this bit to enable the AUTO_CLR function. AUTO_CLR takes place when a data packet (or a data set, in continuous mode) is unloaded from the buffer, even if the data packet is less than MAXP (or data set is less than BUF_SIZ, in continuous mode). See "OUT (Receive) FIFO" operation for details.

USB Endpoint x OUT Control and Status register




Symbol: EPxOCS (x = 1 - 4)

Address: $02B6_{16}$, $02BE_{16}$, $02C6_{16}$, $02CE_{16}$

When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| OUTxCSR0 | OUT_BUF_STS0 flag | These two bits indicate the EPx OUT buffer status:<br>Bit1 Bit0<br>0 0 : No data set in the OUT buffer<br>0 1 : Single buffer mode: N/A<br>Double buffer mode: N/A<br>1 0 : Single buffer mode: N/A<br>Double buffer mode: one data set in the OUT buffer<br>1 1 : Single buffer mode: one data set in the OUT buffer<br>Double buffer mode: two data sets in the OUT buffer | O | X |
| OUTxCSR1 | OUT_BUF_STS1 flag | | O | X |
| OUTxCSR2 | OVER-RUN flag | 0 : No over run detected<br>1 : Over run detected | O | X |
| OUTxCSR3 | FORCE_STALL flag | 0 : No packet size larger than MAXP violation detected<br>1 : Packet size larger than MAXP violation detected | O | X |
| OUTxCSR4 | DATA_ERR flag | 0 : No data error detected<br>1 : Data error detected | O | X |
| OUTxCSR5 | CLR_OUT_BUF_RDY | 0 : No action<br>1 : Data set unloaded from the OUT buffer (updates status flags) | O Note | O |
| OUTxCSR6 | CLR_OVER_RUN | 0 : No action<br>1 : Clears OVER_RUN flag | O Note | O |
| OUTxCSR7 | CLR_FORCE_STALL | 0 : No action<br>1 : Clears FORCE_STALL flag | O Note | O |
| OUTxCSR8 | CLR_DATA_ERR | 0 : No action<br>1 : Clears DATA_ERR flag | O Note | O |
| OUTxCSR9 | TOGGLE_INIT | 0 : No action<br>1 : Initialize the next data PID as a DATA0 for reception | O Note | O |
| OUTxCSR10 | FLUSH | 0 : No action<br>1 : Flush out one data set | O Note | O |
| OUTxCSR11 | ISO | 0 : Select non-isochronous endpoint<br>1 : Select isochronous endpoint | O | O |
| OUTxCSR12 | SEND_STALL | 0 : No STALL by CPU<br>1 : STALL by CPU | O | O |
| OUTxCSR13 | AUTO_CLR | 0 : AUTO_CLR disabled<br>1 : AUTO_CLR enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |

Note: Always read a "0" when writing to this bit

**Figure 1.61. USB Endpoint x OUT Control and Status register (EPxOCS)**

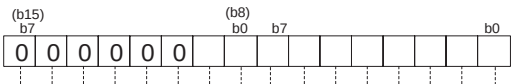


Downloaded from Arrow.com.

### USB Endpoint x OUT MAXP Register (x = 1 to 4)

The USB Endpoint x OUT MAXP register, shown in Figure 1.62, indicates the maximum packet size (MAXP) of EPx OUT packet. The default values for all EPx OUT MAXP are 0 bytes.

USB Endpoint x OUT MAXP register




Symbol: EPxOMP (x = 1 - 4)
Address: $02B8_{16}$, $02C0_{16}$, $02C8_{16}$, $02D0_{16}$
When reset: $0000_{16}$

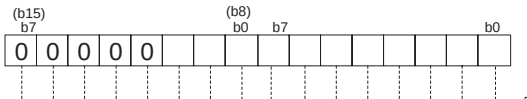
| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| OMAXP9-0 | Maximum packet size | | O | O |
| Reserved | | Must always be "0" | O | O |

**Figure 1.62. USB Endpoint x OUT MAXP register (EPxOMP)**

### USB Endpoint x OUT WRT CNT Register (x = 1 to 4)

The USB Endpoint x OUT WRT CNT Register, shown in Figure 1.63, contains the number of bytes of the current data set in the OUT buffer. The USB FCU sets the value in the WRT_CNT Register after having successfully received a data set from the host. The CPU reads the register to determine the number of bytes to be read from the buffer. The WRT_CNT value does not decrement upon a CPU read of the FIFO Data Register. The WRT_CNT value is cleared (in double buffer mode, the WRT_CNT value corresponding to the dataset being unloaded is cleared) when the CPU writes "1" to the CLR_OUT_BUF_RDY bit of the OUT CSR or in AUTO_CLR mode, when the data set is unloaded from the buffer.

USB Endpoint x OUT WRT CNT register




Symbol: EPxWC (x = 1 - 4)
Address: $02BA_{16}$, $02C2_{16}$, $02CA_{16}$, $02D2_{16}$
When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| WCNT10-0 | Receive byte count | | O | X |
| Reserved | | Must always be "0" | O | O |

**Figure 1.63. USB Endpoint x OUT WRT CNT register (EPxWC)**



Downloaded from Arrow.com.

### USB Endpoint x OUT FIFO configuration Register (x = 1 to 4)

The USB Endpoint x OUT FIFO Configuration Register, shown in Figure 1.64, is used to select various FIFO configurations. When double buffer bit is set, the effective buffer size = 2 x BUF_SIZ. Therefore other EP FIFO buffer's starting locations have to be 2 x BUF_SIZ apart.

The user should ensure:

• Buffer Starting Location + Buffer Size do not exceed the 3K byte boundary.

• Endpoint buffers do not overlap with each other.

USB Endpoint x OUT FIFO Configuration register

(b15) b7 (b8) b0 b7 b0

0 0 0 0

Symbol: EPxOFC (x = 1 - 4)

Address: $02BC_{16}$, $02C4_{16}$, $02CC_{16}$, $02D4_{16}$

When reset: $0000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| BUF_NUM | FIFO buffer start number | Select the starting number for the EPx OUT FIFO (in units of 64 bytes)<br>000000 : buffer starting location = 0<br>000001 : buffer starting location = 64<br>000010 : buffer starting location = 128<br>.....<br>101111 : buffer starting location = 3008 (last starting number) | O | O |
| BUF_SIZ | FIFO buffer size | Select the buffer size for the EPx OUT FIFO (in units of 64 bytes)<br>0000 : buffer size = 64<br>0001 : buffer size = 128<br>0010 : buffer size = 192<br>.....<br>1111 : buffer size = 1024 (largest buffer size) | O | O |
| DBL_BUF | Double buffer mode | 0 : Disabled<br>1 : Enabled | O | O |
| CONTINUE | Continuous transfer mode | 0 : Disabled (Note)<br>1 : Enabled | O | O |
| Reserved | | Must always be set to "0" | O | O |

Note: Valid for bulk transfer type only

**Figure 1.64. USB Endpoint x OUT FIFO register (EPxOFC)**

### USB Endpoint x IN FIFO Data Registers (x = 0 to 4)

The USB Endpoint x IN FIFO Data Registers, shown in Figure 1.65 are the USB IN (transmit) FIFO data registers. The CPU writes data to these registers for the respective Endpoint IN FIFO.

USB Endpoint x IN FIFO Data register

(b15) b7 (b8) b0 b7 b0

Symbol: EPxI (x = 0 - 4)

Address: $02E0_{16}$, $02E4_{16}$, $02E8_{16}$, $02EC_{16}$, $02F0_{16}$

When reset: N/A

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DATA_15-0 | EP0 IN FIFO Data | | X | O |

Note 1: Data is undefined if this register is read.
Note 2: Write only to this register with a Word command or a Byte command to the lower 8 bits. Do not write a byte of data to the upper 8 bits. (b8 - b15)

**Figure 1.65. USB Endpoint x IN FIFO Data register (EPxI)**

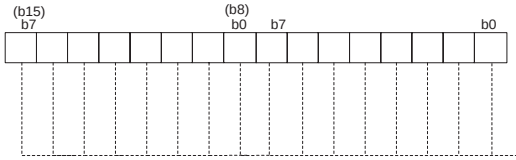


Downloaded from Arrow.com.

### USB Endpoint x OUT FIFO Data Register (x = 0 to 4)

The USB Endpoint x OUT FIFO Data Registers, shown in Figure 1.66 are the USB OUT (receive) FIFO data registers. The CPU reads data from these registers for the respective Endpoint OUT FIFO.

USB Endpoint x OUT FIFO Data register




| Symbol | Address | When reset |
|---|---|---|
| EPxO (x = 0 - 4) | $02E2_{16}$, $02E6_{16}$, $02EA_{16}$, $02EE_{16}$, $02F2_{16}$ | N/A |

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DATA_15-0 | EP0 OUT FIFO Data | | O | X |

Note 1: Writing to this register might cause a system error.
Note 2: Read only from this register with a Word command or a Byte command to the lower 8 bits. Do not read a byte of data from the upper 8 bits. (b8 - b15)

**Figure 1.66. USB Endpoint x OUT FIFO Data register (EPxO)**



Downloaded from Arrow.com.

## Vbus Detect

The Vbus Detect function will detect when the USB host is powered-up during USB self-powered operation. Self-powered operation means the microcontroller has a power source external to the USB. This type of connection requires the need to monitor when the USB host powers-up or powers-down. The other power mode, called Bus-powered mode, means the microcontroller is powered directly from the USB Vbus connection. This type of connection does not require the Vbus detect function because the microcontroller is actually powered from the USB so you know the USB host is already powered-up.

The VbusDTCT pin is used for the Vbus detect function. When operating the USB in self-powered mode, connect the Vbus line from the USB connector to the VbusDTCT pin. The Vbus detect function can be enabled or disabled in the USB attach/detach register (bit 7 at address 001F16). Each time the USB host powers up or powers down, a Vbus detect interrupt will be generated. This interrupt can be enabled or disable using the Vbus detect interrupt control register (address 005C16). When a Vbus detect interrupt is received, the Vbus detect state bit located in the Port 9 data register (bit 1 at address 03F116) should be read to determine if the Vbus is powered up or not.

Figure 1.67 is an example of the USB self-powered mode connection. Figure 1.68 shows the Vbus-related registers.

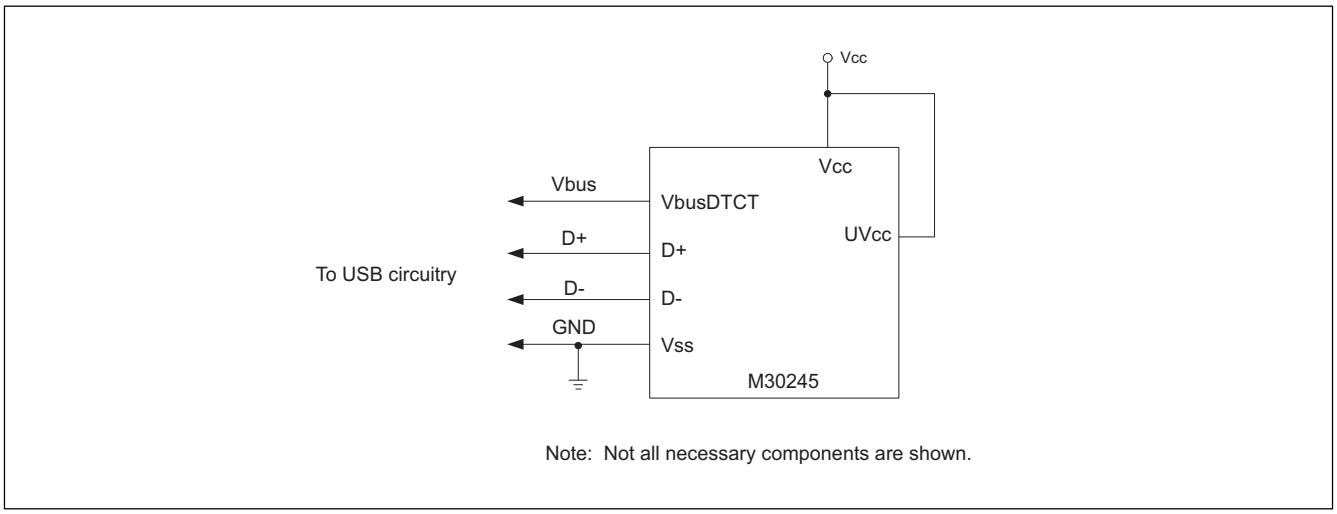



**Figure 1.67. USB self-powered mode connection example**

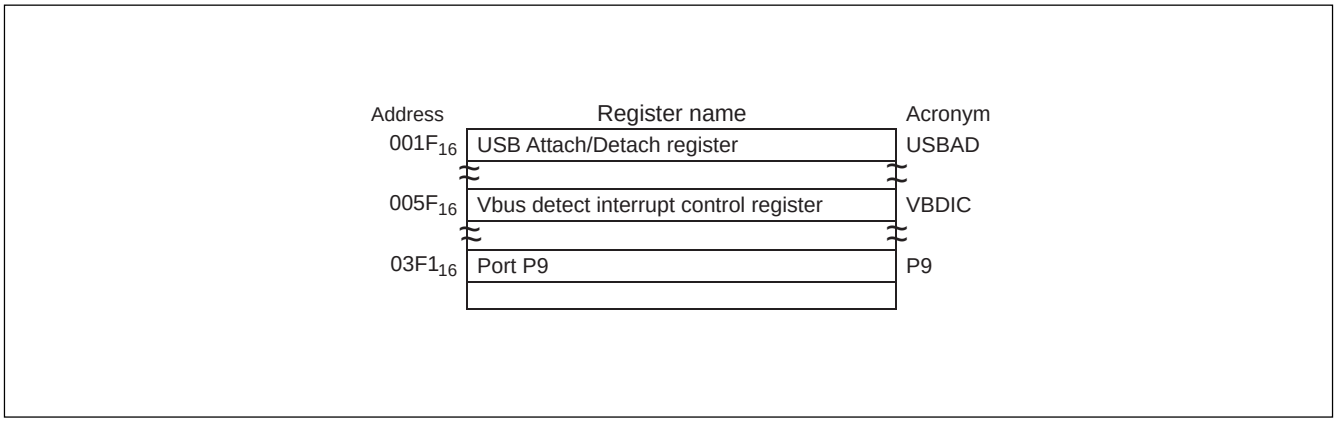



**Figure 1.68. Vbus-related memory map**



Downloaded from Arrow.com.

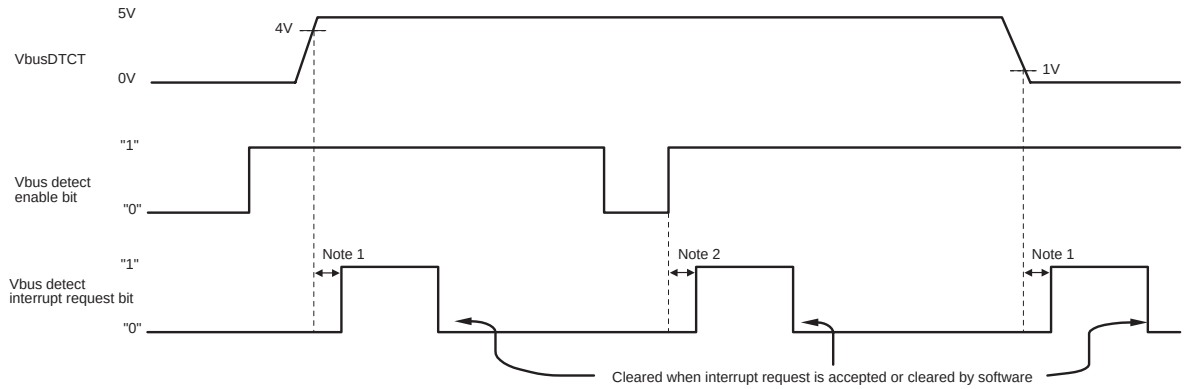
To avoid receiving a false Vbus detect interrupt at start-up, the Vbus detect should be enabled before enabling the Vbus detect interrupt. Use the following procedure when enabling the Vbus detect function:

1) Enable Vbus detect by setting the Vbus detect enable bit to "1" (bit 7 at address 001F16).

2) Clear the Vbus detect interrupt by setting the Vbus detect interrupt request bit to "0" (bit 3 at address 005C16).

3) Enable the Vbus detect interrupt by setting the Vbus detect interrupt priority level greater than "000" (bits 2-0 at address 005C16)

Figure 1.69 shows the Vbus detect interrupt timing




Note 1: Maximum time from Vbus detection to when interrupt request bit is set is 1ms.
Note 2: Maximum time from when setting Vbus detect enable bit to "1" to when interrupt requeset bit is set is 500us.
Note 3. VbusDTCT guaranteed minimum pulse width accepted is 50us.

**Figure 1.69. Vbus detect interrupt timing**



Downloaded from Arrow.com.

## Direct memory access controller

This microcomputer has four DMAC (direct memory access controller) channels that allow data to be sent to memory without using the CPU. DMAC shares the same data bus with the CPU. The DMAC is given a higher right of using the bus than the CPU, which leads to working the cycle stealing method. On this account, the operation from the occurrence of a DMA transfer request signal to the completion of 1-word (16-bit) or 1-byte (8-bit) data transfer can be performed at high speed. Figure 1.70 shows the DMAC block diagram. Table 1.37 shows the DMAC specifications. Figure 1.71 to Figure 1.73 show the registers used by the DMAC.

Either a write signal to the software DMA request bit or an interrupt request signal can be used as the DMA transfer request signal. But the DMA transfer is not affected by either the interrupt enable flag (I flag) or by the interrupt priority level. The DMA transfer doesn't affect any interrupts either.

If the DMAC is active (the DMA enable bit is set to 1), data transfer starts every time a DMA transfer request signal occurs. If the cycle of the occurrences of DMA transfer request signals is higher than the DMA transfer cycle, there can be instances in which the number of transfer requests doesn't match the number of transfers. For details, see the description of the DMA request bit.

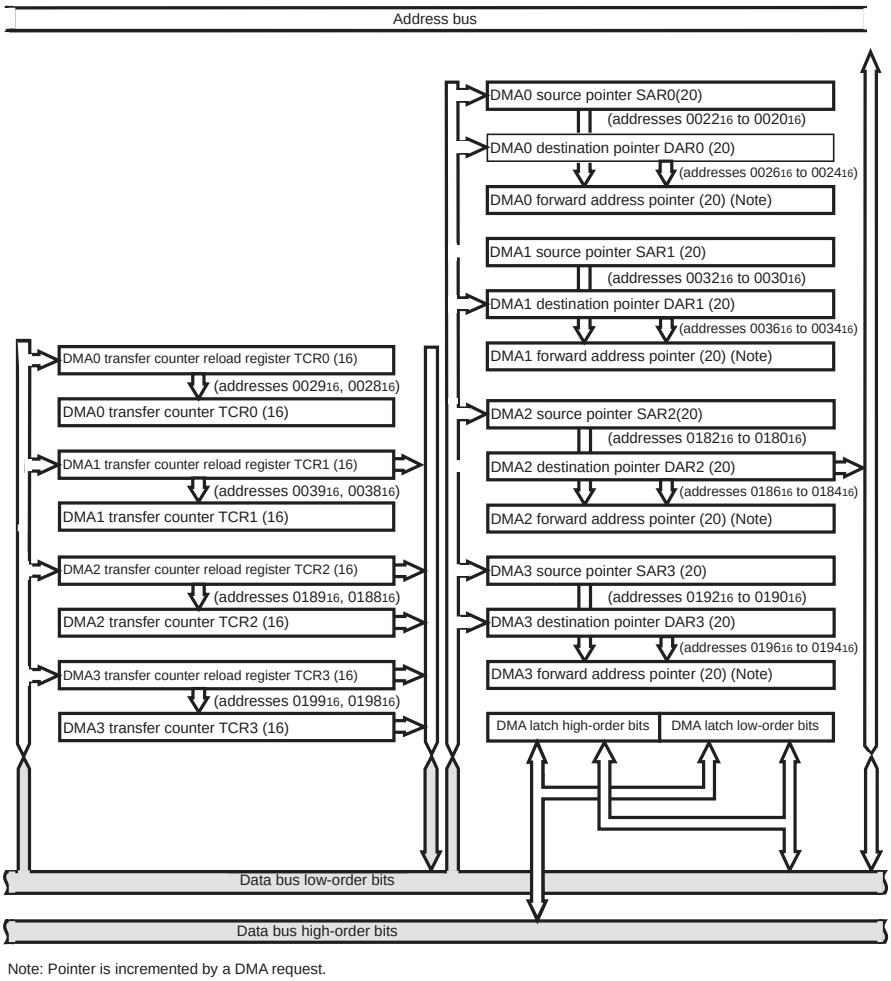



**Figure 1.70. DMAC block diagram**



Downloaded from Arrow.com.

**Table 1.37. DMAC specifications**

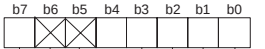
| Item | Specification |
|---|---|
| No. of channels | 4 (cycle steal method) |
| Transfer memory space | • From any address in the 1M bytes space to a fixed address<br>• From a fixed address to any address in the 1M bytes space<br>• From a fixed address to a fixed address (note that DMA related registers [$0020_{16}$ to $003F_{16}$ and $0180_{16}$ to $019F_{16}$] cannot be accessed) |
| Maximum No. of bytes transferred | 128K bytes (with 16-bit transfers) or 64K bytes (with 8-bit transfers) |
| DMA request factors (Note) | • Falling edge of $\overline{INT0}$, $\overline{INT1}$, $\overline{INT2}$ or both edges<br>• Timer A0 to Timer A4 interrupt requests<br>• UART0-3 transfer and receive interrupt requests<br>• A/D conversion interrupt request<br>• Software triggers<br>• DM Atriggers<br>• Serial Sound Interface 0-1 transmit and receive interrupt<br>• USB triggers, selectable by endpoint |
| Channel priority | High to low priority: DMA0, DMA1, DMA2, DMA3 |
| Transfer unit | 8 bits or 16 bits |
| Transfer address direction | Forward/fixed (forward direction cannot be specified for both source and destination simultaneously) |
| Transfer mode | • Single transfer mode<br>After the transfer counter underflows, the DMA enable bit is set to "0" and the DMAC becomes inactive<br>• Repeat transfer mode<br>After the transfer counter underflows, the value of the transfer counter reload register is reloaded to the transfer counter. The DMAC remains active unless a "0" is written to the DMA enable bit. |
| DMA interrupt request generation timing | When an underflow occurs in transfer counter |
| Active | • When the DMA enable bit is set to "1", the DMAC is active.<br>• When the DMAC is active, data transfer starts each time the DMA transfer request signal occurs. |
| Inactive | • When the DMA enable bit is set to "0", the DMAC is inactive<br>• After the transfer counter underflows in single transfer mode. |
| Forward address pointer and reload timing for transfer counter | When data transfer starts immediately after turning DMAC active, or when the transfer counter underflows in repeat transfer mode, the value of the source pointer or destination pointer (whichever is specified for forward direction) is reloaded to the forward direction address pointer, and the value of the transfer counter reload register is reloaded to the transfer counter. |
| Writing to register | • Registers specified for forward direction transfer are always write enabled.<br>• Registers specified for fixed address transfer are write-enabled when the DMA enable bit is "0". |
| Reading the register | Can be read at any time. However, when the DMA enable bit is "1", reading the register set-up as the forward register is the same as reading the value of the forward address pointer. |

Note: DMA transfer is not effective to any interrupts. DMA transfer is not affected by the interrupt enable flag (I flag) or by the interrupt priority level.



Downloaded from Arrow.com.

DMA0 request cause select register (Note)

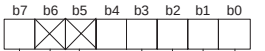



Symbol: DM0SL | Address: $03B8_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bits | b4 b3 b2 b1 b0<br>0 0 0 0 0 : Disabled<br>0 0 0 0 1 : INT0 (falling edge)<br>0 0 0 1 0 : INT0 (two edges)<br>0 0 0 1 1 : USB0<br>0 0 1 0 0 : Timer A0<br>0 0 1 0 1 : Timer A1<br>0 0 1 1 0 : Timer A2<br>0 0 1 1 1 : Timer A3<br>0 1 0 0 0 : Timer A4<br>0 1 0 0 1 : UART0 receive/ACK/SSI0 receive<br>0 1 0 1 0 : UART1 receive/ACK/SSI1 receive<br>0 1 0 1 1 : UART2 receive/ACK<br>0 1 1 0 0 : UART3 receive/ACK<br>0 1 1 0 1 : UART0 transmit/NACK/SSI0 transmit<br>0 1 1 1 0 : UART1 transmit/NACK/SSI1 transmit<br>0 1 1 1 1 : UART2 transmit/NACK<br>1 0 0 0 0 : UART3 transmit/NACK<br>1 0 0 0 1 : A/D<br>1 0 0 1 0 : Disabled<br>1 0 0 1 1 : DMA1<br>1 0 1 0 0 : DMA2<br>1 0 1 0 1 : DMA3<br>1 0 1 1 0 : Disabled<br>1 0 1 1 1 : Disabled<br>1 1 x x x : Disabled | O | O |
| DSEL1 | | | | |
| DSEL2 | | | | |
| DSEL3 | | | | |
| DSEL4 | | | | |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" when read. | | | – | – |
| DSR | Software DMA request bit | Software trigger is always enabled<br>Write "1" to trigger DSR bit. | O | O |

Note: Software is always enabled.

DMA1 request cause select register (Note)




Symbol: DM1SL | Address: $03BA_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bits | b4 b3 b2 b1 b0<br>0 0 0 0 0 : Disabled<br>0 0 0 0 1 : INT1 (falling edge)<br>0 0 0 1 0 : INT1 (two edges)<br>0 0 0 1 1 : USB1<br>0 0 1 0 0 : Timer A0<br>0 0 1 0 1 : Timer A1<br>0 0 1 1 0 : Timer A2<br>0 0 1 1 1 : Timer A3<br>0 1 0 0 0 : Timer A4<br>0 1 0 0 1 : UART0 receive/ACK/SSI0 receive<br>0 1 0 1 0 : UART1 receive/ACK/SSI1 receive<br>0 1 0 1 1 : UART2 receive/ACK<br>0 1 1 0 0 : UART3 receive/ACK<br>0 1 1 0 1 : UART0 transmit/NACK/SSI0 transmit<br>0 1 1 1 0 : UART1 transmit/NACK/SSI1 transmit<br>0 1 1 1 1 : UART2 transmit/NACK<br>1 0 0 0 0 : UART3 transmit/NACK<br>1 0 0 0 1 : A/D<br>1 0 0 1 0 : DMA0<br>1 0 0 1 1 : Disabled<br>1 0 1 0 0 : DMA2<br>1 0 1 0 1 : DMA3<br>1 0 1 1 0 : Disabled<br>1 0 1 1 1 : Disabled<br>1 1 x x x : Disabled | O | O |
| DSEL1 | | | | |
| DSEL2 | | | | |
| DSEL3 | | | | |
| DSEL4 | | | | |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" when read. | | | – | – |
| DSR | Software DMA request bit | Software trigger is always enabled<br>Write "1" to trigger DSR bit. | O | O |

Note: Software is always enabled.

**Figure 1.71. DMAC register (1)**



Downloaded from Arrow.com.

DMA2 request cause select register (Note)

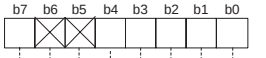



Symbol: DM2SL | Address: $03B0_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bits | b4 b3 b2 b1 b0<br>0 0 0 0 0 : Disabled<br>0 0 0 0 1 : INT2 (falling edge)<br>0 0 0 1 0 : INT2 (two edges)<br>0 0 0 1 1 : USB2<br>0 0 1 0 0 : Timer A0<br>0 0 1 0 1 : Timer A1<br>0 0 1 1 0 : Timer A2<br>0 0 1 1 1 : Timer A3<br>0 1 0 0 0 : Timer A4<br>0 1 0 0 1 : UART0 receive/ACK/SSI0 receive<br>0 1 0 1 0 : UART1 receive/ACK/SSI1 receive<br>0 1 0 1 1 : UART2 receive/ACK<br>0 1 1 0 0 : UART3 receive/ACK<br>0 1 1 0 1 : UART0 transmit/NACK/SSI0 transmit<br>0 1 1 1 0 : UART1 transmit/NACK/SSI1 transmit<br>0 1 1 1 1 : UART2 transmit/NACK<br>1 0 0 0 0 : UART3 transmit/NACK<br>1 0 0 0 1 : A/D<br>1 0 0 1 0 : DMA0<br>1 0 0 1 1 : DMA1<br>1 0 1 0 0 : Disabled<br>1 0 1 0 1 : DMA3<br>1 0 1 1 0 : Disabled<br>1 0 1 1 1 : Disabled<br>1 1 x x x : Disabled | O | O |
| DSEL1 | | | O | O |
| DSEL2 | | | O | O |
| DSEL3 | | | O | O |
| DSEL4 | | | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" when read. | | | – | – |
| DSR | Software DMA request bit | Software trigger is always enabled<br>Write "1" to trigger DSR bit. | O | O |

Note: Software is always enabled.

DMA3 request cause select register (Note)




Symbol: DM3SL | Address: $03B2_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bits | b4 b3 b2 b1 b0<br>0 0 0 0 0 : Disabled<br>0 0 0 0 1 : INT0 (falling edge)<br>0 0 0 1 0 : INT0 (two edges)<br>0 0 0 1 1 : USB3<br>0 0 1 0 0 : Timer A0<br>0 0 1 0 1 : Timer A1<br>0 0 1 1 0 : Timer A2<br>0 0 1 1 1 : Timer A3<br>0 1 0 0 0 : Timer A4<br>0 1 0 0 1 : UART0 receive/ACK/SSI0 recieve<br>0 1 0 1 0 : UART1 receive/ACK/SSI1 receive<br>0 1 0 1 1 : UART2 receive/ACK<br>0 1 1 0 0 : UART3 receive/ACK<br>0 1 1 0 1 : UART0 transmit/NACK/SSI0 transmit<br>0 1 1 1 0 : UART1 transmit/NACK/SSI1 transmit<br>0 1 1 1 1 : UART2 transmit/NACK<br>1 0 0 0 0 : UART3 transmit/NACK<br>1 0 0 0 1 : A/D<br>1 0 0 1 0 : DMA0<br>1 0 0 1 1 : DMA1<br>1 0 1 0 0 : DMA2<br>1 0 1 0 1 : Disabled<br>1 0 1 1 0 : Disabled<br>1 0 1 1 1 : Disabled<br>1 1 x x x : Disabled | O | O |
| DSEL1 | | | O | O |
| DSEL2 | | | O | O |
| DSEL3 | | | O | O |
| DSEL4 | | | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" when read. | | | – | – |
| DSR | Software DMA request bit | Software trigger is always enabled<br>Write "1" to trigger DSR bit. | O | O |

Note: Software is always enabled.

**Figure 1.72. DMAC register (2)**



Downloaded from Arrow.com.

DMAi control register

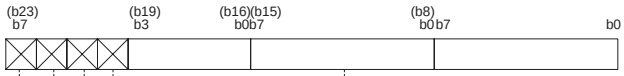



Symbol: DMiCON (i=0-3)
Address: $002C_{16}$, $003C_{16}$, $018C_{16}$, $019C_{16}$
When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| DMBIT | Transfer unit select bit | 0 : 16 bits<br>1 : 8 bits | O | O |
| DMASL | Repeat transfer mode select bit | 0 : Single transfer<br>1 : Repeat transfer | O | O |
| DMAS | DMA request bit (Note 1) | 0 : DMA not requested<br>1 : DMA requested | O | O (Note 2) |
| DMAE | DMA enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| DSD | Source address direction select bit (Note 3) | 0 : Fixed<br>1 : Forward | O | O |
| DAD | Destination address direction select bit (Note 3) | 0 : Fixed<br>1 : Forward | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is "0" when read. | | | – | – |

Note 1: DMA request can be cleared by resetting the bit.
Note 2: This bit can only be set to "0".
Note 3: Source address direction select bit and destination address direction select bit cannot be set to "1" simultaneously.

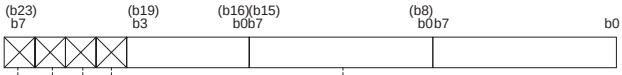
DMAi source pointer (i=0-3)




| Symbol | Address | When reset |
|---|---|---|
| SAR0 | $0022_{16}$ to $0020_{16}$ | Indeterminate |
| SAR1 | $0032_{16}$ to $0030_{16}$ | Indeterminate |
| SAR2 | $0182_{16}$ to $0180_{16}$ | Indeterminate |
| SAR3 | $0192_{16}$ to $0190_{16}$ | Indeterminate |

| Function | Transfer count specification | R | W |
|---|---|---|---|
| Source pointer<br>stores the source address | $00000_{16}$ to $FFFFF_{16}$ | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is "0" if read. | | – | – |

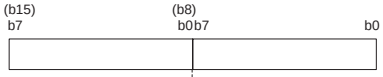
DMAi destination pointer (i=0-3)




| Symbol | Address | When reset |
|---|---|---|
| DAR0 | $0026_{16}$ to $0024_{16}$ | Indeterminate |
| DAR1 | $0036_{16}$ to $0034_{16}$ | Indeterminate |
| DAR2 | $0186_{16}$ to $0184_{16}$ | Indeterminate |
| DAR3 | $0196_{16}$ to $0194_{16}$ | Indeterminate |

| Function | Transfer count specification | R | W |
|---|---|---|---|
| Destination pointer<br>stores the destination address | $00000_{16}$ to $FFFFF_{16}$ | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is "0" if read. | | – | – |

DMAi transfer counter (i=0-3)




| Symbol | Address | When reset |
|---|---|---|
| TCR0 | $0029_{16}$ to $0028_{16}$ | Indeterminate |
| TCR1 | $0039_{16}$ to $0038_{16}$ | Indeterminate |
| TCR2 | $0189_{16}$ to $0188_{16}$ | Indeterminate |
| TCR3 | $0199_{16}$ to $0198_{16}$ | Indeterminate |

| Function | Transfer count specification | R | W |
|---|---|---|---|
| Transfer counter<br>Set a value one less than the transfer count | $0000_{16}$ to $FFFF_{16}$ | O | O |

**Figure 1.73. DMAC register (3)**



Downloaded from Arrow.com.

### Transfer modes

#### Single transfer mode

DMA transfer occurs until the tranfer counter underflows. Afterward, the DMA becomes inactive.

#### Repeat transfer mode

The DMA remains active even after the transfer counter underflows. The transfer counter and forward direction address pointer are reloaded after each transfer counter underflow. The DMA becomes inactive when "0" is written to the DMA enable bit.

### DMA enable bit

Setting the DMA enable bit to "1" makes the DMAC active. If data transfer starts immediately after the DMAC is turned active, the following operations are carried out:

(1) Reloads the value of either the source pointer or the destination pointer - the one specified for the forward direction - to the forward direction address pointer.

(2) Reloads the value of the transfer counter reload register to the transfer counter.

Thus writing "1" to the DMA enable bit with the DMAC being active carries out the operations given above, so the DMAC operates again from the initial state at the instant "1" is written to the DMA enable bit.

### DMA request bit

The DMAC can generate a DMA transfer request signal triggered by a factor chosen in advance out of DMA request causes for each channel (DMiSL registers).
DMA request causes include the following.

• Internal causes triggered by using the interrupt request signals from the built-in peripheral functions and software DMA request all controlled by software.

• External causes effected by utilizing the input from external interrupt signals.

For the selection of DMA request causes, see the descriptions of the DMAi request cause select registers.
The DMA request bit turns to "1" if the DMA transfer request signal occurs regardless of the DMAC's state (regardless of whether the DMA enable bit is set "1" or to "0"). It turns to "0" immediately before data transfer starts.
In addition, this bit can be set to "0" by software, but it cannot be set to "1".
There can be instances in which a change in the DMA request cause selection bits causes the DMA request bit to turn to "1". Make sure to set the DMA request bit to "0" after the DMA request cause selection bits are changed.
The DMA request bit turns to "1" if a DMA transfer request signal occurs, and turns to "0" immediately before data transfer starts. If the DMAC is active, data transfer starts immediately, so the value of the DMA request bit, if read by software, turns out to be "0" in most cases. To examine whether the DMAC is active, read the DMA enable bit.
The timing changes of the DMA request bit are discussed in the following section.



Downloaded from Arrow.com.

**Internal factors**

Except the DMA request factors triggered by software, the timing for the DMA request bit to turn to "1" due to an internal factor is the same as the timing for the interrupt request bit of the interrupt control register to turn to "1" due to several factors.
Turning the DMA request bit to "1" due to an internal factor is timed to be effected immediately before the transfer starts.

**External factors**

An external factor is a DMA request caused from the INTi pin input edge ("i" reflects the DMAC channel used).
Selecting the INTi pins as external factors using the DMA request factor selection bit causes input from these pins to become the DMA transfer request signals.
The timing for the DMA request bit to turn to "1" when an external factor is selected synchronizes with the signal's edge applicable to the function specified by the DMA request factor selection bit (synchronizes with the trailing edge of the input signal to each INTi pin, for example).
With an external factor selected, the DMA request bit is timed to turn to "0" immediately before data transfer starts similarly to the state in which an internal factor is selected.

### Priorities of the channels and DMA transfer timing

If a DMA transfer request signal falls on a single sampling cycle (a sampling cycle means one period from the leading edge to the trailing edge of BCLK), the DMA request bits of applicable channels concurrently turn to "1". If the channels are active at that moment, DMA0 is given a high priority to start data transfer. When DMA0 finishes data transfer, it gives the bus right to the CPU. When the CPU finishes single bus access, then DMA1 starts data transfer and gives the bus right to the CPU. The DMA priority levels are:

DMA0 > DMA1 > DMA2 > DMA3

Figure 1.74 is an example of DMA transfer effected by external factors when DMA0 and DMA1 requests occur in the same sampling cycle.

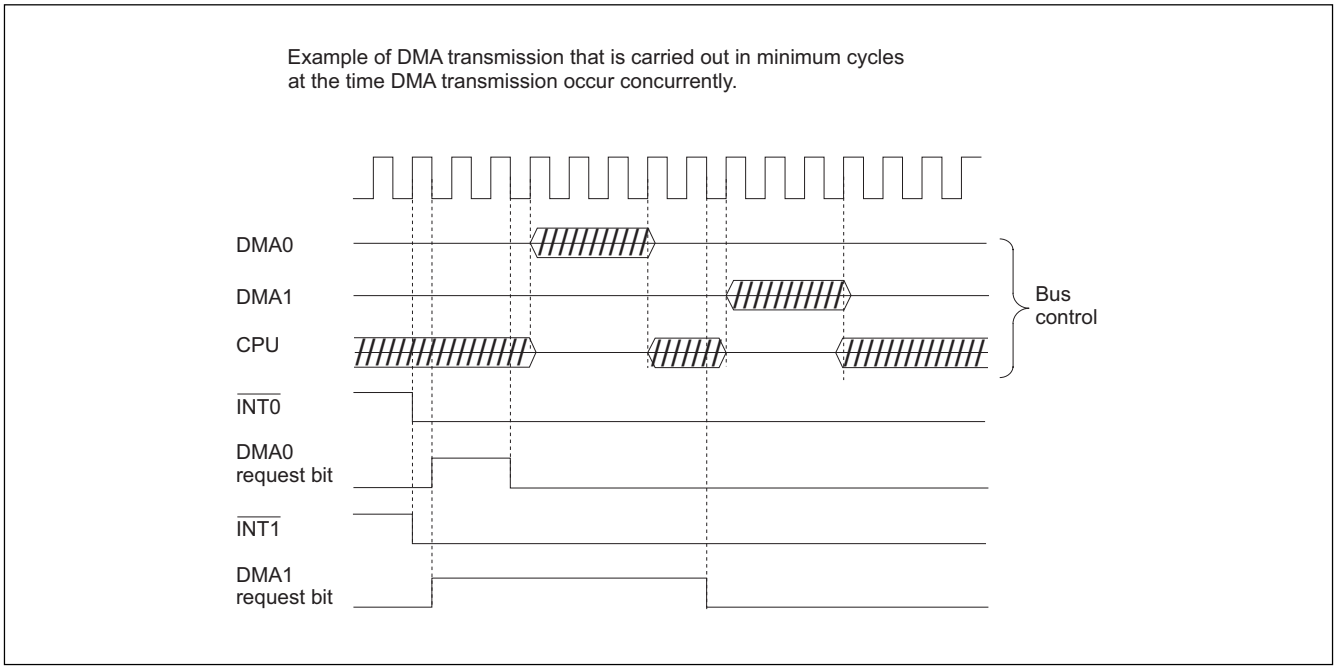



**Figure 1.74. An example of DMA transfer by external factors**



Downloaded from Arrow.com.

### Transfer cycle

The transfer cycle consists of the bus cycle in which data is read from memory or from the SFR area (source read) and the bus cycle in which the data is written to memory or to the SFR area (destination write). The number of read and write bus cycles depends on the source and destination addresses. In memory expansion mode and microprocessor mode, the number of read and write bus cycles also depends on the level of the BYTE pin. Also, the bus cycle is longer when software waits are inserted.

#### Effect of source and destination addresses

When 16-bit data is transferred on a 16-bit data bus, and the source and destination both start at odd addresses, there are one more source read cycle and destination write cycle than when the source and destination both start at even addresses.

#### Effect of BYTE pin level

When transferring 16-bit data over an 8-bit data bus (BYTE pin = H") in memory expansion mode and microprocessor mode, the 16 bits of data are sent in two 8-bit blocks. Therefore, two bus cycles are required for reading the data and two are required for writing the data. Also, in contrast to when the CPU accesses internal memory, when the DMAC accesses internal memory (internal ROM, internal RAM, and SFR), these areas are accessed using the data size selected by the BYTE pin.

#### Effect of software wait

When the SFR area or a memory area with a software wait is accessed, the number of cycles is increased for the wait by 1 bus cycle. The length of the cycle is determined by BCLK.

Figure 1.75 shows the example of the transfer cycles for a source read. For convenience, the destination write cycle is shown as one cycle and the source read cycles for the different conditions are shown. In reality, the destination write cycle is subject to the same conditions as the source read cycle, with the transfer cycle changing accordingly. When calculating the transfer cycle, remember to apply the respective conditions to both the destination write cycle and the source read cycle. For example, if data is being transferred in 16-bit units on an 8-bit bus (2), two bus cycles are required for both the source read cycle and the destination write cycle.

#### Transfer cycle calculations

Any combination of even or odd transfer read and write addresses is possible. Table 1.38a shows the number of DMAC transfer cycles. Table 1.38b shows the Coefficient j,k.

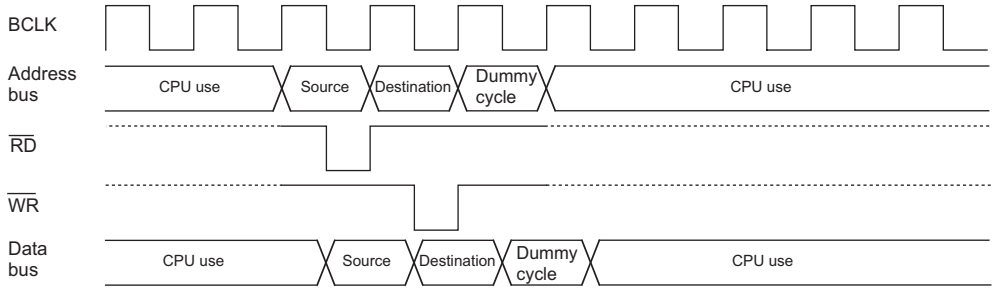
The number of DMAC transfer cycles calculation is:

No. of transfer cycles per transfer unit = No. of read cycles x j + No. of write cycles x k

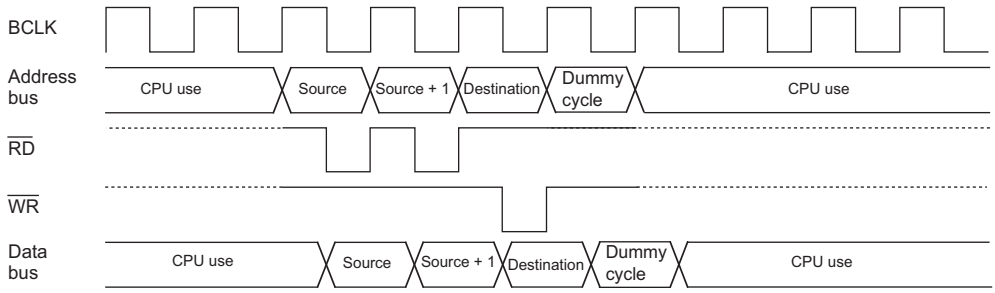


Downloaded from Arrow.com.

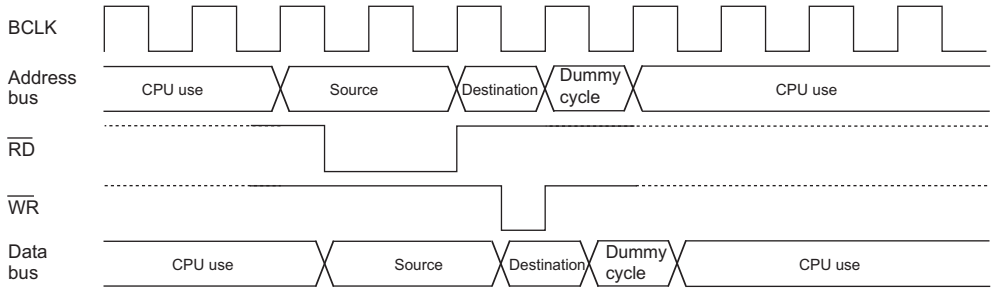
**(1) 8-bit transfers**
**16-bit transfers from even address and the source address is even.**



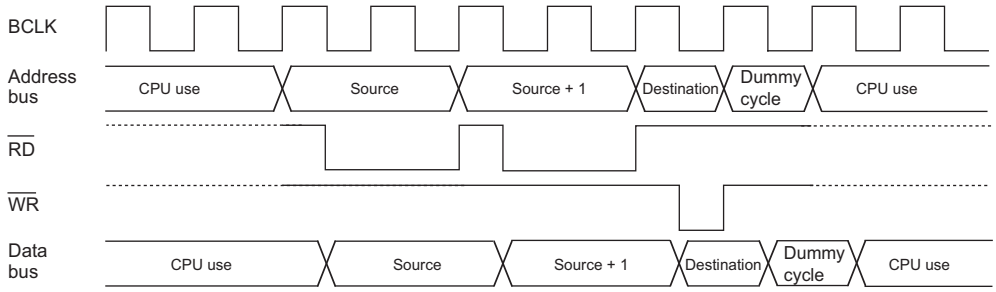

**(2) 16-bit transfers and the source address is odd**
**Transferring 16-bit data on an 8-bit data bus (In this case, there are two destination write cycles)**




**(3) One wait is inserted into the source read under the conditions in (1)**




**(4) One wait is inserted into the source read under the conditions in (2)**
**(When 16-bit data is transferred on an 8-bit data bus, there are two destination write cycles)**




Note : The same timing changes occur with the respective conditions at the destination as at the source.

**Figure 1.75. Example of the transfer cycles for a source read**



Downloaded from Arrow.com.

**Table 1.38a. DMA transfer cycles**

| Transfer unit | Bus width | Acces address | Single-chip mode | | Memory expansion mode Microprocessor mode | |
|---|---|---|---|---|---|---|
| | | | No. of read cycles | No. of write cycles | No. of read cycles | No. of write cycles |
| 8-bit transfers (DMBIT = "1") | 16-bit (BYTE = "L") | Even | 1 | 1 | 1 | 1 |
| | | Odd | 1 | 1 | 1 | 1 |
| | 8-bit (BYTE = "H") | Even | – | – | 1 | 1 |
| | | Odd | – | – | 1 | 1 |
| 16-bit transfers (DMBIT = "0") | 16-bit (BYTE = "L") | Even | 1 | 1 | 1 | 1 |
| | | Odd | 2 | 2 | 2 | 2 |
| | 8-bit (BYTE = "H") | Even | – | – | 2 | 2 |
| | | Odd | – | – | 2 | 2 |

**Table 1.38b. Coefficient j,k**

| | Internal memory | | External memory | | | |
|---|---|---|---|---|---|---|
| | Internal ROM/RAM | SFR area | no wait | with wait (Note 1) | | |
| | | | | 1 wait | 2 waits | 3 waits |
| j | 1 | 2 | 1 | 2 | 3 | 4 |
| k | 1 | 2 | 2 | 2 | 3 | 4 |

Note 1: Depends on the value set in the CSE register.

## Precautions

### Writing to the DMAE bit in DMiCON register

If the following conditions are met:

The DMAE bit is set to "1" again while it is already set to "1" (DMAi is in active state).

A DMA request may occur simultaneously when the DMAE bit is being written.

Follow the steps below:

Step 1: Write "1" to the DMAE bit and DMAS bit in DMiCON register simultaneously (Note 1).

Step 2: Make sure that the DMAi is in an initial state (Note 2) in a program.

If the DMAi is not in an initial state, the above steps should be repeated.

Note 1: The DMAS bit remains unchanged even if "1" is written. However, if "0" is written to this bit, it is set to "0" (DMA not requested). In order to prevent the DMAS bit from being modified to "0", "1" should be written to the DMAS bit when "1" is written to the DMAE bit. In this way the state of the DMAS bit immediately before being written can be maintained. Similarly, when writing to the DMAE bit with a read-modify-write instruction, "1" should be written to the DMAS bit in order to maintain a DMA request which is generated during execution.

Note 2: Read the TCRi register to verify whether the DMAi is in an initial state. If the read value is equal to a value which was written to the TCRi register before DMA transfer start, the DMAi is in an initial state. (If a DMA request occurs after writing to the DMAE bit, the value written to the TCRi register is "1".) If the read value is a value in the middle of a transfer, the DMAi is not in an initial state.



Downloaded from Arrow.com.

## Timer A

Except in event counter mode, Timers A0 through A4 all have the same function. Use the Timer Ai mode register (i = 0 to 4) bits 0 and 1 to choose the desired mode.

Timer A has four operation modes listed as follows:

• Timer mode: The timer counts an internal count source.

• Event counter mode: The timer counts pulses from an external source or a timer over flow.

• One-shot timer mode: The timer stops counting when the count reaches "0000$_{16}$".

• Pulse width modulation (PWM) mode: The timer outputs pulses of a given width.

Figure 1.76 and Figure 1.77 show block diagrams of Timer A. Figure 1.78 to Figure 1.80 show the Timer A-related registers.

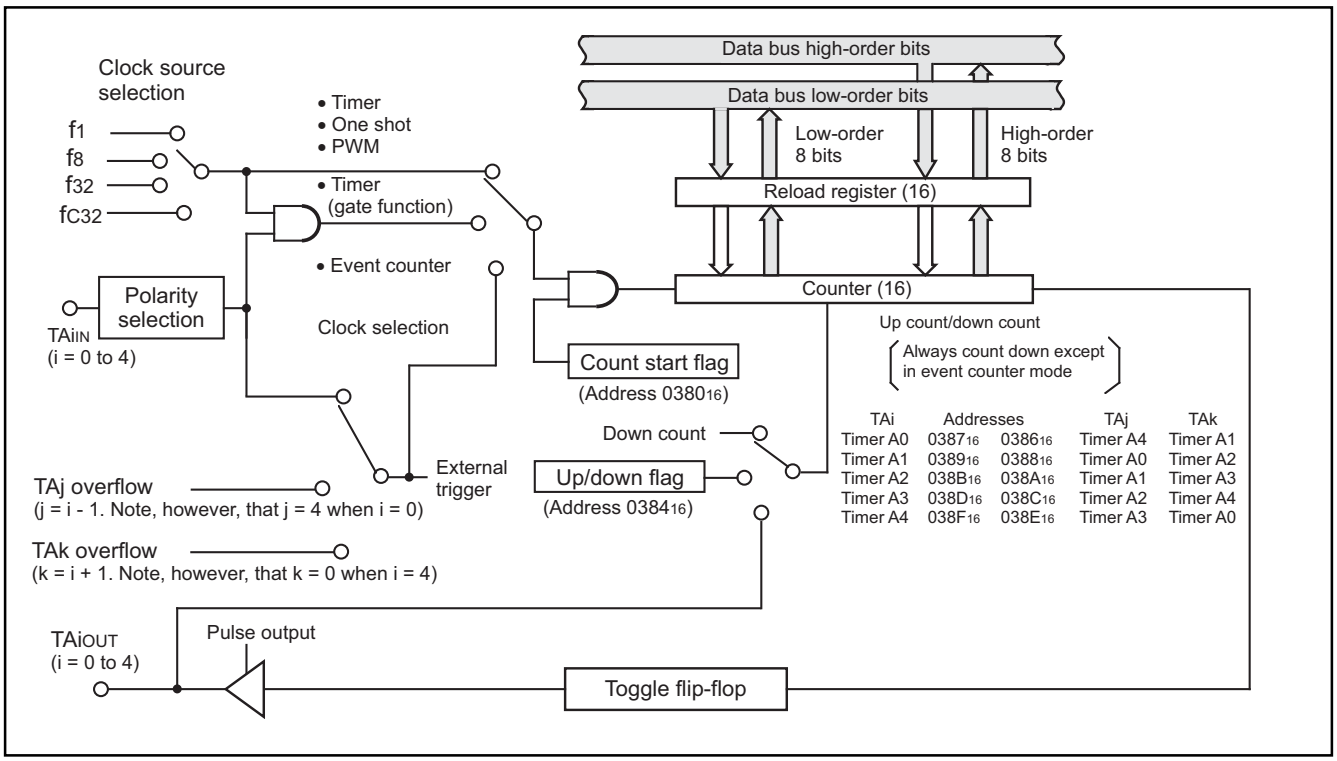



**Figure 1.76. Timer A block diagram (1)**



Downloaded from Arrow.com.

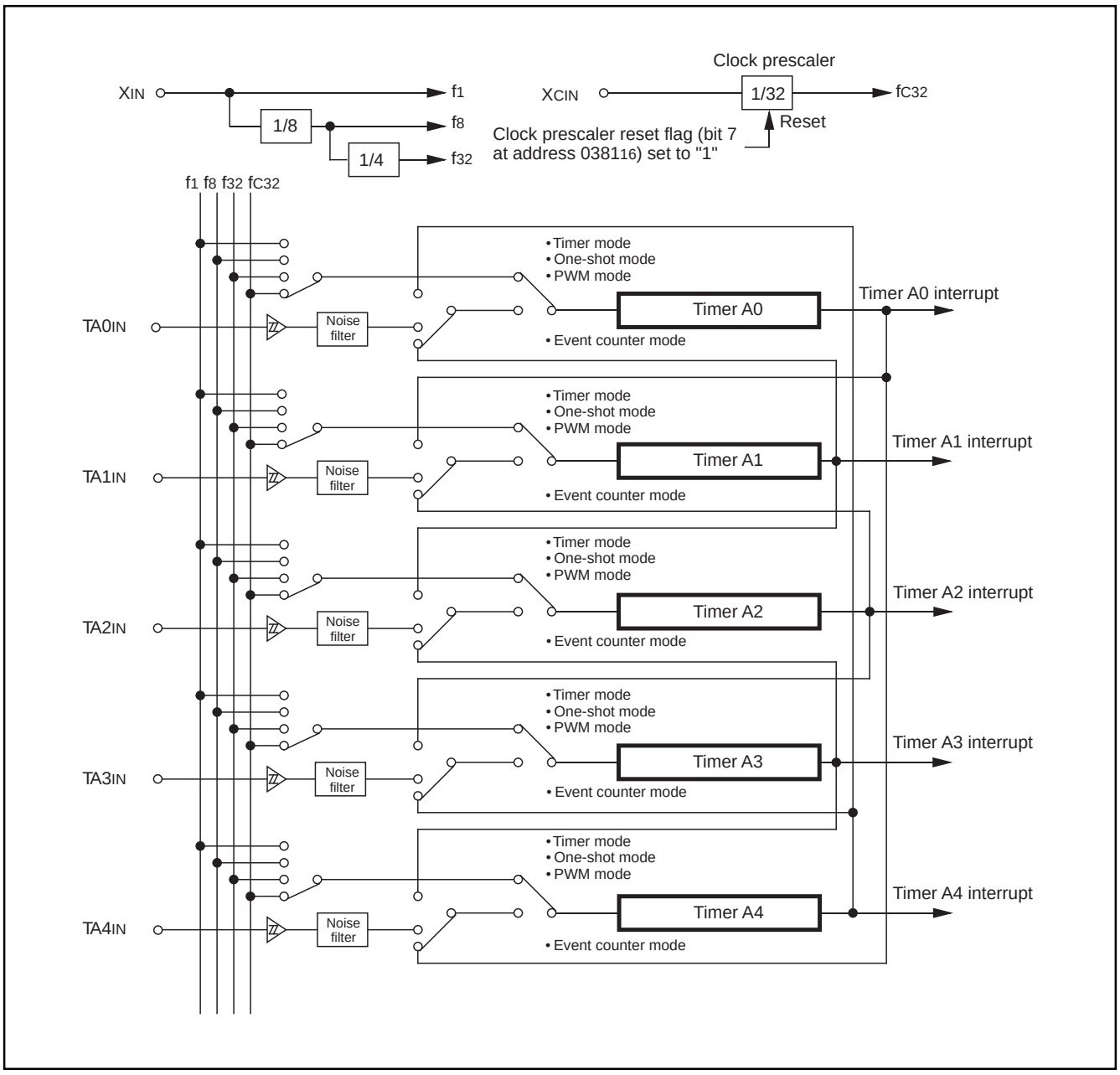
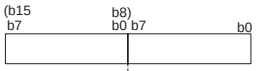



**Figure 1.77. Timer A block diagram (2)**



Downloaded from Arrow.com.

Timer Ai register (i = 0 to 4) (Note 1)




| Symbol | Address | When reset |
|---|---|---|
| TA0 | $0387_{16}$, $0386_{16}$, | Indeterminate |
| TA1 | $0389_{16}$, $0388_{16}$, | Indeterminate |
| TA2 | $038B_{16}$, $038A_{16}$, | Indeterminate |
| TA3 | $038D_{16}$, $038C_{16}$, | Indeterminate |
| TA4 | $038F_{16}$, $038E_{16}$ | Indeterminate |

| Mode | Function | Values that can be set | R | W |
|---|---|---|---|---|
| Timer mode | 16-bit counter (set to divide ratio) | $0000_{16}$ to $FFFF_{16}$ | O | O |
| Event counter mode | 16-bit counter (set to divide ratio) (Note 2) | $0000_{16}$ to $FFFF_{16}$ | O | O |
| One-shot timer mode | 16-bit counter (set to one-shot width) (Note 6) | $0000_{16}$ to $FFFF_{16}$ (Note 3) | X | O |
| 16-bit PWM | 16-bit PWM (set to PWM pulse "H" width) (Note 4, 7) | $0000_{16}$ to $FFFF_{16}$ (Note 3) | X | O |
| 8-bit PWM | Low-order bits: 8-bit prescaler (set to PWM period) (Notes 5, 7) High-order bits : 8-bit PWM (set to PWM pulse "H" width) (Notes 5, 7) | $00_{16}$ to $FE_{16}$ (Both high-order and low-order addresses) (Note 3) | X | O |

Note 1 : Read and write data in 16-bit units.
Note 2 : Counts pulses from an external source or timer overflow.
Note 3 : Use MOV instruction to write to this register.
Note 4 : When setting value is n, PWM period and "H" width of PWM pulses are:
PWM period : $(2^{16} - 1)/fi$
PWM pulse "H" width : n/fi
Note 5 : When setting value of high-order address is n and setting value of low-order address is m, PWM period and "H" width of PWM pulse are:
PWM period : $(2^8 - 1) \times (m + 1)/fi$
PWM pulse "H" width : $(m + 1)n/fi$
Note 6 : When the Timer Ai register is set to "$0000_{16}$", the counter does not operate and the Timer Ai interrupt request is not generated. When the pulse is set to output, the pulse is not output from the TAiOUT pin.
Note 7 : When the Timer Ai register is set to "$0000_{16}$", the pulse width modulator does not operate and the output level of the TAiOUT pin remains "L" level, therefore the Timer Ai interrupt request is not generated. This also occurs in the 8-bit pulse width modulator mode when the significant 8 high-order bits in the Timer Ai register are set to "$00_{16}$".

Trigger select register




| Symbol | Address | When reset |
|---|---|---|
| TRGSR | $0383_{16}$ | $00_{16}$ |

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TA1TGL | Timer A1 event/trigger select bit | b1 b0<br>0 0 : Input on $TA1_{IN}$ is selected (Note)<br>0 1 : Invalid<br>1 0 : TA0 overflow is selected<br>1 1 : TA2 overflow is selected | O | O |
| TA1TGH | | | O | O |
| TA2TGL | Timer A2 event/trigger select bit | b3 b2<br>0 0 : Input on $TA2_{IN}$ is selected (Note)<br>0 1 : Invalid<br>1 0 : TA1 overflow is selected<br>1 1 : TA3 overflow is selected | O | O |
| TA2TGH | | | O | O |
| TA3TGL | Timer A3 event/trigger select bit | b5 b4<br>0 0 : Input on $TA3_{IN}$ is selected (Note)<br>0 1 : Invalid<br>1 0 : TA2 overflow is selected<br>1 1 : TA4 overflow is selected | O | O |
| TA3TGH | | | O | O |
| TA4TGL | Timer A4 event/trigger select bit | b7 b6<br>0 0 : Input on $TA4_{IN}$ is selected (Note)<br>0 1 : Invalid<br>1 0 : TA3 overflow is selected<br>1 1 : TA0 overflow is selected | O | O |
| TA4TGH | | | O | O |

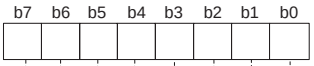
Note: Set the corresponding port direction register to "0"

**Figure 1.78. Timer A-related registers (1)**



Downloaded from Arrow.com.

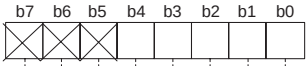
Timer Ai mode register (i = 0 to 4)




Symbol: TAiMR (i=0 to 4) | Address: $0396_{16}$ to $039A_{16}$ | When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode<br>0 1 : Event counter mode<br>1 0 : One-shot timer mode<br>1 1 : PWM mode | O | O |
| TMOD1 | | | O | O |
| MR0 | ——— | Function varies with each mode operation | O | O |
| MR1 | | | O | O |
| MR2 | | | O | O |
| MR3 | | | O | O |
| TCK0 | Count source select bit | Function varies with each mode operation | O | O |
| TCK1 | | | O | O |

Count start flag




Symbol: TABSR | Address: $0380_{16}$ | When reset: $XXX00000_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TA0S | Timer A0 count start flag | 0 : Stops counting<br>1 : Starts counting | O | O |
| TA1S | Timer A1 count start flag | | O | O |
| TA2S | Timer A2 count start flag | | O | O |
| TA3S | Timer A3 count start flag | | O | O |
| TA4S | Timer A4 count start flag | | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate if read. | | | – | – |

Clock prescaler reset flag




Symbol: CPSRF | Address: $0381_{16}$ | When reset: $0XXXXXXX_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate if read. | | | – | – |
| CPSR | Clock prescaler reset flag | 0 : No effect<br>1 : Reset<br>(The value is "0" when read) | O | O |

**Figure 1.79. Timer A-related registers (2)**



Downloaded from Arrow.com.

Up/down flag (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol: UDF
Address: $0384_{16}$
When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TA0UD | Timer A0 up/down flag | 0 : Down count<br>1 : Up count<br>This specification becomes valid when the up/down flag content is selected for up/down switching cause | O | O |
| TA1UD | Timer A1 up/down flag | | O | O |
| TA2UD | Timer A2 up/down flag | | O | O |
| TA3UD | Timer A3 up/down flag | | O | O |
| TA4UD | Timer A4 up/down flag | | O | O |
| TA2P | Timer A2 two-phase pulse signal processing select bit | 0 : Disabled<br>1 : Enabled<br>When not using the two-phase pulse signal processing function, set the select bit to "0" | – | O |
| TA3P | Timer A3 two-phase pulse signal processing select bit | | – | O |
| TA4P | Timer A4 two-phase pulse signal processing select bit | | – | O |

Note : Use MOV instruction to write to this register

One-shot start flag

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 0 | | | | | |

Symbol: ONSF
Address: $0382_{16}$
When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TA0OS | Timer A0 one-shot start flag | 0 : Invalid<br>1 : Timer start (Note 1) | O | O |
| TA1OS | Timer A1 one-shot start flag | | O | O |
| TA2OS | Timer A2 one-shot start flag | | O | O |
| TA3OS | Timer A3 one-shot start flag | | O | O |
| TA4OS | Timer A4 one-shot start flag | | O | O |
| Reserved | | Always set to "0" | O | O |
| TA0TGL | Timer A0 event/trigger select bit | b7 b6<br>0 0 : Input on $TA0_{IN}$ is selected (Notes 2, 3)<br>0 1 : Invalid<br>1 0 : TA4 overflow is selected<br>1 1 : TA1 overflow is selected | O | O |
| TA0TGH | | | O | O |

Note 1 : The value is "0" when read.
Note 2 : Set the corresponding port direction register to "0".
Note 3 : To start count in one-shot timer mode, do not use an extrenal trigger input.

**Figure 1.80. Timer A-related register (3)**



Downloaded from Arrow.com.

### Timer mode

In this mode, the timer counts an internally generated count source. Timer A in timer mode specifications are shown in Table 1.39. Figure 1.81 shows the Timer Ai mode register in timer mode.

**Table 1.39. Timer mode specifications**

| Item | Specification |
|---|---|
| Count source | f1, f8, f32, fc32 |
| Count operation | • Down count<br>• When the timer underflows, it loads the reload register contents before continuing counting |
| Divide ratio | 1/(n+1) n: Set value |
| Count start condition | Count start flag is set (= 1) |
| Count stop condition | Count start flag is reset (= 0) |
| Interrupt request generation timing | When the timer underflows |
| TAiIN pin function | Programmable I/O port or gate input. |
| TAiOUT pin function | Programmable I/O port or pulse output. |
| Read from timer | Count value can be read out by reading Timer Ai register |
| Write to timer | • When counting is stopped and a value is written to Timer Ai register, it is written to both the reload register and counter<br>• Whencountingis in progressand avalue iswritten to Timer Ai register, itis written only to the reload register (to be transferred to counter at the next reload time) |
| Select function | • Gate function-Counting can be started and stopped by TAiIN pin's input signal<br>• Pulse output function-Each time the timer underflows, the TAiOUT pin's polarity isreversed |

Timer Ai mode register (i = 0 to 4)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol: TAiMR (i=0 to 4) Address: $0396_{16}$ to $039A_{16}$ When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode | O | O |
| TMOD1 | | | O | O |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (TAiOUT pin is a pulse output pin) | O | O |
| MR1 | Gate function select bit | b4 b3<br>0 X : Gate funciton not available (TAiIN pin is a normal port pin) (Note 1)<br>1 0 : Timer counts only when TAiIN pin is held "L" (Note 2)<br>1 1 : Timer counts only when TAiIN pin is held "H" (Note 2) | O | O |
| MR2 | | | O | O |
| MR3 | 0 (Set to "0" in timer mode) | | O | O |
| TCK0 | Count source select bit | b7 b6<br>0 0 : f1<br>0 1 : f8<br>1 0 : f32<br>1 1 : fc32 | O | O |
| TCK1 | | | O | O |

Note 1: X value can be "0" or "1"
Note 2: Set the corresponding port direction register to "0".

**Figure 1.81. Timer Ai mode register in timer mode**



Downloaded from Arrow.com.

### Event counter mode

In this mode, the timer counts an external signal or an internal timer's overflow. Timers A0 and A1 can count a single-phase external signal. Timers A2, A3, and A4 can count a single-phase and a two-phase external signal. Table 1.40 lists timer specifications when counting a single-phase external signal. Table 1.41 lists timer specifications when counting a two-phase external signal. Figure 1.82 shows the Timer Ai mode register in event counter mode (excluding two-phase pulse signal processing). Figure 1.83 shows the Timer Ai mode register in event counter mode when using two-phase pulse signal processing.

**Table 1.40. Event counter mode specifications (excluding two-phase pulse signal)**

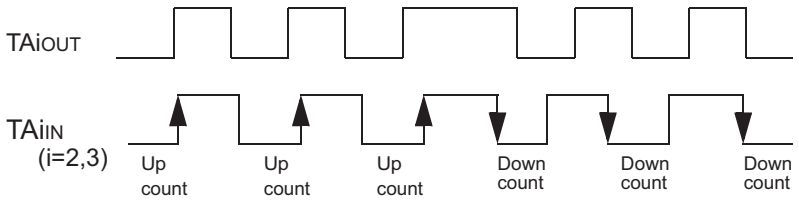
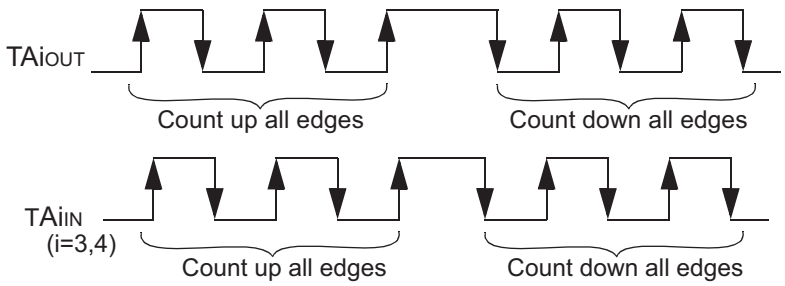
| Item | Specification |
|---|---|
| Count source | • External signals input to TAiIN pin (effective edge can be selected by software)<br>• TAj overflow |
| Count operation | • Up count or down count can be selected by external signal or software.<br>• When the timer overflows or underflows, it reloads the reload register contents before counting continues. (Note) |
| Divide ratio | $1/(FFFF_{16} - n + 1)$ for up count<br>1/(n + 1) for down count n: Set value |
| Count start condition | Count start flag is set (= 1) |
| Count stop condition | Count start flag is reset (= 0) |
| Interrupt request generation timing | The timer overflows or underflows |
| TAiIN pin function | Programmable I/O port or count source input |
| TAiOUT pin function | Programmable I/O port, pulse output, or up/down count select input. |
| Read from timer | Count value can be read out by reading Timer Ai register |
| Write to timer | • When counting is stopped and a value is written to Timer Ai register, it is written to both the reload register and counter<br>• Whencountingis in progressand avalue iswritten to Timer Ai register, itis written only to the reload register (to be transferred to the counter at the next reload time) |
| Select function | • Free-run count function<br>Even when the timer overflows or underflows, the reload register content is not reloaded.<br>• Pulse output function<br>Each time the timer overflows or underflows, the TAiOUT pin's polarity is reversed |

Note: This does not apply when the free-run function is selected



Downloaded from Arrow.com.

**Table 1.41. Timer specifications in event counter mode (when processing two-phase pulse signal)**

| Item | Specification |
|---|---|
| Count Source | •Two-phase pulse signals input to TAiIN or TAiOUT pin |
| Count operation | •Up count or down count can be selected by two-phase pulse signal<br>•When the timer overflows or underflows, the reload register content is loaded and the timer starts over again (Note 1) |
| Divide ratio | 1/ ($FFFF_{16}$ - n + 1) for up count<br>1/ (n+1) for down count n: Set value |
| Count start condition | Count start flag is set (=1) |
| Count stop condition | Count start flag is reset (=0) |
| Interrupt request generation timing | Timer overflow or underflows |
| TAiIN pin function | Two-phase pulse input |
| TAiOUT pin function | Two-phase pulse input |
| Read from timer | Count value can be read out by reading Timer A2, A3, or A4 register |
| Writer to timer | •When counting is stopped and a value is written to Timer A2, A3, or A4 register, it is written to both the reload register and counter<br>•When counting is in progress and a value is written to Timer A2, A3, or A4 register, it is written only to the reload register (to be transferred to the counter at the next reload time). |
| Select function (Note 2) | •Normal processing operation (Timer A2 and A3)<br>The timer counts up rising edges or counts down falling edges on the TAiIN pin when the input signal on the TAiOUT pin is "H"<br><br>•Multiply-by-4 processing operation (Timer A3 and Timer A4)<br>If the phase relationship is such that the TAiIN pin goes "H" when the input signal on the TAiOUT pin is "H", the timer counts up rising and falling edges on the TAiOUT and TAiIN pins.<br>If the phase relationship is such that the TAiIN pin goes "L" when the input signal on the TAiOUT pin is "H", the timer counts down rising and falling edges on the TAiOUT and TAiIN pins.<br> |

Note 1: This does not apply when the free-run function is selected.

Note 2: Timer A3 is selectable. Timer A2 is fixed to normal processing operation and Timer A4 is fixed to multiply-by-4 operation.



Downloaded from Arrow.com.

Timer Ai mode register (i = 0 to 4)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol: TAiMR (i=0 to 4)
Address: $0396_{16}$ to $039A_{16}$
When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 1 : Event counter mode (Note 1) | O | O |
| TMOD1 | | | O | O |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (TAiOUT pin is a pulse output pin) | O | O |
| MR1 | Count polarity select bit (Note 2) | 0 : Counts external signals falling edges<br>1 : Counts external signals rising edges | O | O |
| MR2 | Up/down switching cause select bit | 0 : Up/down flag data<br>1 : TAiOUT pin's input signal (Note 3) | O | O |
| MR3 | 0 (Set to "0" in event counter mode) | | O | O |
| TCK0 | Count operation type select bit | 0 : Reload type<br>1 : Free-run type | O | O |
| TCK1 | Two-phase pulse signal processing operation select bit (Note 4) | 0 (Set to "0" when not using two-phase pulse signal processing) | O | O |

Note 1: Count source is selected by the event/trigger select bit (addresses $0382_{16}$, $0383_{16}$) in event counter mode.
Note 2: This bit is valid only when counting an external signal.
Note 3: Set the corresponding port direction register to "0".
Note 4: This bit is valid for Timer A3 mode registers. Timer A2 is fixed to normal processing operation and Timer A4 is fixed to multiply-by-4 processing operation.

**Figure 1.82. Timer Ai mode register in event counter mode (not using two-phase processing)**

Timer Ai mode register (i = 0 to 4)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol: TAiMR (i=0 to 4)
Address: $0396_{16}$ to $039A_{16}$
When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 1 : Event counter mode (Note 1) | O | O |
| TMOD1 | | | O | O |
| MR0 | Pulse output function select bit | 0 (Set to "0" when using two-phase pulse signal processing) | O | O |
| MR1 | Count polarity select bit (Note 2) | 0 (Set to "0" when using two-phase pulse signal processing) | O | O |
| MR2 | Up/down switching cause select bit | 0 (Set to "1" when using two-phase pulse signal processing) | O | O |
| MR3 | 0 (Set to "0" in event counter mode) | | O | O |
| TCK0 | Count operation type select bit | 0 : Reload type<br>1 : Free-run type | O | O |
| TCK1 | Two-phase pulse signal processing operation select bit (Notes 3, 4) | 0 : Normal processing operation<br>1 : Multiply-by-4 operation | O | O |

Note 1: Count source is selected by the event/trigger select bit (addresses $0382_{16}$, $0383_{16}$) in event counter mode.
Note 2: This bit is valid only when counting an external signal.
Note 3: This bit is valid for Timer A3 mode registers. Timer A2 is fixed to normal processing operation and Timer A4 is fixed to multiply-by-4 processing operation.
Note 4: When performing two-phase pulse signal processing, make sure the two-phase pulse signal processing operation select bit (address $0384_{16}$) is set to "1". Also be sure to always set the event/trigger select bit (address $0383_{16}$) to "00".

**Figure 1.83. Timer Ai mode register in event counter mode (using two-phase processing)**



Downloaded from Arrow.com.

### One-shot timer mode

In this mode, the timer operates only once. Table 1.42 shows the timer specifications for Timer A in one-shot timer mode. When a trigger occurs, the timer starts counting down until it reaches $0000_{16}$. Figure 1.84 shows the Timer Ai mode register in one-shot timer mode.

**Table 1.42. Timer specifications in one-shot timer mode**

| Item | Specification |
|---|---|
| Count source | f1, f8, f32, fc32 |
| Count operation | • The timer counts down<br>• When the count reaches $0000_{16}$, the timer stops counting after reloading a new count<br>• If a trigger occurs when counting, the timer reloads a newcount and restarts counting |
| Divide ratio | 1/n n: Set value |
| Count start condition | • An external trigger is input<br>• The timer overflows<br>• The one-shot flag is set (=1) |
| Count stop condition | • A new count is reloaded after the count has reached $0000_{16}$<br>• The count start flag is reset (=0) |
| Interrupt request generation timing | The count reaches $0000_{16}$ |
| TAiIN pin function | Programmable I/O port or trigger input. |
| TAiOUT pin function | Programmable I/O port or pulse output. |
| Read from timer | When Timer Ai register is read, the value is indeterminate. |
| Write to timer | • When counting has stopped and a value is written to Timer Ai, it is also written to the reload register and counter.<br>• When counting is in progress and a value is written to Timer Ai, it is written to only the reload register (Transferred to the counter at next reload time) |

Timer Ai mode register (i = 0 to 4)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol: TAiMR (i=0 to 4) — Address: $0396_{16}$ to $039A_{16}$ — When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 0 : One shot timer mode | O | O |
| TMOD1 | | | O | O |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (TAiOUT pin is a pulse output pin) | O | O |
| MR1 | External trigger select bit (Note 1) | 0 : Falling edge of TAiIN input signal (Note 2)<br>1 : Rising edge of TAiIN input signal (Note 2) | O | O |
| MR2 | Trigger select bit | 0 : One-shot start flag is valid<br>1 : Selected by event/trigger select register | O | O |
| MR3 | 0 (Set to "0" in one-shot timer mode) | | O | O |
| TCK0 | Count operation type select bit | b7 b6<br>0 0 : f1<br>0 1 : f8<br>1 0 : f32<br>1 1 : fc32 | O | O |
| TCK1 | | | O | O |

Note 1: Valid only when the TAiIN pin is selected by the event trigger select bit (addresses $0382_{16}$, $0383_{16}$).
Note 2: Set the corresponding port direction register to "0".

**Figure 1.84. Timer Ai mode register in one-shot timer mode**



Downloaded from Arrow.com.

### Pulse width modulation (PWM) mode

In this mode, the timer outputs pulses of a given width in succession. Table 1.43 shows the timer specification for Timer in pulse width modulation mode. In this mode, the counter functions as either a 16-bit pulse width modulator or an 8-bit pulse width modulator. Figure 1.85 shows the Timer Ai mode register in pulse width modulation mode. Figure 1.86 shows an example of how a 16-bit pulse width modulator operates. Figure 1.87 shows an example of how an 8-bit pulse width modulator operates.

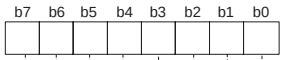
**Table 1.43. Timer specifications in pulse width modulation mode**

| Item | Specification |
|---|---|
| Count source | f1, f8, f32, fc32 |
| Count operation | • The timer counts down (operating as an 8-bit or a 16-bit pulse width modulator)<br>• The timer reloads a new count at a rising edge of the PWM pulse and continues counting<br>• The timer is not affected by a trigger that occurs when counting |
| 16-bit PWM | • High level width n/fi n=Set value<br>• Cycle time $(2^{16}-1)/fi$ fixed |
| 8-bit PWM | • High level width n X (m+1)/fi n: values set to Timer Ai register's high-order address<br>• Cycle time $(2^{8}-1)$ X (m+1)/fi m:values set to Timer Ai register's low-order address |
| Count start condition | • External trigger is input<br>• The timer overflows<br>• The count start flag is set (=1) |
| Count stop condition | The count start flag is reset (=0) |
| Interrupt request generation timing | PWM pulse goes "L" |
| TAiIN pin function | Programmable I/O port or trigger input |
| TAiOUT pin function | Pulse output Two-phase pulse input. |
| Read from timer | When Timer Ai is read, the value is indeterminate. |
| Write to timer | • When counting has stopped and a value is written to Timer Ai, it is written to both the reload register and counter.<br>• When counting is in progress and a value is written to Timer Ai, it is written to only the reload register (Transferred to the counter at next reload time) |



Downloaded from Arrow.com.

Timer Ai mode register (i = 0 to 4)




Symbol: TAiMR (i=0 to 4) | Address: $0396_{16}$ to $039A_{16}$ | When reset: $00000X00_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 1 : Pulse width modulator (PWM) mode | O | O |
| TMOD1 | | | O | O |
| MR0 | 1 ( Must always be "1" in PWM mode) | | O | O |
| MR1 | External trigger select bit (Note 1) | 0 : Falling edge of TAiIN input signal (Note 2)<br>1 : Rising edge of TAiIN input signal (Note 2) | O | O |
| MR2 | Trigger select bit | 0 : Count start flag is valid<br>1 : Selected by event/trigger select register | O | O |
| MR3 | 16/8 PWM mode select bit | 0 : Functions as a 16-bit PWM<br>1 : Functions as an 8-bit PWM | O | O |
| TCK0 | Count operation type select bit | b7 b6<br>0 0 : f1<br>0 1 : f8<br>1 0 : f32<br>1 1 : fc32 | O | O |
| TCK1 | | | O | O |

Note 1: Valid only when the $TAi_{IN}$ pin is selected by the event trigger select bit (addresses $0382_{16}$, $0383_{16}$).
Note 2: Set the corresponding port direction register to "0".

**Figure 1.85. Timer Ai mode register in pulse width modulation mode**

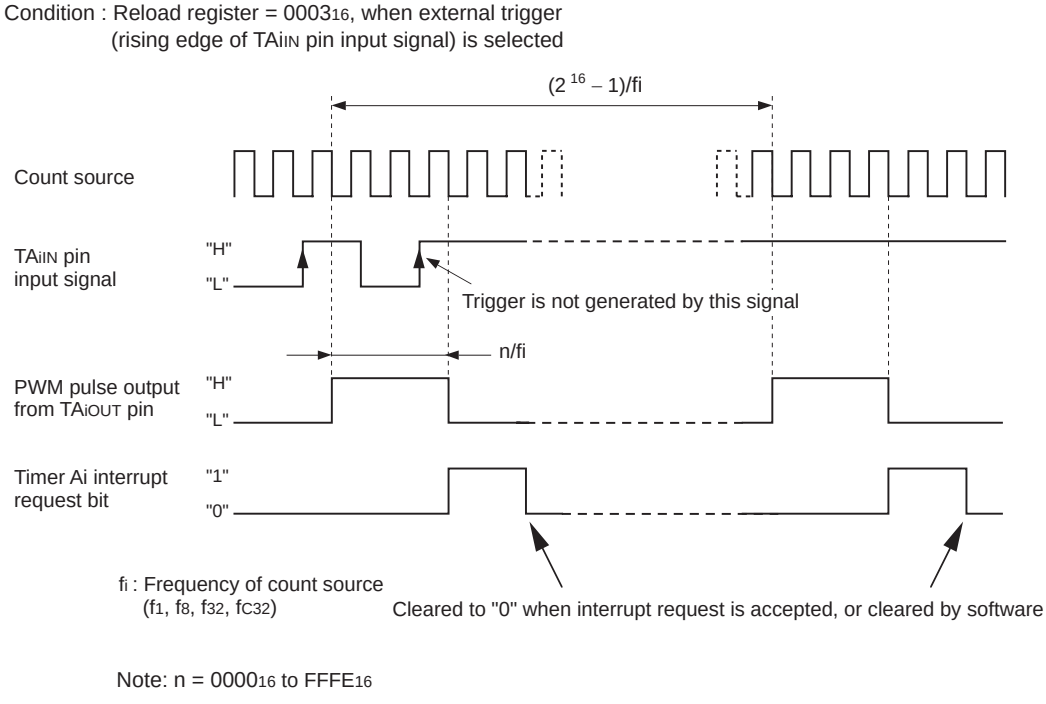



**Figure 1.86. Example of a 16-bit pulse width modulator operation**



Downloaded from Arrow.com.

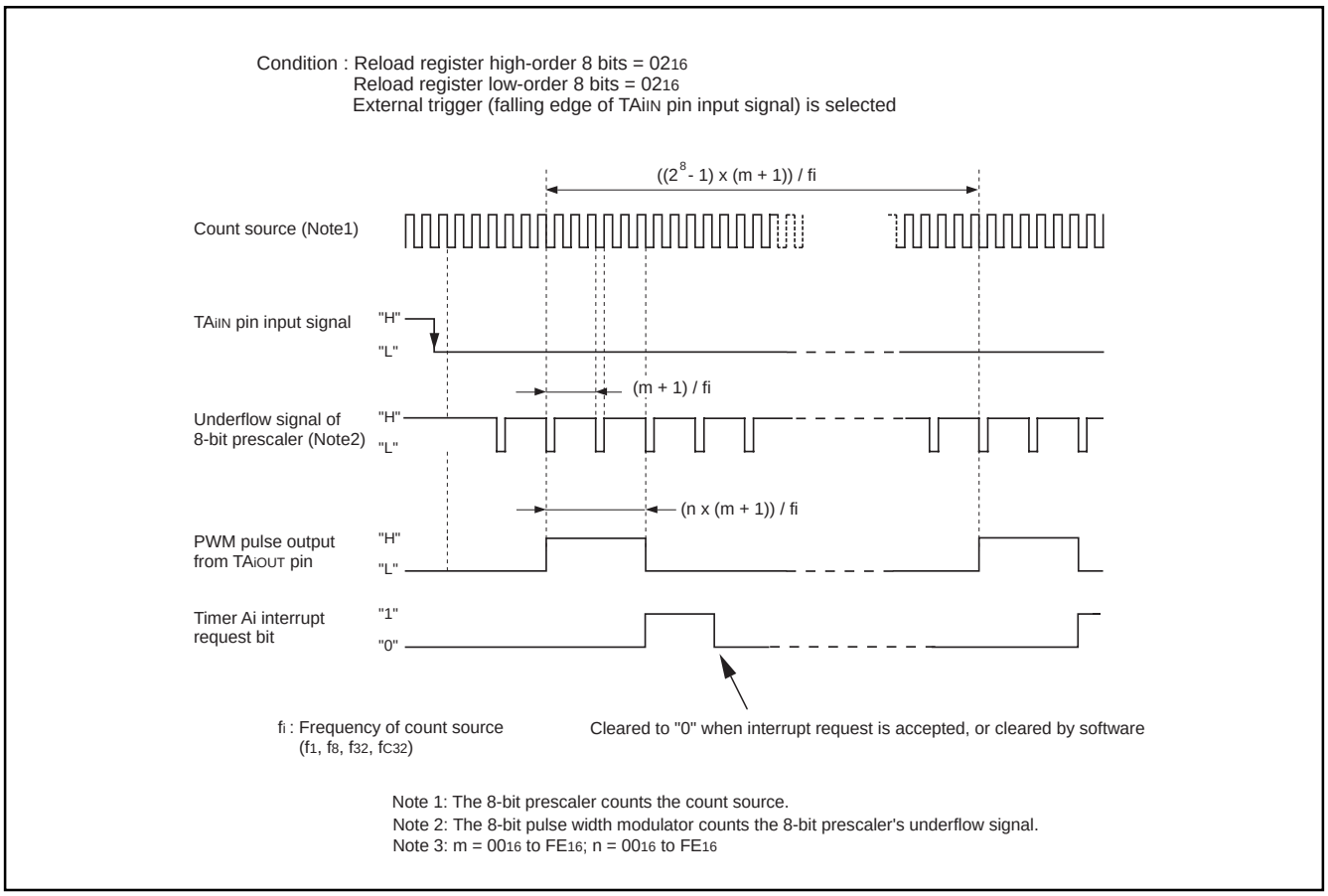



**Figure 1.87. Example of an 8-bit pulse width modulator operation**



Downloaded from Arrow.com.

## Precautions

### Timer mode

The value of the counter can be read, with arbitrary timing, by reading the Timer Ai register while a count is in progress.

Reading the Timer Ai register with the reload timing gets "$FFFF_{16}$".

After setting a value in the Timer Ai register, a proper value can be read with the counter stopped before it starts counting .

### Event counter mode

The value of the counter can be read, with arbitrary timing, by reading the Timer Ai register while a count is in progress. Reading the Timer Ai register with the reload timing gets "$FFFF_{16}$" by underflow or "$0000_{16}$" by overflow. After setting a value in the Timer Ai register, a proper value can be read with the counter stopped before it starts counting .

Reset the timer when counting has stopped in free run type.

If using "Free-Run type", the timer register contents may be unknown when counting begins. Set the timer value immediately after counting has started.

Example if the up/down count is not switched:

• Enable the "Reload" function and write to the timer register before counting begins.

• Rewrite the value to the timer register immediately after counting has started.

• If counting up, rewrite "$0000_{16}$" to the timer register.

• If counting down, rewrite "$FFFF_1$" to the timer register. This will cause the same operation as "Free-Run type".

Example if the up/down count is switched:

• Use the "Reload type" operation until the first count pulse is input.

• Switch to "Free-Run type".

### One-shot timer mode

Setting the count start flag to "0" while a count is in progress causes as the following:

• The counter stops counting and a content of reload register is reloaded.

• The $TAi_{OUT}$ pin outputs "L" level.

• The interrupt request generated and the Timer Ai interrupt request bit goes to"1".

The output from the one-shot timer synchronizes with the count source generated internally. Therefore, when an external trigger has been selected, a delay of one cycle of count source (maximum) occurs between the trigger input to the $TAi_{IN}$ pin and the one-shot timer output.



Downloaded from Arrow.com.

The Timer Ai interrupt request bit goes to "1" if the timer's operation mode is set using any of the following procedures:

• Selecting one-shot timer mode after reset.

• Changing operation mode from timer mode to one-shot timer mode.

• Changing operation mode from event counter mode to one-shot timer mode.

Therefore, to use Timer Ai interrupt (interrupt request bit), set Timer Ai interrupt request bit to "0" after the above listed changes have been made.

If a trigger occurs while a count is in progress, after the counter performs one down count following the reoccurrence of a trigger, the reload register contents are reloaded, and the count continues.

To generate a trigger while a count is in progress, generate the second trigger after a period longer than one cycle of the timer's count source after the previous trigger occurred.

**Pulse modulation mode**

The Timer Ai interrupt request bit becomes "1" if setting operation mode of the timer in compliance with any of the following procedures:

• Selecting PWM mode after reset.

• Changing operation mode from timer mode to PWM mode.

• Changing operation mode from event counter mode to PWM mode.

Therefore, to use Timer Ai interrupt (interrupt request bit), set Timer Ai interrupt request bit to "0" after the above listed changes have been made.

Setting the count start flag to "0" while PWM pulses are being output causes the counter to stop counting. If the TAiOUT pin is outputting an "H" level in this instance, the output level goes to "L", and the Timer Ai interrupt request bit goes to "1". If the TAiOUT pin is outputting an "L" level in this instance, the level does not change, and the Timer Ai interrupt request bit does not becomes "1".



Downloaded from Arrow.com.

## Serial Communication

Serial I/O is configured as four channels: UART0 to UART3.

UART0 to UART3 have an exclusive timer to generate a transfer clock so they can operate independently of one another.

Figure 1.88 shows the block diagram of UARTi (i = 0 to 3).

UARTi has two operation modes:

• Clock synchronous serial I/O mode

• Clock asynchronous serial I/O (UART) mode.

The contents of the serial I/O mode select bits (bits 0 to 2 at address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) determine whether UARTi is used as a clock synchronous serial I/O or as a UART. It also has the bus collision detection function that generates an interrupt request if the TxD pin and RxD pin are in different levels.

Figure 1.89 to Figure 1.93 show the UARTi related-registers.



Downloaded from Arrow.com.

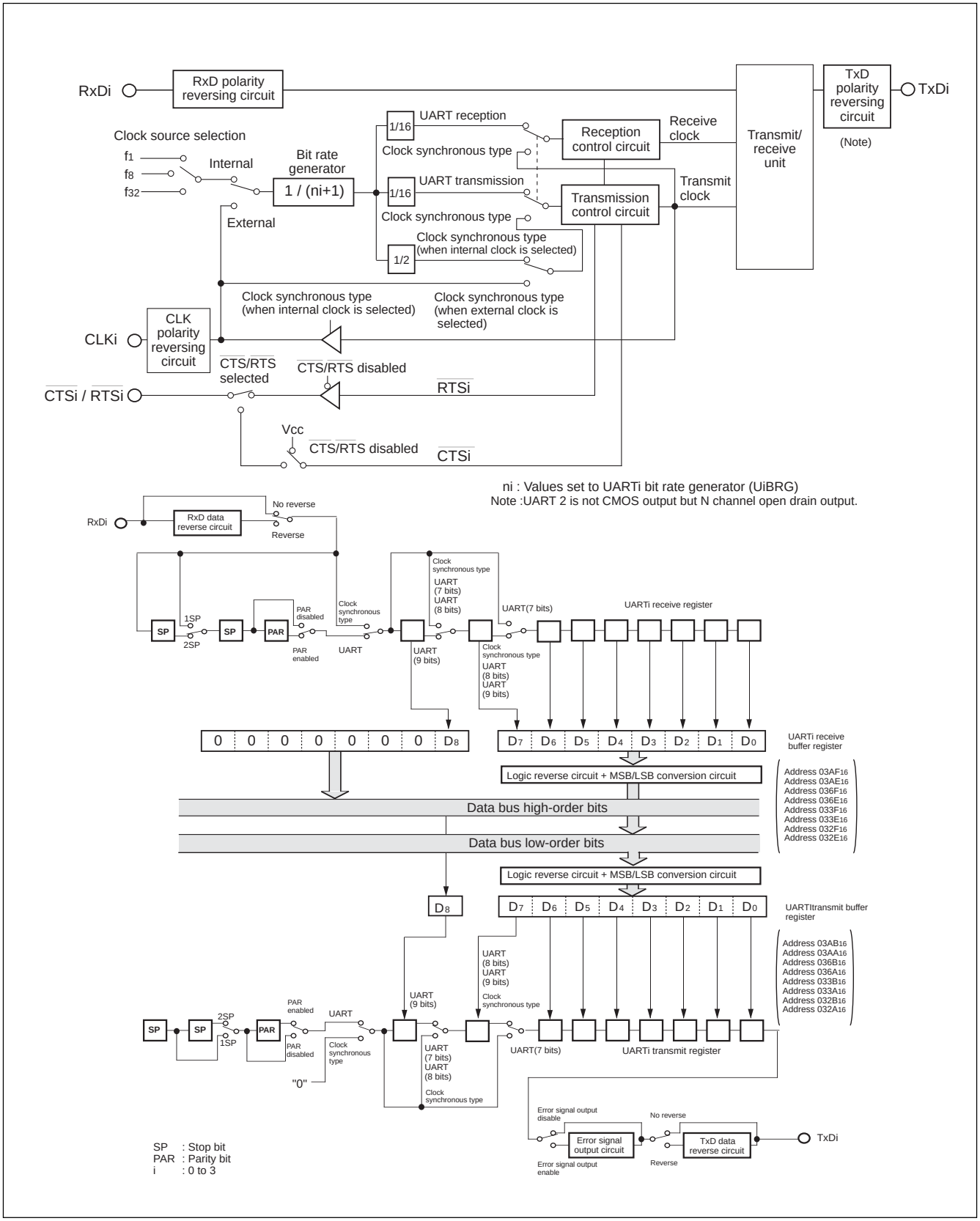
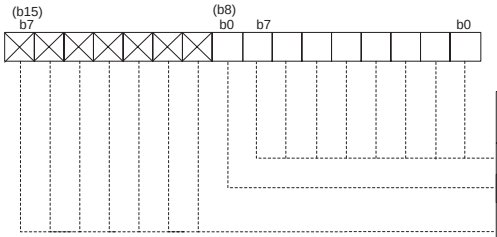



**Figure 1.88. UARTi block diagram**



Downloaded from Arrow.com.

UARTi transmit buffer register (i= 0 to 3) (Note)

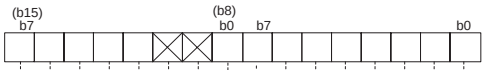



| Symbol | Address | When reset |
|---|---|---|
| U0TB | $03AB_{16}$, $03AA_{16}$ | Indeterminate |
| U1TB | $036B_{16}$, $036A_{16}$ | Indeterminate |
| U2TB | $033B_{16}$, $033A_{16}$ | Indeterminate |
| U3TB | $032B_{16}$, $032A_{16}$ | Indeterminate |

| Bit Symbol | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|
| ———— | Transmit data | Transmit data | X | O |
| ———— | ———— | Transmit data (9th bit) | X | O |
| Nothing is assigned. Write "0" when writing to these bits. The values are indeterminate when read. | | | – | – |

Note: Use MOV instruction to write to this register.

UARTi receive buffer register (i= 0 to 3)




| Symbol | Address | When reset |
|---|---|---|
| U0RB | $03AF_{16}$, $03AE_{16}$ | Indeterminate |
| U1RB | $036F_{16}$, $036E_{16}$ | Indeterminate |
| U2RB | $033F_{16}$, $033E_{16}$ | Indeterminate |
| U3RB | $032F_{16}$, $032E_{16}$ | Indeterminate |

| Bit Symbol | Bit name | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|---|
| ——— | ——— | Receive data | Receive data | O | X |
| ——— | ——— | ——— | Receive data (9th bit) | O | X |
| Nothing is assigned. Write "0" when writing to these bits. The values are indeterminate when read. | | | | – | – |
| ABT | Arbitration lost detecting flag (Note 1) | 0 : Not detected<br>1 : Detected | Invalid | O | O |
| OER | Overrun error flag (Note 2) | 0 : No overrrun error<br>1 : Overrun error | 0 : No overrun error<br>1 : Overrun error | O | X |
| FER | Framing error flag (Note 2) | Invalid | 0 : No framing error<br>1 : Framing error | O | X |
| PER | Parity error flag (Note 2) | Invalid | 0 : No parity error<br>1 : Parity error | O | X |
| SUM | Error sum flag (Note 2) | Invalid | 0 : No error<br>1 : Error | O | X |

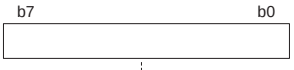
Note 1: Only "0" can be written to this bit.
Note 2: Bits 15 to 12 are set to "$0000_2$" when the serial I/O mode select bit (bits 0 to 2 at addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) are set to "$000_2$" or the receive enable bit is set to "0".
Bits 14 and 13 are also set to "0" when the lower byte of the UARTi receive buffer register (addresses $03AE_{16}$, $036E_{16}$, $033E_{16}$, $032E_{16}$) is read.

**Figure 1.89. Serial I/O-related registers (1)**



Downloaded from Arrow.com.

UARTi bit rate generator (i= 0 to 3) (Notes 1, 2)

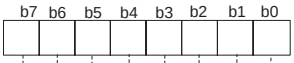



Symbol: UiBRG (i = 0 to 3)
Address: $03A9_{16}$, $0369_{16}$, $0339_{16}$, $0329_{16}$
When reset: Indeterminate

| Bit Symbol | Function | Values that can be set | R | W |
|---|---|---|---|---|
| | Assuming that set values = n, BRGi divides the count source by n + 1 | $00_{16}$ to $FF_{16}$ | X | O |

Note 1: Use MOV instruction to write to this register
Note 2: Write a value to this register while transmit/receive is stopped.

UARTi transmit/receive mode register (i= 0 to 3)




Symbol: UiMR (i = 0 to 3)
Address: $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$
When reset: $00_{16}$

| Bit Symbol | Bit Name | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit (Note 3) | b2 b1 b0<br>0 0 0: Serial I/O invalid<br>0 0 1: Serial I/O mode<br>0 1 0: $I^2C$ mode<br>Inhibited except in cases listed above | b2 b1 b0<br>0 0 0: Serial I/O invalid<br>1 0 0: Transfer data 7 bits long<br>1 0 1: Transfer data 8 bits long<br>1 1 0: Transfer data 9 bits long<br>Inhibited except in cases listed above | O | O |
| SMD1 | | | | O | O |
| SMD2 | | | | O | O |
| CKDIR | Internal/external clock select bit | 0 : Internal clock<br>1 : External clock (Note 1) | 0 : Internal clock<br>1 : External clock (Note 1) | O | O |
| STPS | Stop bit length select bit | Invalid | 0 : One stop bit<br>1 : Two stop bits | O | O |
| PRY | Odd/even parity select bit | Invalid | Valid when bit 6 = "1"<br>0 : Odd parity<br>1 : Even parity | O | O |
| PRYE | Parity enable bit | Invalid | 0 : Parity disabled<br>1 : Parity enabled | O | O |
| IOPOL | TxD, RxD input/ output polarity switch bit | 0 : Normal<br>1 : Reversed (Note 2) | | O | O |

Note 1: When $I^2C$ bus interface mode is selected, set the port direction register for the corresponding port (SCLi) to 0, or the port direction register to 1 and the port data register to 1. When a mode other than serial I/O mode is selected, set the port direction register for the corresponding port (CLKi) to 0.
Note 2: Normally set to "0".
Note 3: Set the corresponding port direction register to "0" when receiving.

**Figure 1.90. Serial I/O-related registers (2)**



Downloaded from Arrow.com.

UARTi transmit/receive control register 0 (i= 0 to 3)




Symbol: UiC0 (i = 0 to 3) | Address: $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$ | When reset: $08_{16}$

| Bit Symbol | Bit Name | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|---|
| CLK0 | BRG count source select bit | b1 b0<br>0 0 : f1 is selected<br>0 1 : f8 is selected<br>1 0 : f32 is selected<br>1 1 : Invalid | | O | O |
| CLK1 | | | | O | O |
| CRS | $\overline{CTS}/\overline{RTS}$ function select bit | Valid when bit 4 = "0"<br>0 : $\overline{CTS}$ is selected (Note 1)<br>1 : $\overline{RTS}$ is selected (Note 4) | | O | O |
| TXEPT | Transmit register empty flag | 0 : Data present in transmit register<br>1 : No data present in transmit register | | O | X |
| CRD | $\overline{CTS}/\overline{RTS}$ disable bit | 0 : $\overline{CTS}/\overline{RTS}$ function enabled<br>1 : $\overline{CTS}/\overline{RTS}$ function disabled | | O | O |
| NCH (Note 2) | Data output select bit | 0 : TxDi/SDAi and SCLi pin is CMOS output<br>1 : TxDi/SDAi and SCLi pin is N-channel open drain output | | O | O |
| CKPOL | CLK polarity select bit | 0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge<br>1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge | Set to "0" | O | O |
| UFORM | Transfer format select bit (Note 3) | 0 : LSB first<br>1 : MSB first | | O | O |

Note 1: Set the corresponding port direction register to "0".
Note 2: UART2 transfer pin (TxD2:P7$_0$ and SCL2:P7$_1$) is N-channel open drain output. It cannot be set to CMOS output.
Note 3: Valid only in clock synchronous serial I/O mode and 8-bit UART mode.
Note 4: The corresponding port register and port direction register are invalid.

UARTi transmit/receive control register 1 (i= 0 to 3)




Symbol: UiC1 (i = 0 to 3) | Address: $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ | When reset: $02_{16}$

| Bit Symbol | Bit Name | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|---|
| TE | Transmit enable bit | 0 : Transmit disabled<br>1 : Transmit enable | | O | O |
| TI | Transmit buffer empty flag | 0 : Data present in transmit buffer register<br>1 : No data present in transmit buffer register | | O | X |
| RE | Receive enable bit | 0 : Receive disabled<br>1 : Receive enabled | | O | O |
| RI | Receive complete flag | 0 : No data packet in receive buffer register<br>1 : Data packet in receive buffer register | | O | X |
| UiIRS | UARTi transmit interrupt cause select bit | 0 : Transmit buffer empty (TI =1)<br>1 : Transmit buffer completed ( TXEPT =1) | | O | O |
| UiRRM | UARTi continuous receive mode enable bit | 0 : Continuous receive mode disabled<br>1 : Continuous receive mode enabled | Set to "0" | O | O |
| UiLCH | Data logic select bit | 0 : No reverse<br>1 : Reverse | | O | O |
| UiERE | Error signal output enable bit | Set to "0" | 0 : Output disabled<br>1 : Output enabled | O | O |

**Figure 1.91. Serial I/O-related registers (3)**



Downloaded from Arrow.com.

UARTi special mode register (i= 0 to 3)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: UiSMR (i = 0 to 3)

Address: $03A7_{16}$, $0367_{16}$, $0337_{16}$, $0327_{16}$

When reset: $00_{16}$

| Bit Symbol | Bit Name | Function (clock synchronous serial I/O mode) | Function (UART mode) | R | W |
|---|---|---|---|---|---|
| IICM | $I^2C$ mode select bit | 0 : Normal mode<br>1 : $I^2C$ mode | Set to "0" | O | O |
| ABC | Arbitration lost detecting flag control bit | 0 : Update per bit<br>1 : Update per byte | Set to "0" | O | O |
| BBS | Bus busy flag | 0 : STOP detected<br>1 : START detected | Set to "0" (Note 1) | O | O |
| LSYN | SCLL sync output enable bit | 0 : Disabled<br>1 : Enabled (Note 3) | Set to "0" | O | O |
| ABSCS | Bus collision detect sampling clock select bit | Set to "0" | 0 : Rising edge of transfer clock<br>1 : Timer Ai underflow signal (Note 2) | O | O |
| ACSE | Auto-clear function select bit of transmit enable bit | Set to "0" | 0 : No auto clear function<br>1 : Auto clear when bus occurs | O | O |
| SSS | Transmit start condition select bit | Set to "0" | 0 : Ordinary<br>1 : Falling edge of RxDi | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is indeterminate if read. | | | | – | – |

Note 1: Only "0" may be written
Note 2: UART0: Timer A3 underflow signal, UART1: Timer A4 underlfow signal, UART2 :Timer A0 underflow signal, UART3: Timer A3 underflow signal
Note 3: Set to "0" in normal mode (IICM="0")

UARTi special mode register 2 (i= 0 to 3)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: UiSMR2 (i = 0 to 3)

Address: $03A6_{16}$, $0366_{16}$, $0336_{16}$, $0326_{16}$

When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| IICM2 | $I^2C$ mode select bit 2 | 0 : NACK/ACK interrupt (DMA source-ACK) Transfer to receive buffer at the rising edge of last bit of receive clock. Receive interrupt occurs at the rising edge of last bit of receive clock.<br>1 : UART transfer/receive interrupt (DMA source-UART receive) Transfer to receive buffer at the falling edge of last bit of receive clock. Receive interrupt occurs at the falling edge of last bit of receive clock | O | O |
| CSC | Clock synchronous bit | 0 : Disabled<br>1 : Enabled | O | O |
| SWC | SCL wait output bit (Note) | 0 : Disabled<br>1 : Enabled | O | O |
| ALS | SDA output stop bit | 0 : Disabled<br>1 : Enabled | O | O |
| STC | UARTi initialize bit (Note) | 0 : Disabled<br>1 : Enabled | O | O |
| SWC2 | SCL wait output bit 2 (Note) | 0 : UARTi clock<br>1 : output | O | O |
| SDHI | SDA output inhibit bit | 0 : Disabled<br>1 : Enabled (high impedance) | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is indeterminate if read. | | | – | – |

Note: These bits are unavailable when SCLi is external clock.

**Figure 1.92. Serial I/O-related registers (4)**



Downloaded from Arrow.com.

UARTi special mode register 3 (i= 0 to 3)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: UiSMR3 (i = 0 to 3)

Address: $03A5_{16}$, $0365_{16}$, $0335_{16}$, $0325_{16}$

When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| SSE | $\overline{SS}$ port function enable bit (Note 1) | 0 : $\overline{SS}$ function disabled<br>1 : $\overline{SS}$ function enabled | O | O |
| CKPH | Clock phase set bit | 0 : No clock delay<br>1 : Clock delay | O | O |
| DINC | Serial input port set bit | 0 : Select TxDi and RxDi (master mode)<br>1 : Select STxDi and SRxDi (slave mode) | O | O |
| NODC | Clock output select bit | 0 : CLKi is CMOS output<br>1 : CLKi is N-channel open drain output | O | O |
| ERR | Fault error flag | 0 : No fault error<br>1 : Fault error (Note 2) | O | O |
| DL0 | SDA (TxDi) digital delay time set bit (Notes 3,4) | b7 b6 b5<br>0 0 0 : No delay<br>0 0 1 : 1 to 2-cycle of UiBRG count source<br>0 1 0 : 2 to 3-cycle of UiBRG count source<br>0 1 1 : 3 to 4-cycle of UiBRG count source<br>1 0 0 : 4 to 5-cycle of UiBRG count source<br>1 0 1 : 5 to 6-cycle of UiBRG count source<br>1 1 0 : 6 to 7-cycle of UiBRG count source<br>1 1 1 : 7 to 8-cycle of UiBRG count source | O | O |
| DL1 | | | O | O |
| DL2 | | | O | O |

Note 1: Set $\overline{SS}$ function after setting $\overline{CTS/RTS}$ disable bit (bit 4 of UARTi transfer/receive control register 0) to "1".
Note 2: Only "0" may be written.
Note 3: These bits are used for SDAi (TxDi) output digital delay when using UARTi for $I^2C$ interface. Otherwise, set these to "000".
Note 4: The amount of delay varies with the load on SCLi and SDAi pins. Also, when external clock is selected, delay is increased by approximately 100ns.

UARTi special mode register 4 (i= 0 to 3)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol: UiSMR4 (i = 0 to 3)

Address: $03A4_{16}$, $0364_{16}$, $0334_{16}$, $0324_{16}$

When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| STAREQ | Start condition generate bit (Note 1) | 0 : Clear<br>1 : Start | O | O |
| RSTAREQ | Restart condition generate bit (Note 1) | 0 : Clear<br>1 : Start | O | O |
| STPREQ | Stop condition generate bit (Note 1) | 0 : Clear<br>1 : Start | O | O |
| STSPSEL | SCL, SDA output select bit | 0 : Ordinal block<br>1 : Start/stop condition generate block | O | O |
| ACKD | ACK data bit | 0 : ACK<br>1 : NACK | O | O |
| ACKC | ACK data output enable bit | 0 : SI /O data output<br>1 : ACKD output | O | O |
| SCLHI | SCL output stop enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| SWC9 | SCL waitt output bit 3 (Note 2) | 0 : SCL "L" hold disabled<br>1 : SCL "L" hold enabled | O | O |

Note 1: These bits automatically become "0" when a start condition is generated.
Note 2: This bit is unavailable when SCLi is external clock.

**Figure 1.93. Serial I/O-related registers (5)**



Downloaded from Arrow.com.

### Clock synchronous serial I/O mode

The clock synchronous serial I/O mode uses a transfer clock to transmit and receive data. Table 1.44 list the specifications of the clock synchronous serial I/O mode.

**Table 1.44. Clock synchronous serial I/O mode specifications**

| Item | Specification |
|---|---|
| Transfer data format | Transfer data length: 8-bits |
| Transfer clock | • When internal clock is selected (bit 3 at address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ ="0"):<br>fi/2(m+1) (Note 1) fi = f1, f8, f32<br>• When external clock is selected (bit 3 at $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "1"):<br>Input from CLKi pin |
| Transmission/reception control | $\overline{CTS}$ function/$\overline{RTS}$ function/$\overline{CTS}$, $\overline{RTS}$ function not used |
| Transmission start condition | • To start transfer, the following criteria must be met:<br>-Transmit enable bit (bit 0 at $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1"<br>-Transmit buffer empty flag (bit 1 at $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "0"<br>-When $\overline{CTS}$ function selected, $\overline{CTS}$ input level = "L"<br>• Furthermore, if external clock is selected, the following requirements must also be met:<br>-CLKi polarity select bit (bit 6 at $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$)= "0": CLKi input level = "H"<br>-CLKi polarity select bit (bit 6 at $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$)= "1": CLKi input level = "L" |
| Receive start condition | • To start reception, the following requirement must be met:<br>-Receive enable bit (bit 2 at $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1"<br>-Transmit enable bit (bit 0 at $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1"<br>-Transmit buffer empty flag (bit 1 at $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "0"<br>• Furthermore, if external clock is selected, the following requirement must be met:<br>-CLKi polarity select bit (bit 6 at $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$)= "0": CLKi input level = "H"<br>-CLKi polarity select bit (bit 6 at $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$)= "1": CLKi input level = "L" |
| Interrupt request generation timing | • When transmitting:<br>-Transmit interrupt cause select bit (bit 4 at $3AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "0": Interrupt requested when data transfer from UARTi transfer buffer register to UARTi transmit register is complete<br>-Transmit interrupt cause select bit (bit 4 at $3AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1": Interrupt requested when data transmission from UARTi transmit register is complete<br>• When receiving:<br>-Interrupt requested when data transfer from UARTi receive register to UARTi receive buffer register is completed. |
| Error detection | • Overrun error (Note 2)<br>This error occurs if the serial I/O starts receiving the next data and receives the 7th bit of the next data before reading the UiRB register. |
| Select function | • CLK polarity selection<br>Whether transmit data is output/input at the rising edge or falling edge or the transfer clock can be selected<br>• LSB first/MSB first selection<br>Whether transmission/reception begins with bit 0 or bit 7 can be selected<br>• Continuous receive mode selection<br>Reception is enabled simultaneously by a read form the receive buffer register<br>• Switching serial data log<br>Reverse data when writing to the transmission buffer register or reading the reception buffer register can selected<br>• TxD, RxD, I/O polarity reverse<br>This function reverses the TxD port output and RxD port input. All I/O data level is reversed. |

Note 1: "m" denotes the value $00_{16}$ to $FF_{16}$ that is set in the UART bit rate generator.

Note 2: If an overrun error occurs, the value of the UiRB register will be indeterminate. The IR bit of the SiRIC register does not change.



Downloaded from Arrow.com.

Table 1.45 lists the functions of the input/output pins during clock synchronous serial I/O mode. Note that for a period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs a "H". (If the N-channel open drain is selected, this pin is in floating state.)

Figure 1.94. shows the typical transmit receive timings in clock synchronous serial I/O mode.

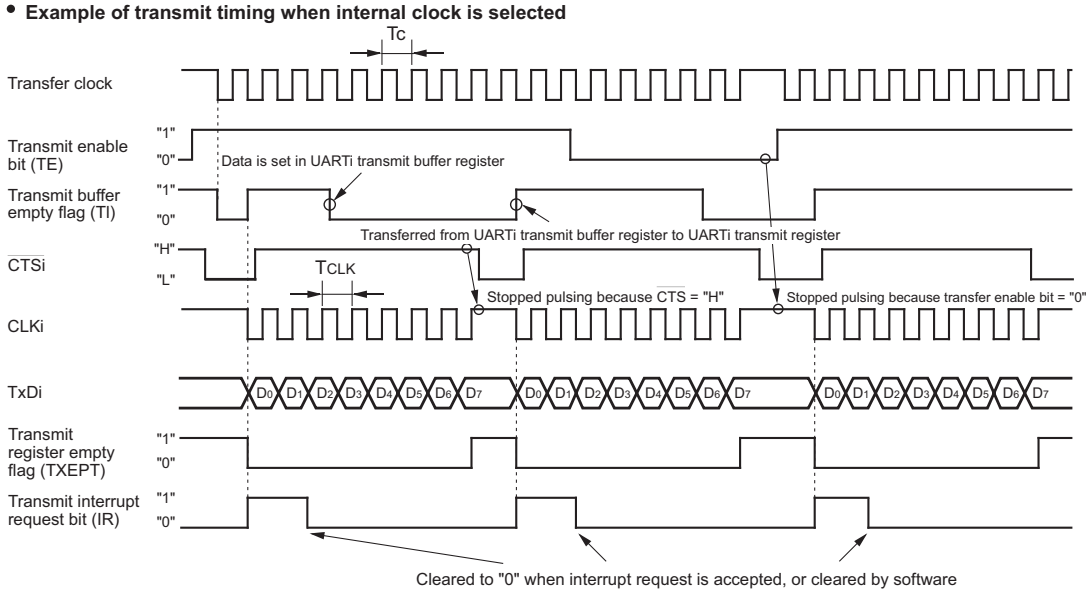
**Table 1.45. Input/output pin functions in clock synchronous serial I/O mode**

| Pin name | Function | Method of selection |
|---|---|---|
| TxDi ($P6_3$, $P6_7$, $P7_0$, $P7_4$) | Serial data output | |
| RxDi ($P6_2$, $P6_6$, $P7_1$, $P7_5$) | Serial data input | Port $P6_2$, $P6_6$, $P7_1$ and $P7_5$ direction register (bits 2 and 6 at address $03EE_{16}$, bit 1 and 5 at address $03EF_{16}$) = "0". Can be used as an input port when performing transmission only. |
| CLKi ($P6_1$, $P6_5$, $P7_2$, $P7_6$) | Programmable I/O port | Internal/external clock select bit (bit 3 at addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) = "0" |
| | Transfer clock input | Internal/external clock select bit (bit 3 at addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) = "1"<br>Port $P6_1$, $P6_5$, $P7_2$ and $P7_6$ direction register (bits 1 and 5 at address $03EE_{16}$, bit 2 and 6 at address $03EF_{16}$) = "0" |
| $\overline{CTS}i/\overline{RTS}i$ ($P6_0$, $P6_4$, $P7_3$, $P7_7$) | $\overline{CTS}$ input | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "0"<br>$\overline{CTS}/\overline{RTS}$ function select bit (bit 2 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "0"<br>Port $P6_0$, $P6_4$, $P7_3$ and $P7_7$ direction register (bits 0 and 4 at address $03EE_{16}$, bits 3 and 7 at address $03EF_{16}$) = "0" |
| | $\overline{RTS}$ output | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at addresses $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{166}$) = "0"<br>$\overline{CTS}/\overline{RTS}$ function select bit (bit 2 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "1" |
| | Programmable I/O port | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "1" |



Downloaded from Arrow.com.

• **Example of transmit timing when internal clock is selected**

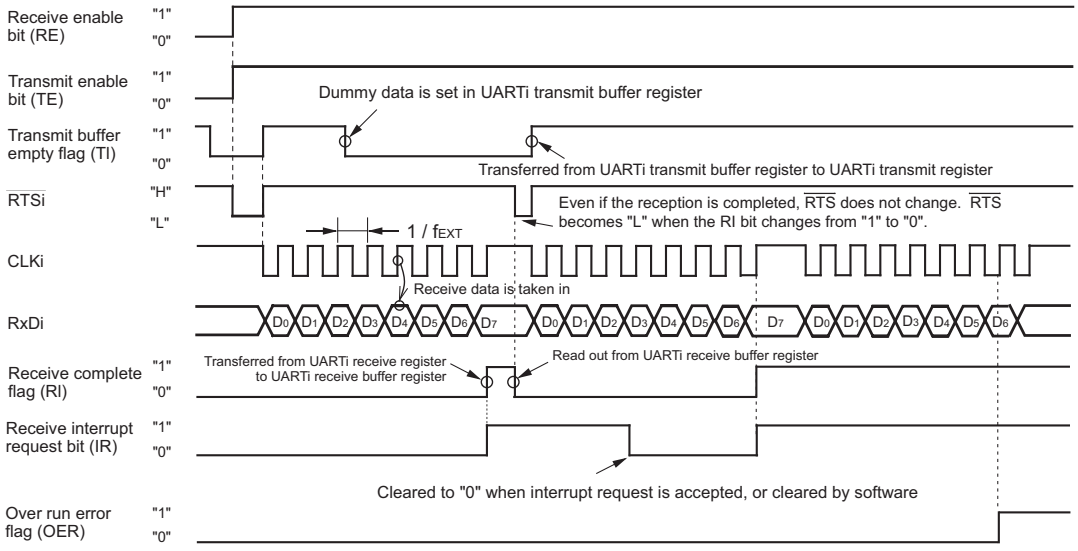



Shown in ( ) are bit symbols.

The above timing applies to the following settings:
- Internal clock is selected.
- CTS function is selected.
- CLK polarity select bit = "0".
- Transmit interrupt cause select bit = "0".

Tc = TCLK = 2(m + 1) / fi
fi : frequency of BRGi count source (f1, f8, f32)
m : value set to BRGi

• **Example of receive timing when external clock is selected**




Shown in ( ) are bit symbols.

The above timing applies to the following settings:
- External clock is selected.
- RTS function is selected.
- CLK polarity select bit = "0".

fEXT: frequency of external clock

The following conditions are met when the CLKi input before data reception = "H"
- Transmit enable bit →"1"
- Receive enable bit →"1"
- Dummy data write to UARTi transmit buffer register

**Figure 1.94. Typical transmit/receive timing in clock synchronous serial I/O mode**



Downloaded from Arrow.com.

### Polarity select function

As shown in Figure 1.95 the CLK polarity select bit (bit 6 at addresses 03AC16, 036C16, 033C16, 032C16) allows selection of the polarity of the transfer clock.

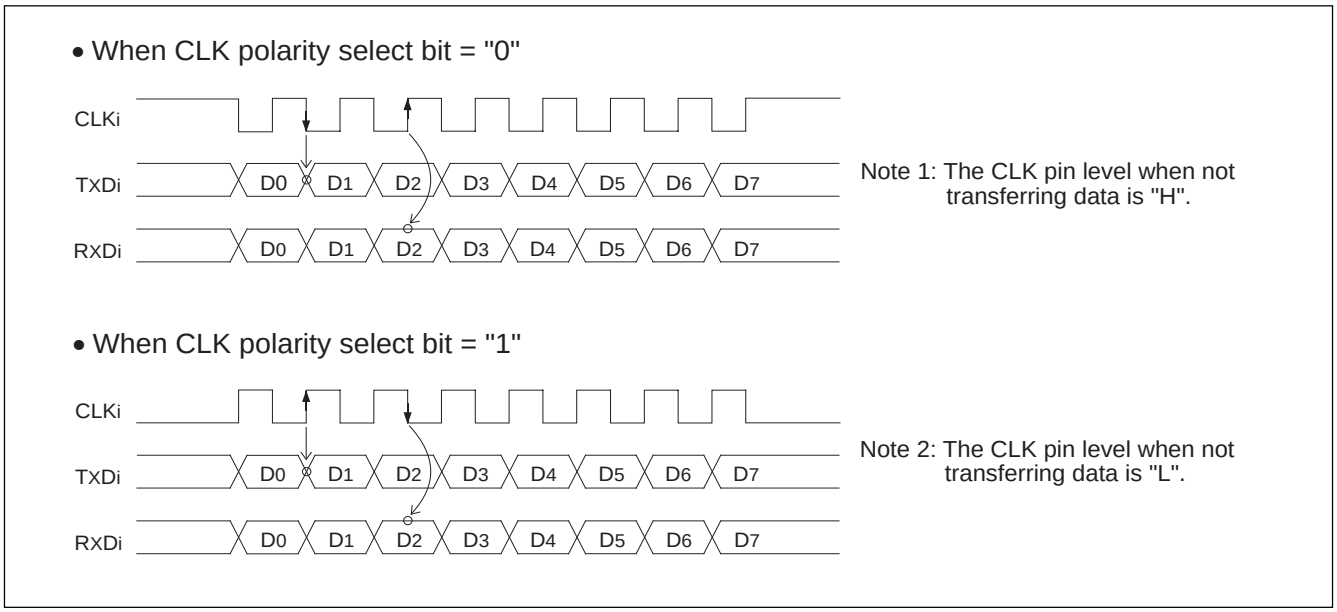



**Figure 1.95. Transfer clock polarity**

### LSB first/MSB first select function

As shown in Figure 1.96, when the transfer format select bit (bit 7 at addresses 03AC16, 036C16, 033C16, 032C16) = "0", the transfer format is "LSB first"; when the bit = "1", the transfer format is "MSB first".

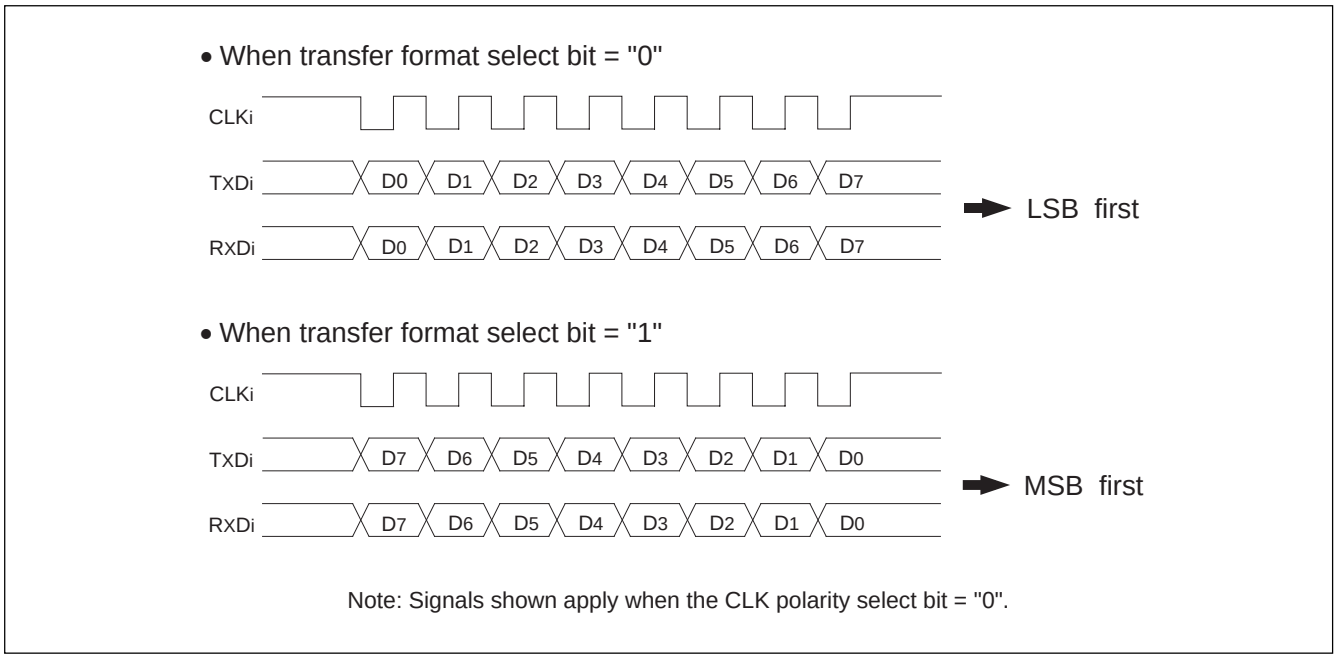



**Figure 1.96. Transfer format**



Downloaded from Arrow.com.

**Continuous receive function**

If the continuous receive mode enable bit (bit 5 at address 03AD16, 036D16, 033D16, 032D16) is set to "1", the unit is placed in continuous receive mode. In this mode, when the receive buffer is read out, the unit simultaneously goes to a receive enable state without having to set dummy data back to the transmit buffer register again.

**Serial data logic switch function**

When the data logic select bit (bit 6 at address 03AD16, 036D16, 033D16, 032D16) = "1", and writing to the transmit buffer register or reading from the receive buffer register, data are inverted. Figure 1.97 shows the example of serial data logic switch timing.

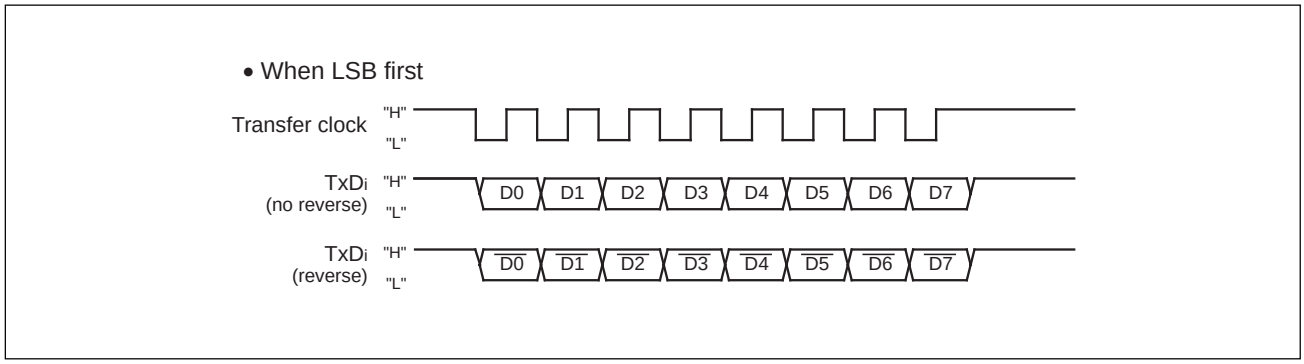



**Figure 1.97. Serial data logic switch timing**



Downloaded from Arrow.com.

### Clock asynchronous serial I/O (UART) mode

UART mode allows transmitting and receiving data after setting the desired transfer rate and transfer data format. Table 1.46 lists the specifications of the UART mode.

**Table 1.46. Specifications of clock asynchronous serial I/O mode**

| Item | Specification |
|---|---|
| Transfer data format | • Character bit (transfer data): 7 bits, 8 bits, or 9 bits as selected<br>• Start bit: 1 bit<br>• Parity bit: odd, even, or neither is selected<br>• Stop bit: 1 bit or 2 bits as selected |
| Transfer clock | • When internal clock is selected (bit 3 at address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "0"): fi/16(m+1) (Note 1) fi = f1, f8, f32<br>• When external clock is selected (bit 3 at address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "1"): fEXT/16/(m+1) (Notes 1, 2) |
| Transmission/reception control | $\overline{CTS}$ function, $\overline{RTS}$ function, $\overline{CTS}/\overline{RTS}$ function chosen to be invalid |
| Transmission start condition | • To start transmission, the following requirements must be met:<br>-Transmit enable bit (bit 0 at addresses $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1"<br>-Transmit buffer empty flag (bit 1 at address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "0"<br>-When $\overline{CTS}$ function is selected $\overline{CTS}$ input level = "1" |
| Receive start condition | • To start receive, the following conditions must be met:<br>-Receive enable bit (bit 2 at addresses $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1"<br>-Start bit detection |
| Interrupt request generation timing | • When transmitting<br>-Transmit interrupt cause select bits (bit 4 at address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "0": Interrupts requested when data transfer from UARTi transfer buffer register to UARTi transmit register is complete.<br>-Transmit interrupt cause select bits (bit 4 at address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$) = "1": Interrupts requested when data transmission from UARTi transfer register is complete.<br>• When receiving<br>-Interrupts requested when data transfer from UARTi receive register to UARTi receive buffer register is complete. |
| Error detection | • Overrun error (Note 3)<br>This error occurs if the serial I/O starts receiving the next data and receives the 7th bit of the next data before reading the UiRB register.<br>• Framing error<br>This error occurs when the number of stop bits set is not detected<br>• Parity error<br>If parity is enabled this error occurs when the number of "1"s in parity and character bits does not match the number of "1"s set<br>• Error sum flag<br>This flag is set (=1) when any overrun, framing, and parity error occurs |
| Select function | • Serial data logic switch<br>This function reverses the logic of transferred data. Start bit, parity bit and stop bit are not reversed.<br>• TxD, RxD I/O polarity switch<br>This function reverses the TxD port output and RxD port input. All I/O data levels are reversed. |

Note 1: 'm' denotes the value $00_{16}$ to $FF_{16}$ that is set to the UARTi bit rate generator.
Note 2: fEXT is input from the CLKi pin.
Note 3: If an overrun error occurs, the value of the UiRB register will be indeterminate. The IR bit of the SiRIC register does not change.



Downloaded from Arrow.com.

Table 1.47 lists the functions of the input/output pins during UART mode. Note that the period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs an "H". (If the N-channel open drain is selected, this pin is in floating state.) Figure 1.98 shows typical transmit timings in UART mode. Figure 1.99 shows typical receive timings in UART mode.

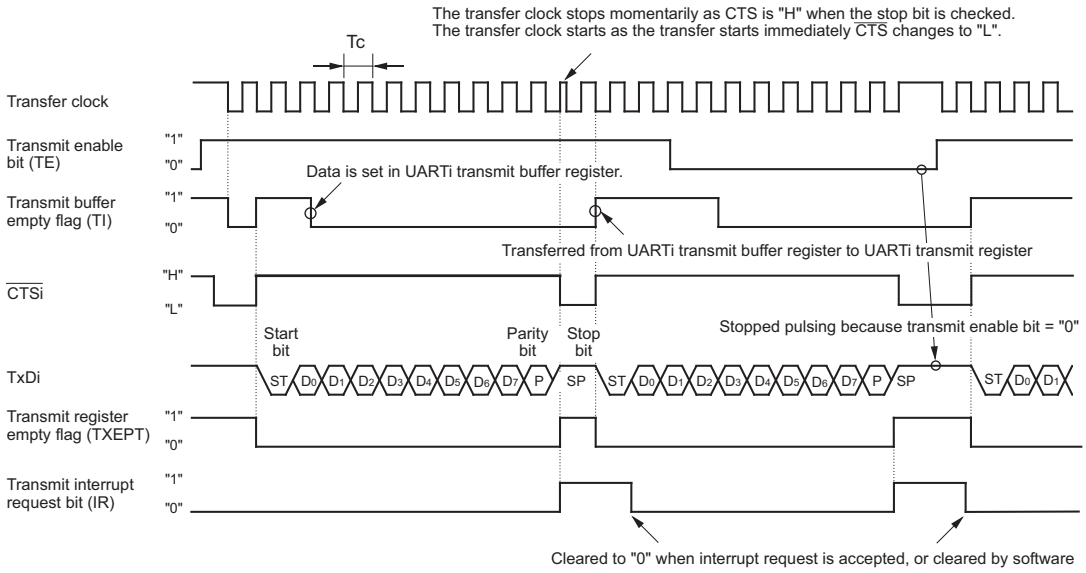
**Table 1.47. Input/output pin functions in UART mode**

| Pin name | Function | Method of selection |
|---|---|---|
| TxDi ($P6_3$, $P6_7$, $P7_0$, $P7_4$) | Serial data output | |
| RxDi ($P6_2$, $P6_6$, $P7_1$, $P7_5$) | Serial data input | Port $P6_2$, $P6_6$, $P7_1$ and $P7_5$ direction register (bits 2 and 6 at address $03EE_{16}$, bit 1 and 5 at address $03EF_{16}$) = "0". Can be used as an input port when performing transmission only. |
| CLKi ($P6_1$, $P6_5$, $P7_2$, $P7_6$) | Programmable I/O port | Internal/external clock select bit (bit 3 at addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) = "0" |
| | Transfer clock input | Internal/external clock select bit (bit 3 at addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$) = "1"<br>Port $P6_1$, $P6_5$, $P7_2$ and $P7_6$ direction register (bits 1 and 5 at address $03EE_{16}$, bit 2 and 6 at address $03EF_{16}$) = "0" |
| $\overline{CTS}i/\overline{RTS}i$ ($P6_0$, $P6_4$, $P7_3$, $P7_7$) | $\overline{CTS}$ input | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "0"<br>$\overline{CTS}/\overline{RTS}$ function select bit (bit 2 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "0"<br>Port $P6_0$, $P6_4$, $P7_3$ and $P7_7$ direction register (bits 0 and 4 at address $03EE_{16}$, bits 3 and 7 at address $03EF_{16}$) = "0" |
| | $\overline{RTS}$ output | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at addresses $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{166}$) = "0"<br>$\overline{CTS}/\overline{RTS}$ function select bit (bit 2 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "1" |
| | Programmable I/O port | $\overline{CTS}/\overline{RTS}$ disable bit (bit 4 at address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$) = "1" |



Downloaded from Arrow.com.

- **Example of transmit timing when transfer data is 8 bits long (parity enabled, one stop bit)**

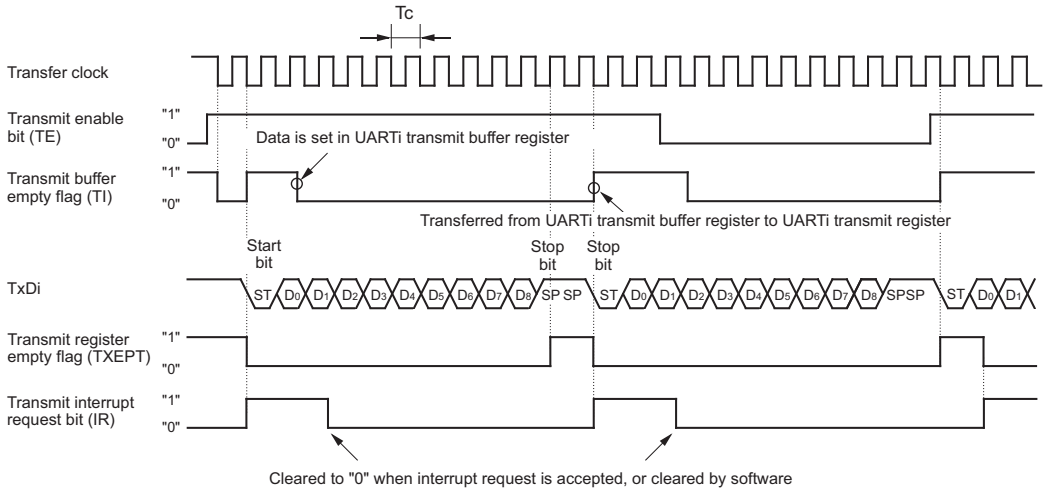



Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is enabled.
- One stop bit.
- CTS function is selected.
- Transmit interrupt cause select bit = "1".

Tc = 16 (m + 1) / fi or 16 (m + 1) / fEXT
fi : frequency of BRGi count source (f1, f8, f32)
fEXT : frequency of BRGi count source (external clock)
m : value set to BRGi

- **Example of transmit timing when transfer data is 9 bits long (parity disabled, two stop bits)**




Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is disabled.
- Two stop bits.
- TCS function is disabled.
- Transmit interrupt cause select bit = "0".

Tc = 16 (m + 1) / fi or 16 (m + 1) / fEXT
fi : frequency of BRGi count source (f1, f8, f32)
fEXT : frequency of BRGi count source (external clock)
m : value set to BRGi

**Figure 1.98. Typical transmit timings in UART mode**



Downloaded from Arrow.com.

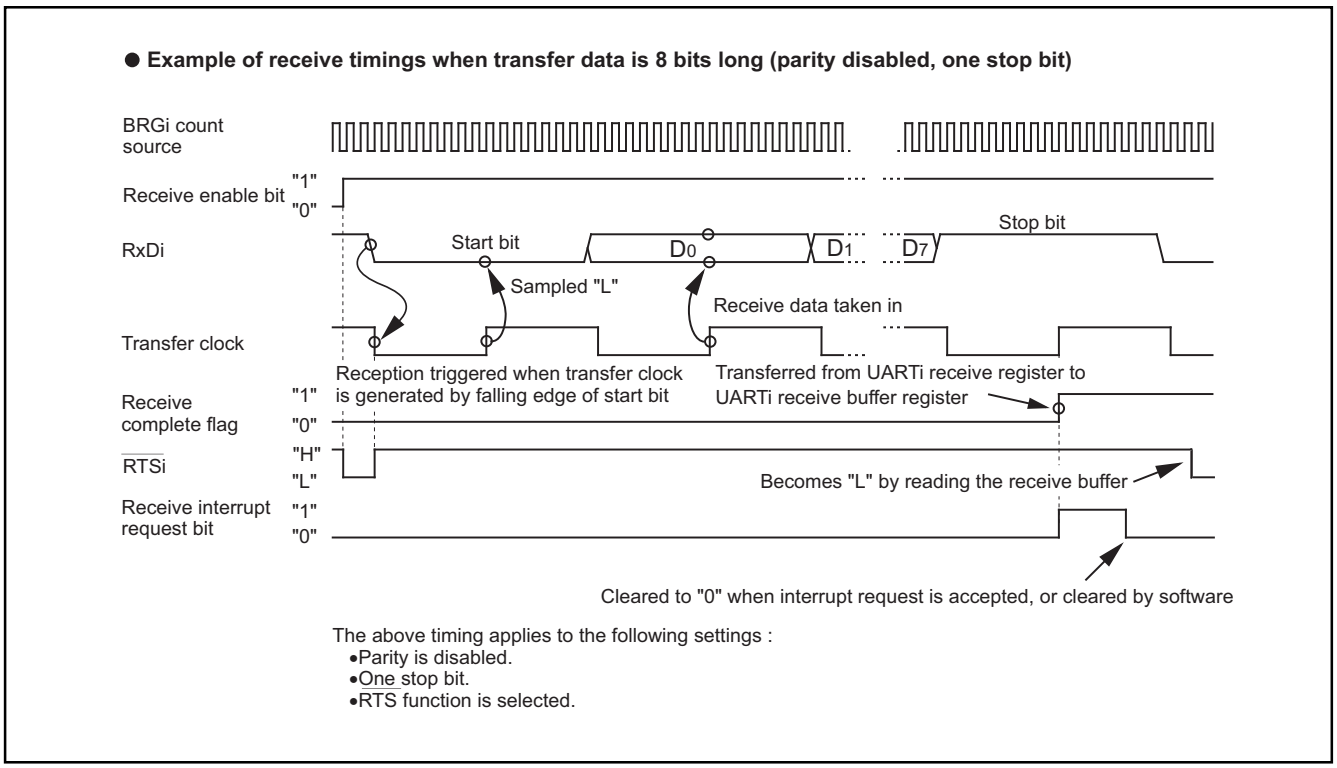



**Figure 1.99. Typical receive timing in UART mode**

### Serial data logic switch function

When the data logic select bit (bit 6 of address 03AD16, 036D16, 033D16, 032D16) is set to "1", data is inverted when writing to the transmission buffer register or reading the reception buffer register. Figure 1.100 shows an example of timing for switching serial data logic.

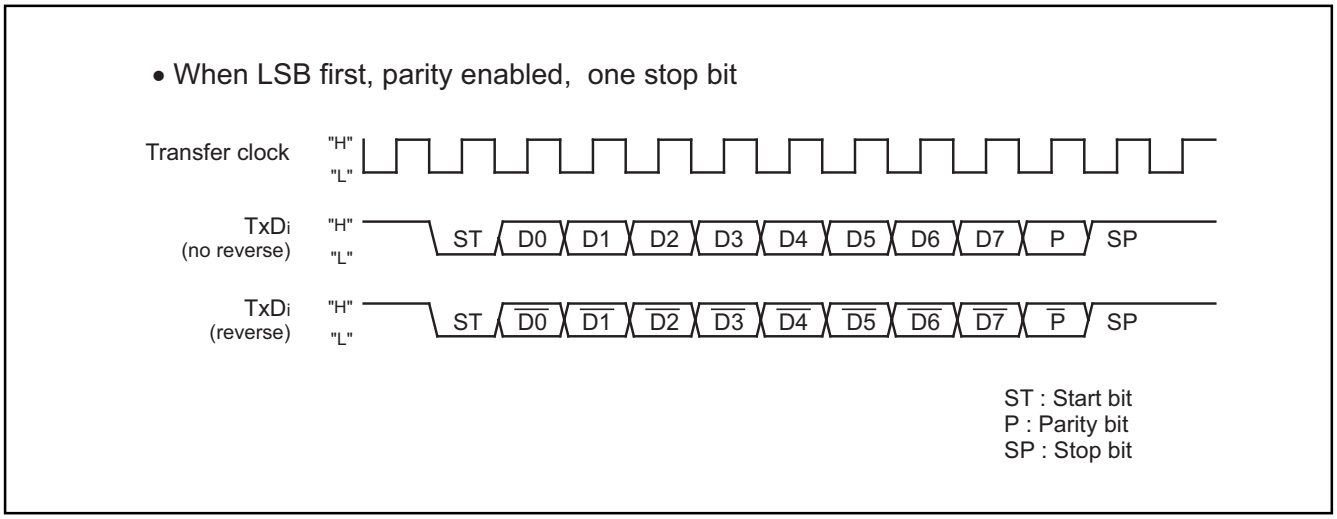



**Figure 1.100. Timing for switching serial data logic**



Downloaded from Arrow.com.

**TxD, RxD I/O polarity reverse function**

This function reverses the TxD pin output and RxD pin input. The level of any data input or output including the start bit, stop bits and parity bit, is reversed. Set this function to "0", (not to reverse) for normal use.

**Bus collision detection function**

This function samples the output level of the TxD pin and the input level of the RxD pin at the rising edge of the transfer clock. If their values are different, then an interrupt request occurs. Figure 1.101 shows an example of detection timing of a bus collision in UART mode.

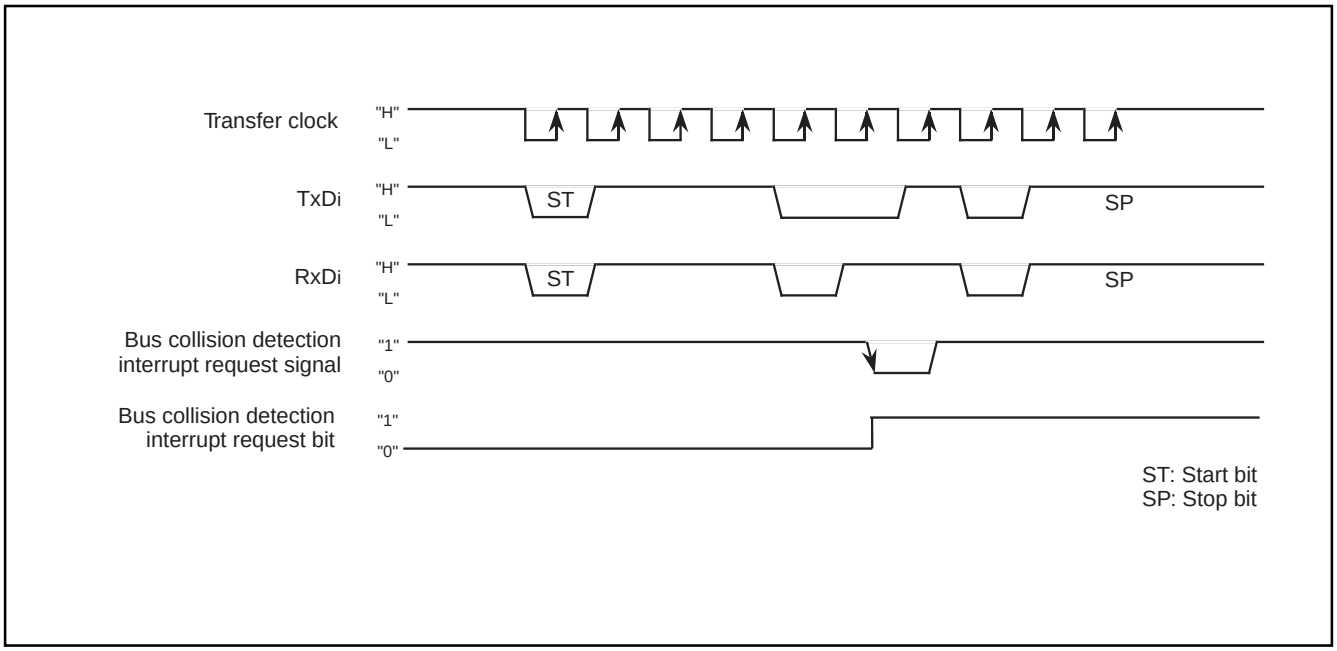



**Figure 1.101. Detection timing of a bus collision in UART mode**



Downloaded from Arrow.com.

## UART mode (compliant with the SIM interface)

The SIM interface is used for connecting the microcomputer with a memory card IC or a similar device. Adding some extra settings in UART mode allows the user to effect this function. Table 1.48 shows the specifications of UART mode compliant with SIM interface. Figure 1.102 shows typical transmit/receive timing in UART mode compliant with SIM interface.

**Table 1.48. Specifications of UART mode compliant with the SIM interface**

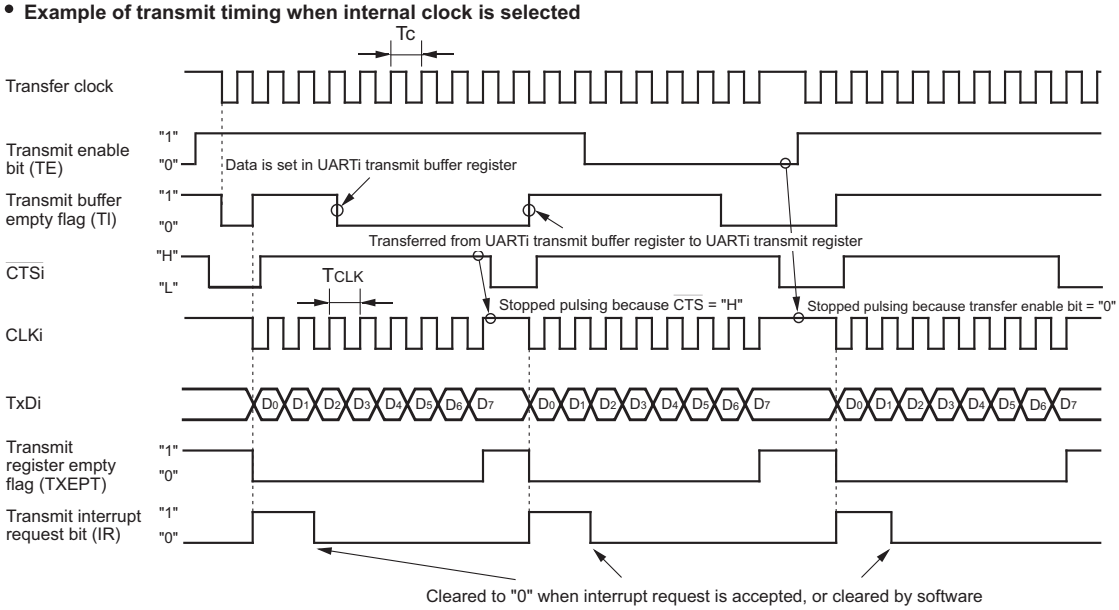
| Item | Specification |
|---|---|
| Transfer data format | •Transfer data 8-bit UART mode (bits 2 to 0 of address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "$101_2$")<br>•One stop bit (bit 4 of addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "0")<br>•With the direct format:<br>-Set parity to "even" (bits 5 and 6 of addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "1")<br>-Set data logic to "direct" (bit 6 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "0")<br>-Set transfer format to LSB (bit 7 of address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$ = "0")<br>•With the inverse format:<br>-Set parity to "odd" (bit 5 and 6 of address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "0" and "1" respectively)<br>-Set data logic to "inverse" (bit 6 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1")<br>-Set transfer format to MSB (bit 7 of address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$ = "1") |
| Transfer clock | •With the internal clock selected (bit 3 of address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "0"): fi/16(m+1) (Note 1): fi=f1, f8, f32<br>•With an external clock selected (bit 3 of address $03A8_{16}$, $0368_{16}$, $0338_{16}$, $0328_{16}$ = "1"): fEXT/16(m+1) (Notes 1,2) |
| Transmission/reception control | •Disable the CTS and RTS function (bit 4 of address $03AC_{16}$, $036C_{16}$, $033C_{16}$, $032C_{16}$ = "1") |
| Other settings | •Set transmission interrupt factor to “transmission completed” (bit 4 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1")<br>•Set N-channel open drain output to TxD pin in UART0, 1, 3 (bit 5 of address $03AC_{16}$, $036C_{16}$, $032C_{16}$ = "1") |
| Transmission start condition | •Transmit enable bit (bit 0 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1")<br>•Transmit buffer empty flag (bit 1 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "0") |
| Receive start condition | •Receive enable bit (bit 2 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1")<br>•Detection of a start bit |
| Interrupt request generation timing | •When transmitting<br>-When data transmission from the UART0 to UART3 transfer register is completed (bit 4 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1")<br>•When receiving<br>-When data transfer from the UART0 to UART3 receive register to the UART0 to UART3 receive buffer register is completed. |
| Error detection | •Overrun error (See UART specifications)<br>•Framing error (See UART specifications)<br>•Parity error (See UART specifications)<br>-On the reception side, an "L" level is output from the TxDi pin by use of the parity error signal output functions (bit 7 of address $03AD_{16}$, $036D_{16}$, $033D_{16}$, $032D_{16}$ = "1") when a parity error is detected.<br>-On the transmission side, a parity error is detected by the level of input to the RxDi pin when a transmit interrupt occurs<br>•The error sum flag (See UART specifications) |

Note 1: 'm' denotes the value $00_{16}$ to $FF_{16}$ that is set to the UARTi bit rate generator
Note 2: fEXT is input from the CLKi pin.



Downloaded from Arrow.com.

• **Example of transmit timing when internal clock is selected**

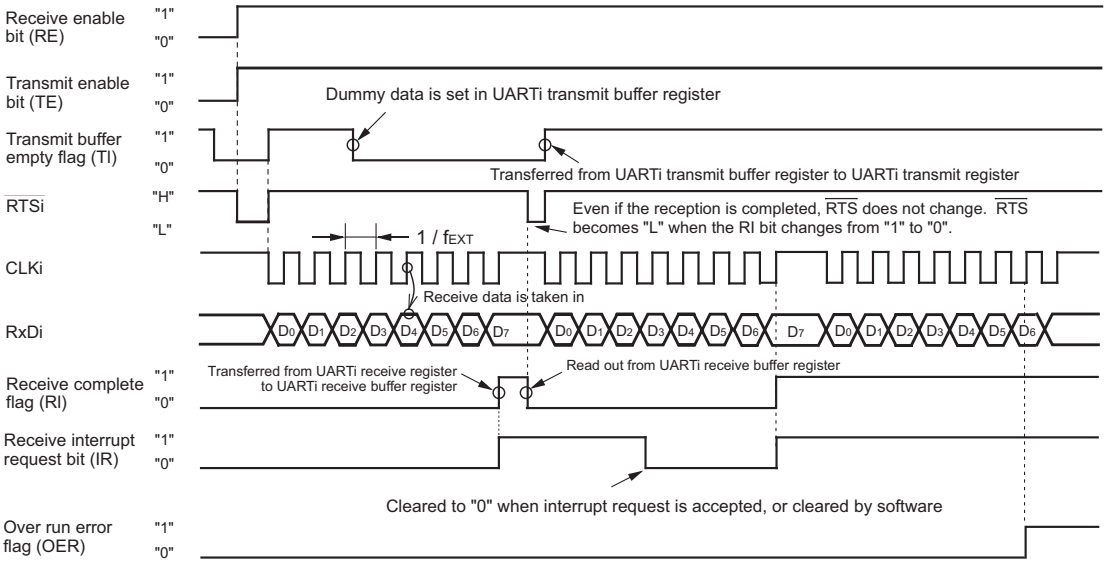



Shown in ( ) are bit symbols.

The above timing applies to the following settings:
- Internal clock is selected.
- CTS function is selected.
- CLK polarity select bit = "0".
- Transmit interrupt cause select bit = "0".

Tc = TCLK = 2(m + 1) / fi
fi : frequency of BRGi count source (f1, f8, f32)
m : value set to BRGi

• **Example of receive timing when external clock is selected**




Shown in ( ) are bit symbols.

The above timing applies to the following settings:
- External clock is selected.
- RTS function is selected.
- CLK polarity select bit = "0".

fEXT: frequency of external clock

The following conditions are met when the CLKi input before data reception = "H"
- Transmit enable bit →"1"
- Receive enable bit →"1"
- Dummy data write to UARTi transmit buffer register

**Figure 1.102. Typical transmit/receive timing in UART mode compliant with the SIM interface**



Downloaded from Arrow.com.

### Parity error signal function output

With the error signal output enable bit (bit 7 of addresses 03AD16, 036D16, 033D16, 032D16) set to "1", output an "L" level from the TxDi pin when a parity error is detected. When this occurs, the generation of a transmit complete interrupt changes to the detection of a parity error signal. Figure 1.103 shows the output timing of the parity error signal.

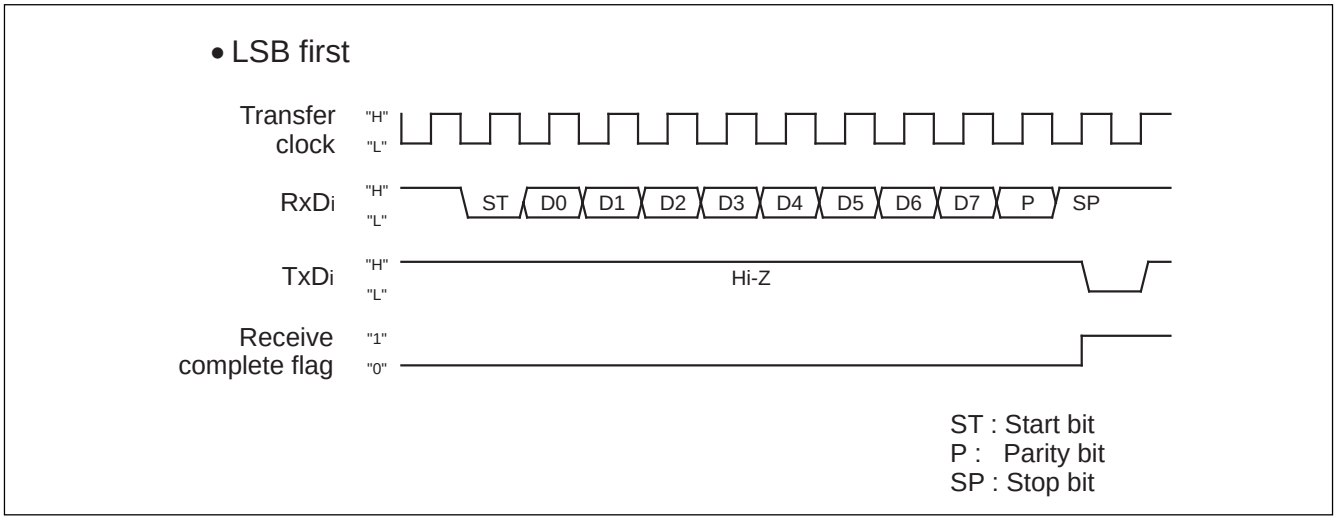



**Figure 1.103. Parity error signal output timing**

### Direct format/inverse format

Connecting the SIM card allows switching between direct format and inverse format. If the direct format is selected, D0 data is output from TxDi. If the inverse format is selected, D7 is inverted and output from TxDi. Figure 1.104 shows the SIM interface format. Figure 1.105 shows an example of connect the SIM interface. Connect TxDi and RxDi and apply pull-up.

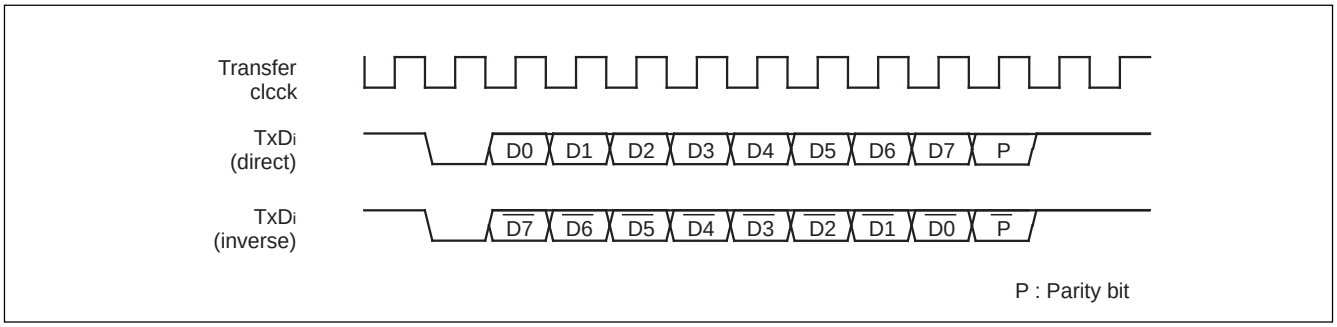



**Figure 1.104. SIM interface format**

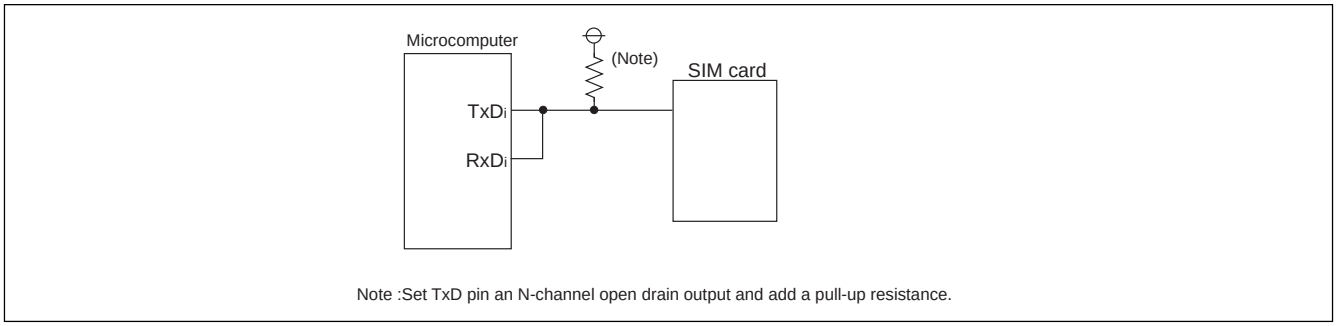



**Figure 1.105. Connecting the SIM interface**



Downloaded from Arrow.com.

### I²C Bus interface mode

The I²C bus interface mode is provided with UARTi. When the I²C mode select bit (bit 0 in addresses 03A7₁₆, 036716, 033716, and 032716) is set to "1", the I²C bus interface circuit is enabled.
To use the I²C bus in slave mode, SCLi should be set to input or to output “1”. Also for UART0, 1 and 3, set the data output select bit (bit 5 in address 03AC₁₆, 036C₁₆, and 032C₁₆) to N-channel open drain output. Note: UART2 TxD and RxD (P7₀ and P7₁) are always N-channel open drain outputs and require external pull-up resistors.
Table 1.49 shows the relationship of the I²C mode select bit to control. To use the chip in the clock synchronized serial I/O mode or UART mode, always set this bit to “0”. Figure 1.106 shows a block diagram of I²C mode.

**Table 1.49. I²C features**

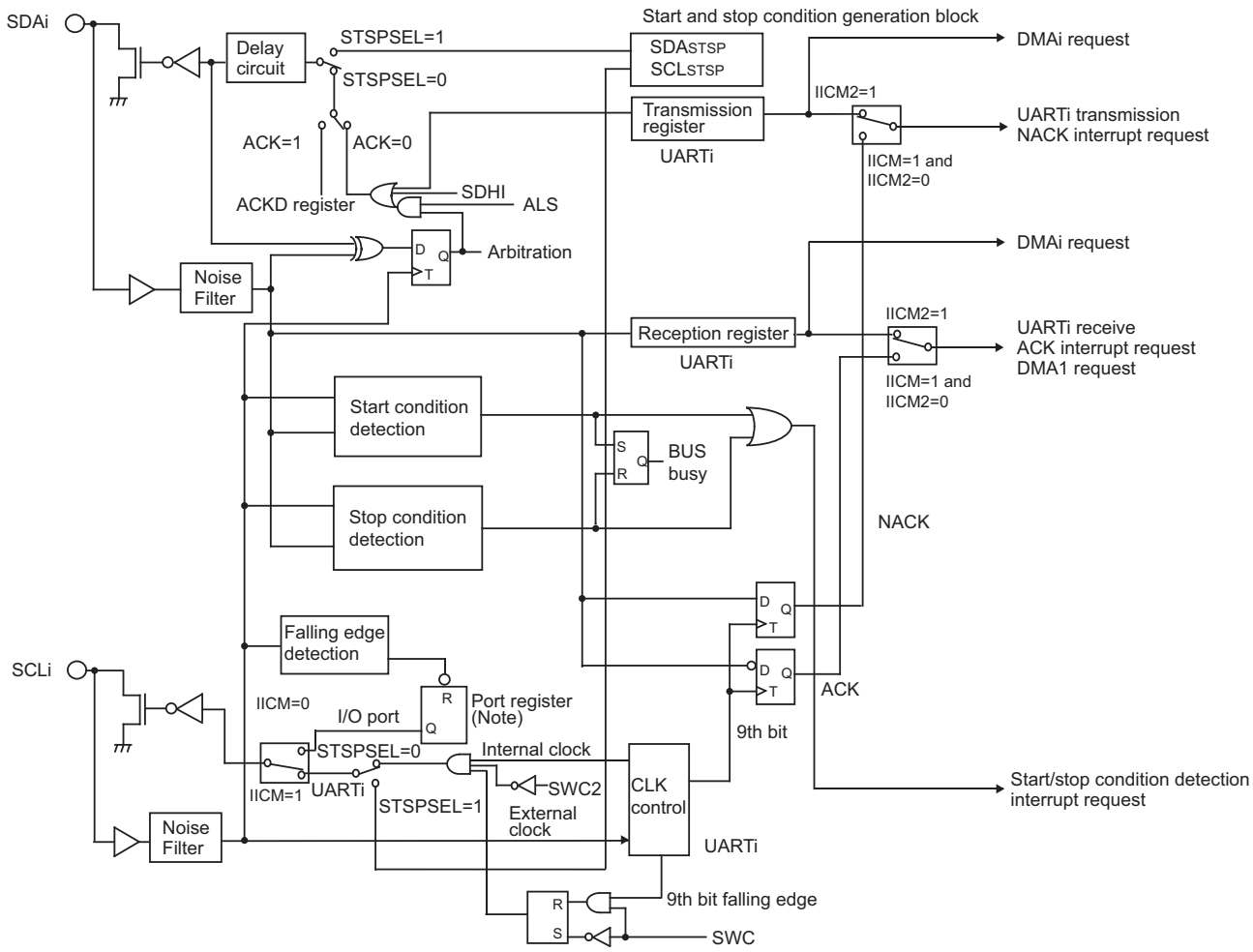
| | Function | Normal mode (IICM=0) | I²C mode (IICM=1) (Note 1) |
|---|---|---|---|
| 1 | Cause of interrupt number 3 and 9 (Note 2) | Bus collision detection | Start condition detection or stop condition detection |
| 2 | Cause of interrupt number 13 and 15 (Note 2) | UARTi transmit | No acknowledgement detection (NACK) |
| 3 | Cause of interrupt number 2 and 21 (Note 2) | UARTi receive | Acknowledgment detection (ACK) |
| 4 | UARTi transmit output delay | Not delayed | Delayed |
| 5 | $P6_3$, $P6_7$, $P7_0$, $P7_4$ at the same time UARTi is in use | TxDi (output) | SDAi (input/output) (Note 3) |
| 6 | $P6_2$, $P6_6$, $P7_1$, $P7_5$ at the same time UARTi is in use | RxDi (input) | SCLi (input/output) |
| 7 | $P6_1$, $P6_5$, $P7_2$, $P7_6$ at the same time UARTi is in use | CLKi | $P6_1$, $P6_5$, $P7_2$, $P7_6$ |
| 8 | DMA1 factor at the same time | UARTi receive | Acknowledgement detection (ACK) |
| 9 | Noise filter width | 15 ns | 50 ns |
| 10 | Reading $P6_2$, $P6_6$, $P7_1$, $P7_5$ | Reading the terminal when 0 is assigned to the direction register | Master mode: Reading the terminal regardless of the value of the direction register<br>Slave mode: Reading the terminal when the corresponding port register is set to "0" |
| 11 | Initial value of UARTi output | "H" level (when 0 is assigned to CLKi polarity select bit) | The value set in latch $P6_3$, $P6_7$, $P7_0$, $P7_4$ when the port is selected (Note 3) |

Note 1: When using I²C mode, set 0 1 0 in bits 2, 1, 0 of the UARTi transmit/receive mode register. Disable the $\overline{CTS}$/$\overline{RTS}$ function. Select MSB first function.
Note 2: To switch from one factor to another:
1. Disable the interrupt of the corresponding number.
2. Switch to another factor.
3. Reset the interrupt request flag of the corresponding number.
4. Set the interrupt level of the corresponding number.

Note 3: Set an initial value of SDA transmission output when I²C mode (I²C mode select bit = "1") is valid and serial I/O is invalid.



Downloaded from Arrow.com.




This diagram applies to the case where the UiMR register's SMD2 to SMD0 bits=010₂ and the UiSMR register's IICM bit=1.

IICM : UiSMR register bit
IICM2, SWC, ALS, SWC2, SDHI : UiSMR2 register bit
STSPSEL, ACKD, ACKC : UiSMR4 register bit

i=0 to 3

Note: In I²C master mode, the port terminal is to be readable even if "1" is assigned to P6₂, P6₆, P7₁, P7₅ of the direction register

**Figure 1.106. I²C mode functional block diagram**



Downloaded from Arrow.com.

**UARTi Special Mode Register (UiSMR)**

**Bit 0** is the I2C mode select bit 1. When set to "1", ports operate respectively as the SDAi data transmit/receive pin, SCLi clock input/output pin and port. A delay circuit is added to SDAi transmission output, therefore after SCLi is at "L" level, SDAi output changes. In I2C master mode, port (SCLi) is designed to read pin level regardless of the content of the port direction register. SDAi transmission output is initially set to port in this mode. Furthermore, interrupt factors for the bus collision detection interrupt and UARTi transmission interrupt change respectively to the start/stop condition detection interrupts, acknowledge non-detection interrupt and acknowledge detection interrupt.
The start condition detection interrupt is generated when the falling edge at the SDAi pin is detected while the SCLi pin is in "H" state. The stop condition detection interrupt is generated when the falling edge at the SDAi pin is detected while the SCLi pin is in the "H" state.
The acknowledge non-detect interrupt is generated when the "H" level at the SDAi pin is detected at the 9th rise of the transmission clock. The acknowledge detect interrupt is generated when the "L" level at the SDAi pin is detected at the 9th fall of the transmission clock. Also, DMA transfer can be started when the acknowledge is detected if UARTi transmission is selected as the DMAi request factor.

**Bit 1** is the arbitration detection flag control bit. Arbitration detects a conflict between data transmitted at SCLi rise and data at the SDAi pin. This detect flag is allocated to bit 11 in UARTi transmit buffer register (addresses $036F_{16}$, $02EF_{16}$, $033F_{16}$, $032F_{16}$, $02FF_{16}$). It is set to "1" when a conflict is detected. With the arbitration lost detect flag control bit, it can be selected to update the flag in units of bits or bytes. When this bit is set to "1", update is set to units of byte. If a conflict is still detected, the arbitration lost detect flag control bit will be set to "1" at the 9th rise of the clock. When updating in units of byte, always clear ("0" interrupt) the arbitration lost detect flag control bit after the first byte has been acknowledge but before the next byte starts transmitting.

**Bit 2** is the bus busy flag. It is set to "1" when the start condition is detected, and reset to "0" when the stop condition is detected.

**Bit 3** is the SCLi L synchronization output enable bit. When this bit is set to "1", the port data register is set to "0" in sync with the "L" level at the SCLi pin.

**Bit 4 to Bit 6 :** These are not used in I2C bus interface mode. See "IE mode" section.



Downloaded from Arrow.com.

### UARTi Special Mode Register 2 (UiSMR2)

**Bit 0** is the I²C mode select bit 2. Table 1.50 lists the control changes by bit when the I²C mode select bit is "1". Start and stop condition detection timing characteristics are shown in Figure 1.107.

**Table 1.50. Functions changed by I²C mode select bit 2**

| Function | IICM2=0 | IICM2=1 |
|---|---|---|
| Interrupt numbers 13, 15, 17, 19 factor | Acknowledge not detected (NACK) | UARTi transfer (rising edge of the last bit) |
| Interrupt number 2, 8, 10, 21 factor | Acknowledge detected (ACK) | UARTi receive (falling edge of the last bit) |
| DMA factor | Acknowledge detected (ACK) | UARTi receive (falling edge of the last bit) |
| Data transfer timing from UART receive shift register to receive buffer | Rising edge of the last bit of receive clock | Rising edge of the last bit of receive clock |
| UART receive/ACK interrupt request generation timing | Rising edge of the last bit of receive clock | Rising edge of the last bit of receive clock |

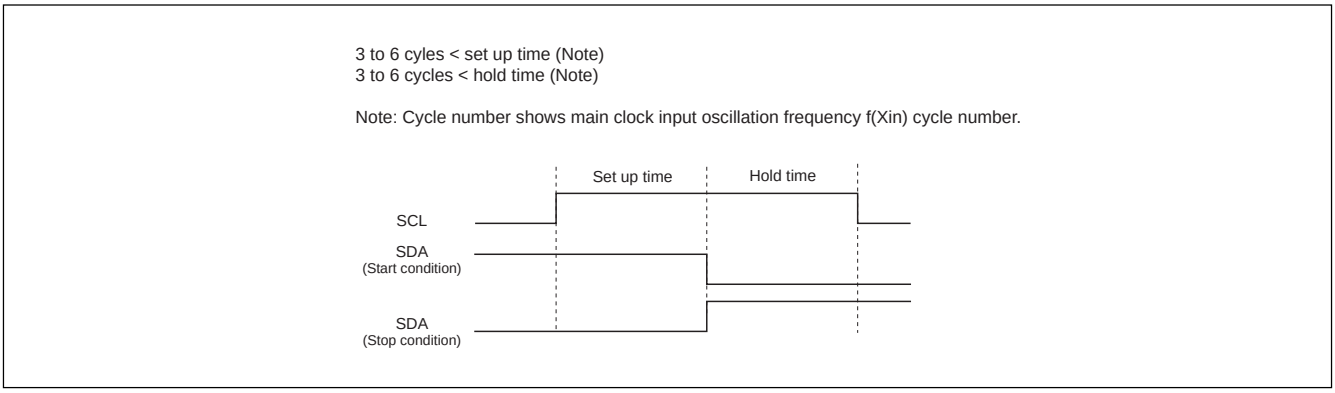



**Figure 1.107. Start/stop condition detect timing characteristics**

**Bit 1** is the clock synchronizing bit. When this bit is set to "1", if the falling edge is detected at pin SCLi while the internal SCL is "H", the internal SCL is changed to "L", the baud rate generator value is reloaded and the L sector count starts. Also, while the SCLi pin is "L", if the internal SCL changes from "L" to "H", baud rate generator stops counting. If the SCLi pin is "H", counting restarts. Because of this function, the UARTi transmit/receive clock takes the AND condition for the internal SCL and SCLi pin signals. This function operates from the clock half period before the first rise of the UARTi clock to the 9th rise. To use this function, select the internal clock as the transfer clock.

**Bit 2** is the SCL wait output bit. When this bit is set to "1", output from the SCLi pin is fixed to "L" at the clock's 9th fall. When set to "0", the "L" output lock is released. This bit is unavailable when SCLi is external clock.

**Bit 3** is the SDA output stop bit. When this bit is set to "1", an arbitration lost generated. If the arbitration lost detection flag is "1", the SDAi pin simultaneously becomes high impedance.

**Bit 4** is the UARTi initialize bit. While this bit is set to "1", the following operations are performed when the start condition is detected.

• The transmit shift register is initialized and the content of the transmission register is transmitted to the transmission shift register. Transmission starts with the first bit of the next input clock. However, the UARTi output value does not change when the start condition is detected. It also doesn't change when the clock is input and when the first bit of data is output.

• The receive shift register is initialized and reception starts with the first bit of the next input clock.

• The SCL wait output is set to "1". The SCLi pin becomes "L" level at the fall of the 9th bit of the clock.



Downloaded from Arrow.com.

When UART transmit/receive is started using this function, the content of the transmit buffer available flag does not change. Also, to use this function, select an external clock as the transfer clock. This bit is unavailable when SCLi is external clock.

**Bit 5** is SCL wait output bit 2. When this bit is set to "1" and serial I/O is selected, an "L" level can be output from the SCLi pin even during UART operation. When this bit is set to "0", the "L" output from the SCLi pin is cancelled and the UARTi clock is input and output. This bit is unavailable when SCLi is external clock.

**Bit 6** is the SDA output disable bit. When this bit is set to "1", the SDAi pin is forced to high impedance. Overwrite this bit at the rise of the UART transfer clock. The arbitration lost detection flag may be set.

### UARTi Special Mode Register 3 (UiSMR3)

**Bit 0 :** Not used in I²C bus interface mode. See "SPI mode" section.

**Bit 1** is the clock phase set bit. When both the I²C mode select bit (bit 0 of UiSMR) and the I²C mode select bit 2 (bit 0 of UiSMR2) are "1", functions changed by these bits are shown in Table 1.51 and Figure 1.108.

**Bit 2 :** Not used in I²C bus interface mode. See "SPI mode" section

**Bit 3 :** Not used in I²C bus interface mode.

**Bit 4 :** Not used in I²C bus interface mode. See "SPI mode" section.

**Bit 5 to 7** are the I²C SDAi digital delay setting bits. By setting these bits, it is possible to turn the SDAi delay OFF or set the f(Xin) delay to 2 to 8 cycles.

**Table 1.51. Functions changed by clock phase set bits**

| Function | CKPH=0, IICM=1, IICM2=1 | CKPH=1, IICM=1, IICM2=1 |
|---|---|---|
| SCL initial and last value | Initial value = "H", last value = "H" | Initial value = "L", last value = "L" |
| Transfer interrupt factor | Rising edge of 9th bit | Falling edge of 10th bit |
| Data transfer times from UART receive shift register to receive buffer register | Falling edge of 9th bit | Two-times: falling edge of 9th bit and rising edge of 9th bit |

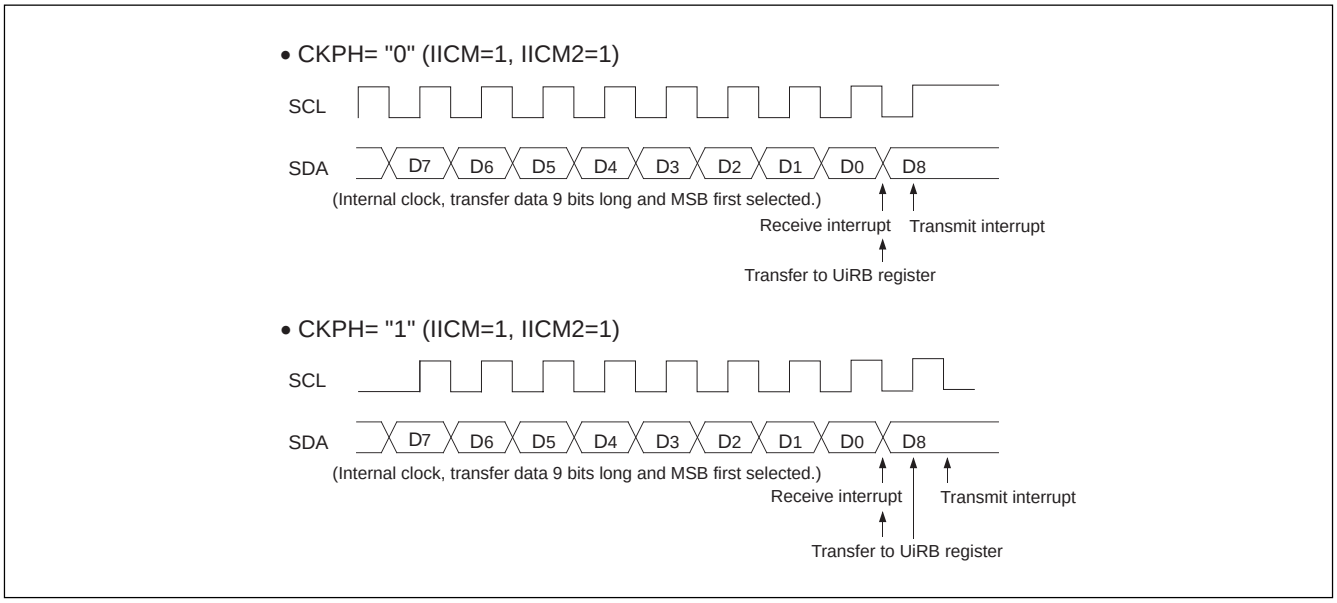



**Figure 1.108. Function changed by clock phase set bits**



Downloaded from Arrow.com.

### UARTi Special Mode Register 4 (UiSMR4)

**Bit 0** is the start condition generate bit. When the SCL, SDA output select bit (bit 3 of UiSMR4) is "1" and this bit is "1", the start condition is generated.

**Bit 1** is the restart condition generate bit. When the SCL, SDA output select bit (bit 3 of UiSMR4) is "1" and this bit is "1", the restart condition is generated.

**Bit 2** is the stop condition generate bit. When the SCL, SDA output select bit (bit 3 of UiSMR4) is "1" and this bit is "1", the stop condition is generated.

**Bit 3** is SCL, SDA output select bit. Table 1.52 shows the functions that are changed by this bit. Figure 1.109 shows the functions changed by SCL, SDA output select bit.

**Bit 4** is ACK data bit. When the SCL, SDA output select bit (bit 3 of UiSMR4) is "0" and the ACK data output enable bit (bit 5 of UiSMR4) is "1", the content of ACK data bit is output to SDAi pin.

**Bit 5** is ACK data output enable bit. When the SCL, SDA output select bit (bit 3 of UiSMR4) is "0" and this bit is "1", the content of ACK data bit is output to SDAi pin.

**Bit 6** is SCL output stop bit. When this bit is "1", SCLi output is stopped at stop condition detection. (High-Z status).

**Bit 7** is SCL wait output bit 3. When this bit is "1", SCLi output is fixed to "L" at the falling edge of the 10th clock bit. When this bit is "0", SCLi output fixed to "L" is released. This bit is unavailable when SCLi is external clock.

**Table 1.52. Functions changed by SCL, SDA output select bit**

| Function | STSPSEL=0 | STSPSEL=1 |
|---|---|---|
| SCL, SDA output | Output of S I/O control circuit | Output of start/stop condition control circuit |
| Start/stop condition interrupt factor | Start/stop condition detection | Completion of start/stop condition generation |

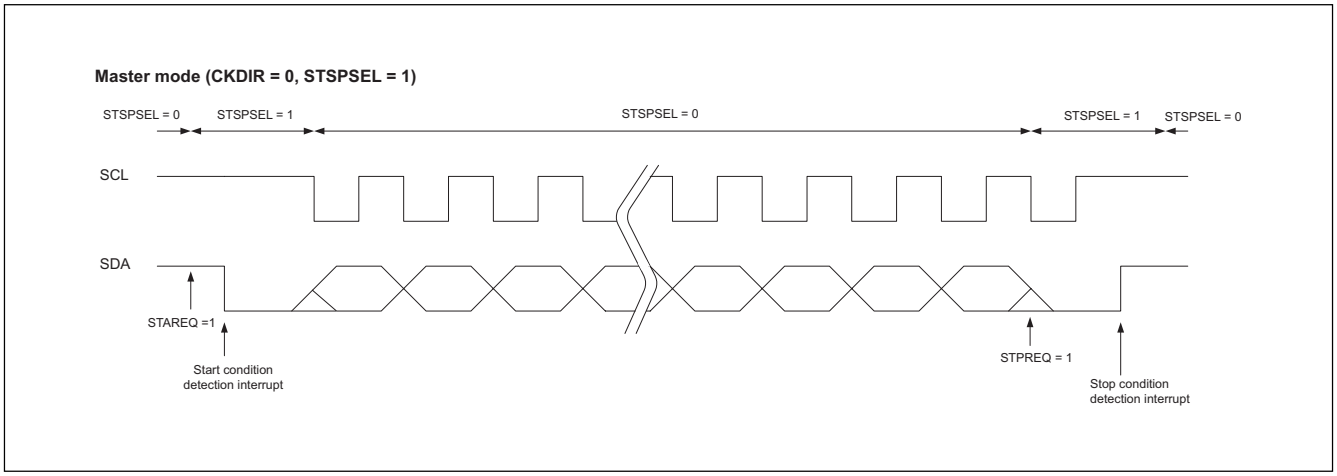



**Figure 1.109. Functions changed by SCL, SDA output select bit**



Downloaded from Arrow.com.

# Serial Interface Special Function (SPI mode)

### SPI mode related control bit

#### UARTi Special Mode Register 3 (UiSMR3)

**Bit 0** is the SS port function enable bit. Set this bit to "1" to enable the slave select output.

**Bit 1** is the clock phase set bit.

**Bit 2** is the serial input port set bit.

**Bit 4** is the fault error flag. When this bit is "1", a fault error has been detected.

**Bit 3, 5 to 7 :** Not used in SPI mode.

UARTi can control communications on the serial bus using the SSi input pins. The master outputting the transfer clock transfers data to the slave inputting the transfer clock. To prevent a bus collision, the master floats the output pin of other slaves/masters using the SSi input pins. Figure 1.110 shows the SSi input pin factors between the master and slave.

**Slave mode (STxDi and SRxDi are selected, DINC="1")**

When an "H" level signal is input to an SSi input pin, the STxDi and SRxDi pins both become high impedance and the clock input is ignored. When an "L" level signal is input to an input pin, SSi clock input becomes effective and serial communications are enabled.

**Master mode (TxDi and RxDi are selected, DINC="0")**

The SSi input pins are used with a multiple master system. When an SSi input pin is "H" level, transmission has priority and serial communications are enabled. When an "L" signal is input to an SSi input pin, another master exists, and the TxDi, RxDi and CLKi pins become high impedance and the trouble error interrupt request bit becomes "1". Communications do not stop when a trouble error is generated during communications. To stop communications, set bit 0, 1, 2 of the UARTi transmit/receive mode register (addresses $03A8_{16}$, $0368_{16}$, $0338_{16}$, and $0328_{16}$) to "0".

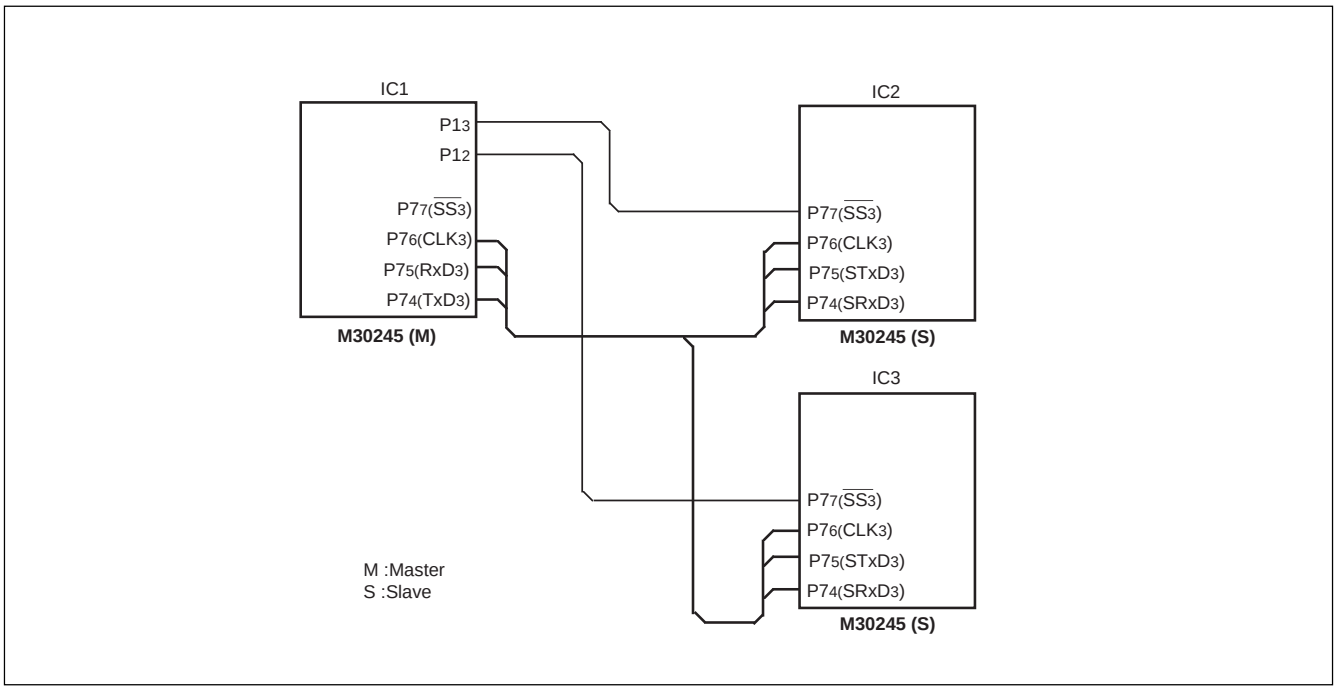



**Figure 1.110. Example of serial bus communication control using SSi input pins**



Downloaded from Arrow.com.

### Clock phase setting

With bit 1 of UARTi special mode register 3 and bit 6 of UARTi transmit/receive control register 0, four combinations of transfer clock phase and polarity can be selected. Bit 6 of UARTi transmit/receive control register 0 sets transfer clock polarity, whereas bit 1 of UiSMR3 register sets transfer clock phase. Transfer clock phase and polarity must be the same between the master and slave involved in the transfer.

**• Master (Internal clock) (DINC=0)**

Figure 1.111 shows the transmit and receive timing.

**• Slave (External clock) (DINC=1)**

When "0" for CKPH bit (bit 1 of UiSMR3) is selected and SSi input pin is "H" level, output data is high impedance. When an SSi input pin is "L" level, the serial transmission start condition is satisfied even though output is indeterminate and serial transmission is synchronized with the clock. Figure 1.112 shows the timing.
When "1" is selected for CLPH bit and SSi input pin is "H" level, output data is high impedance. When an SSi input pin is "L" level, the first data is output and serial transmission is synchronized with the clock. Figure 1.113 shows the timing.

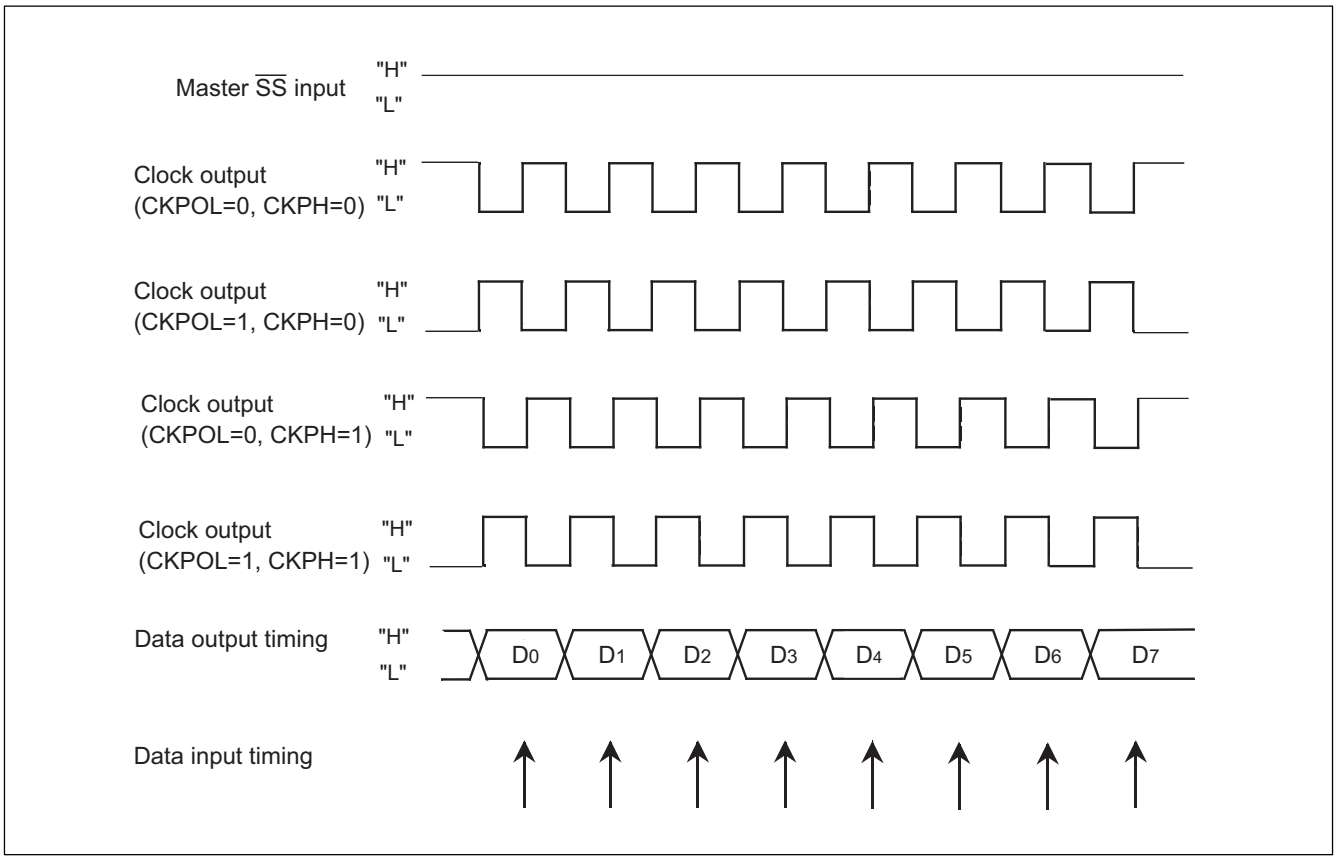
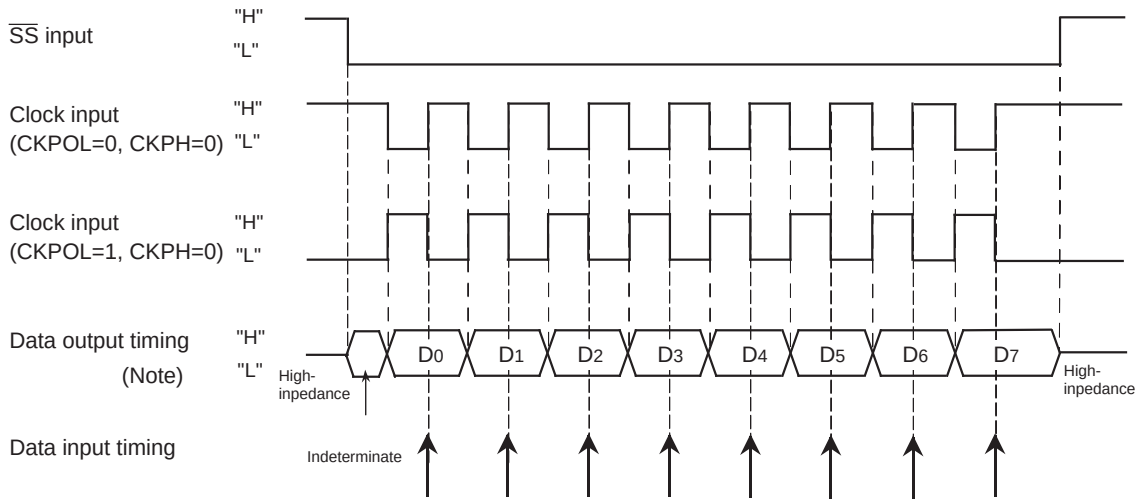



**Figure 1.111. Transmit/receive timing in master mode (Internal clock)**

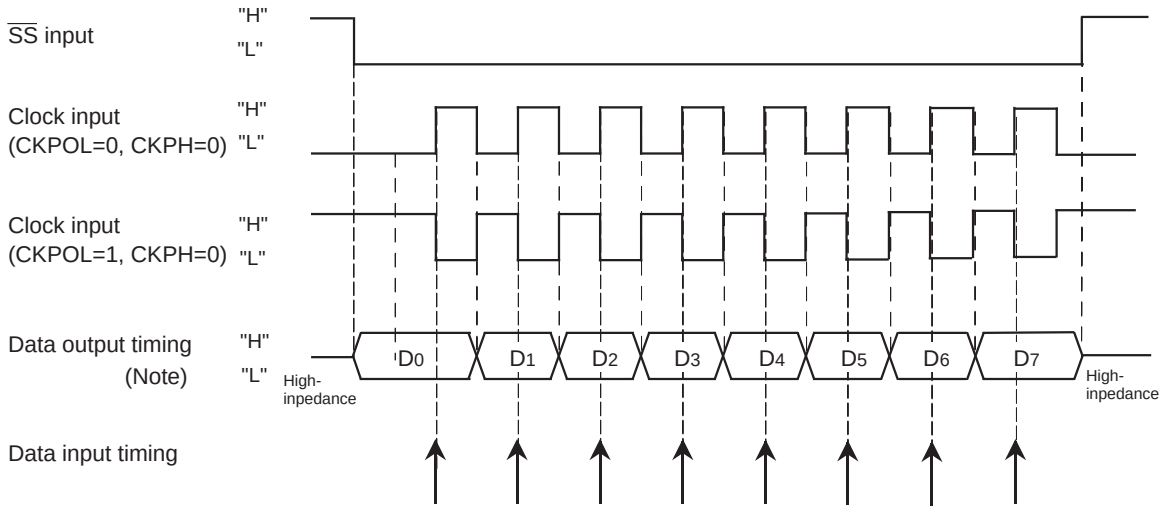


Downloaded from Arrow.com.




Note :UART2, (P70, P71) output is an N-channel open drain and needs to be pulled-up externally.

**Figure 1.112. Transmit/receive timing (CKPH=0) in slave mode (External clock)**




Note :UART2 (P70, P71) output is an N-channel open drain and needs to be pulled-up externally.

**Figure 1.113. Transmit/receive timing (CPKH=1) in slave mode (External clock)**



Downloaded from Arrow.com.

**IE Mode (UiSMR)**

**Bit 0 to 3 :** Not used in IE mode.

**Bit 4** is the bus collision detection sampling clock select bit. The bus collision detection interrupt is generated when RxDi and TxDi level conflict with each other. When this bit is "0", a conflict is detected in sync with the rise of the transfer clock. When this bit is "1", detection is made when Timer Aj (Timer A3:UART0, Timer A4:UART1, Timer A0:UART2, Timer A3:UART3 and Timer A4:UART4) underflows. Timer Aj (one-shot mode) should be triggered with corresponding RxDi pin by connecting RxDi pin to TAjIN pin. The operation is shown in Figure 1.114.

**Bit 5** is the transmission enable bit automatic clear select bit. By setting this bit to "1", the transmission bit is automatically reset to "0" when the bus collision detection interrupt factor bit is "1" (when a conflict is detected).

**Bit 6** is the transmit start condition select bit. By setting this bit to "1", TxDi transmission starts in sync with the rise at the RxDi pin.

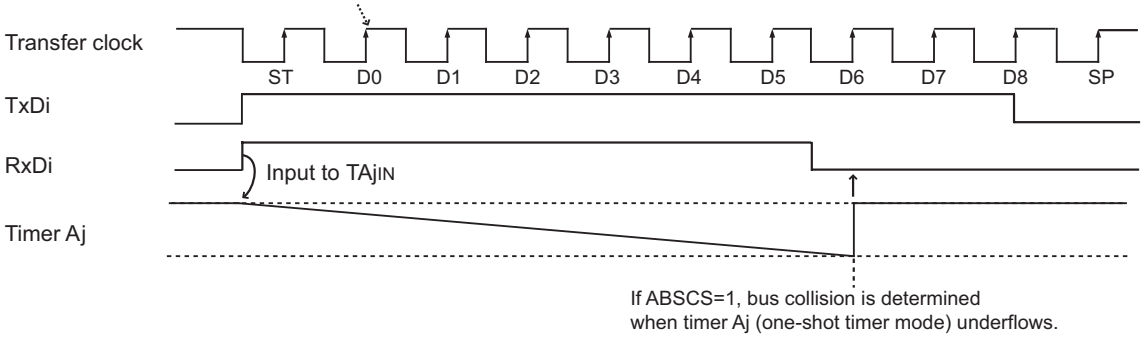


Downloaded from Arrow.com.

**(1) UiSMR register ABSCS bit (bus collision detect sampling clock select)**

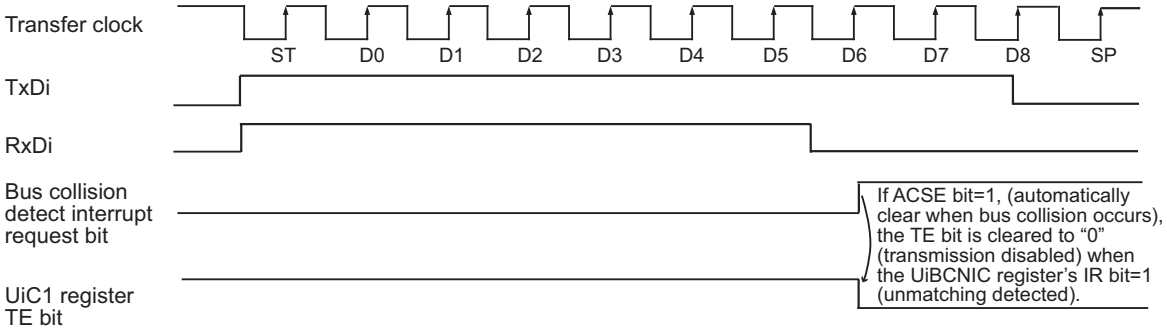
(i=0 to 3)

If ABSCS=0, bus collision is determined at the rising edge of the transfer clock



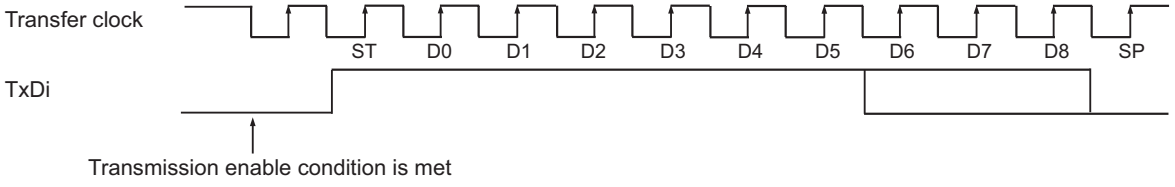

Timer Aj : timer A3 when UART0; timer A4 when UART1; timer A0 when UART2; timer A3 when UART3)

**(2) UiSMR register ACSE bit (auto clear of transmit enable bit)**

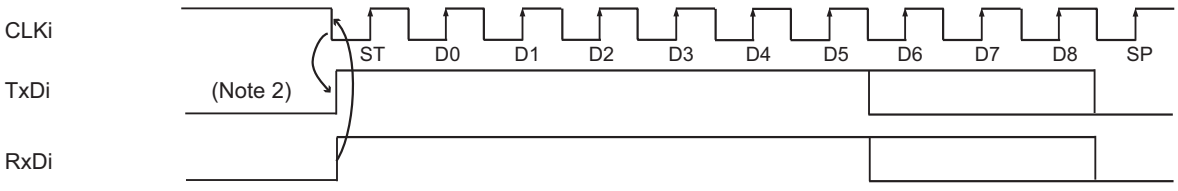



**(3) UiSMR register SSS bit (Transmit start condition select)**

If SSS bit=0, the serial I/O starts sending data one transfer clock cycle after the transmission enable condition is met.




If SSS bit=1, the serial I/O starts sending data at the rising edge (Note 1) of RxDi




Note 1: The falling edge of RxDi when IOPOL=0; the rising edge of RxDi when IOPOL=1.
Note 2: TheTransmit condition must be met before the falling edge (Note 1) of RxD.

This diagram applies to the case where IOPOL=1 (reserved).

**Figure 1.114. Bus collision Detect Function-Related Bits**



Downloaded from Arrow.com.

## Serial Sound Interface

Serial Sound Interface is a synchronous serial data interface used primarily for transferring digital audio data. This functional block is compatible with the $I^2S$ standard but also adds some extra configurability for custom interfaces. A channel is any single output of an audio system. For example, the left and right speakers are the two channels of a simple stereo audio system. The bus has 4 lines:

- Continuous serial clock (SCK);
- Word (channel) select (WS);
- Serial data out (XMT);
- Serial data in (RX).

The channel being transmitted changes on every transition of WS. A Serial Sound Interface-based communication system has two Serial Sound Interfaces and a master controller which generates both SCK and WS. A Serial Sound Interface which generates the controls (SCK and WS) along with its transmit and receive signals is operating as a master. A Serial Sound Interface which uses external control signals is operating as a slave. The Serial Sound Interface on this device operates only as a slave.
Figure 1.115 shows a high level system diagram of a Serial Sound Interface setup and its associated waveforms when configured for six channels.

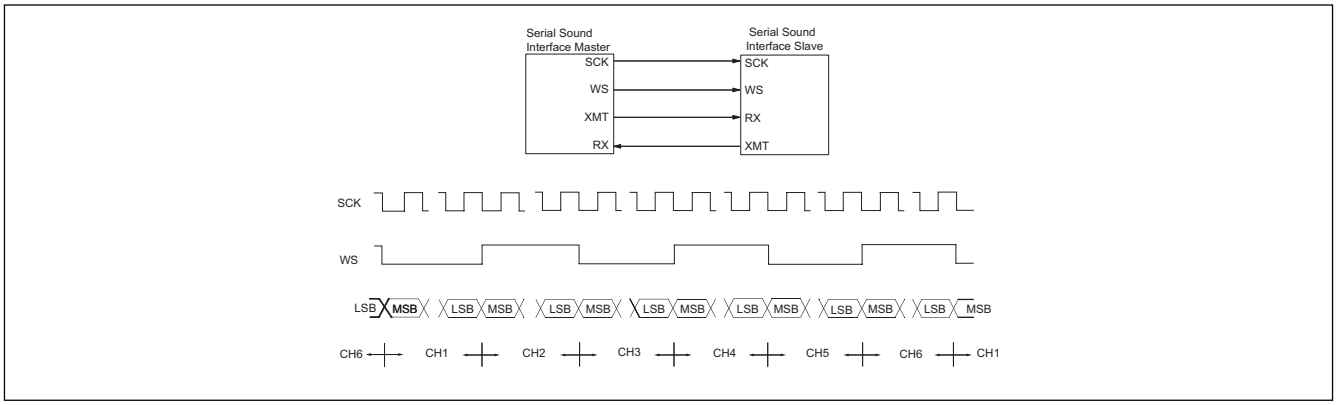



**Figure 1.115. Serial Sound Interface System diagram**



Downloaded from Arrow.com.

### Data transmission format

The transmitter/receiver must change channels on every WS transition. If the number of SCKs within a WS high/low period exceeds the channel width (set by the user via mode bits), the transmitter continues to transmit '0's, while the receiver will stop receiving data until the next WS edge. However, if the number of SCKs falls short, both the transmitter and the receiver will immediately switch to the next channel transmit and receive, respectively. The Serial Sound Interface architecture is shown in Figure 1.116.

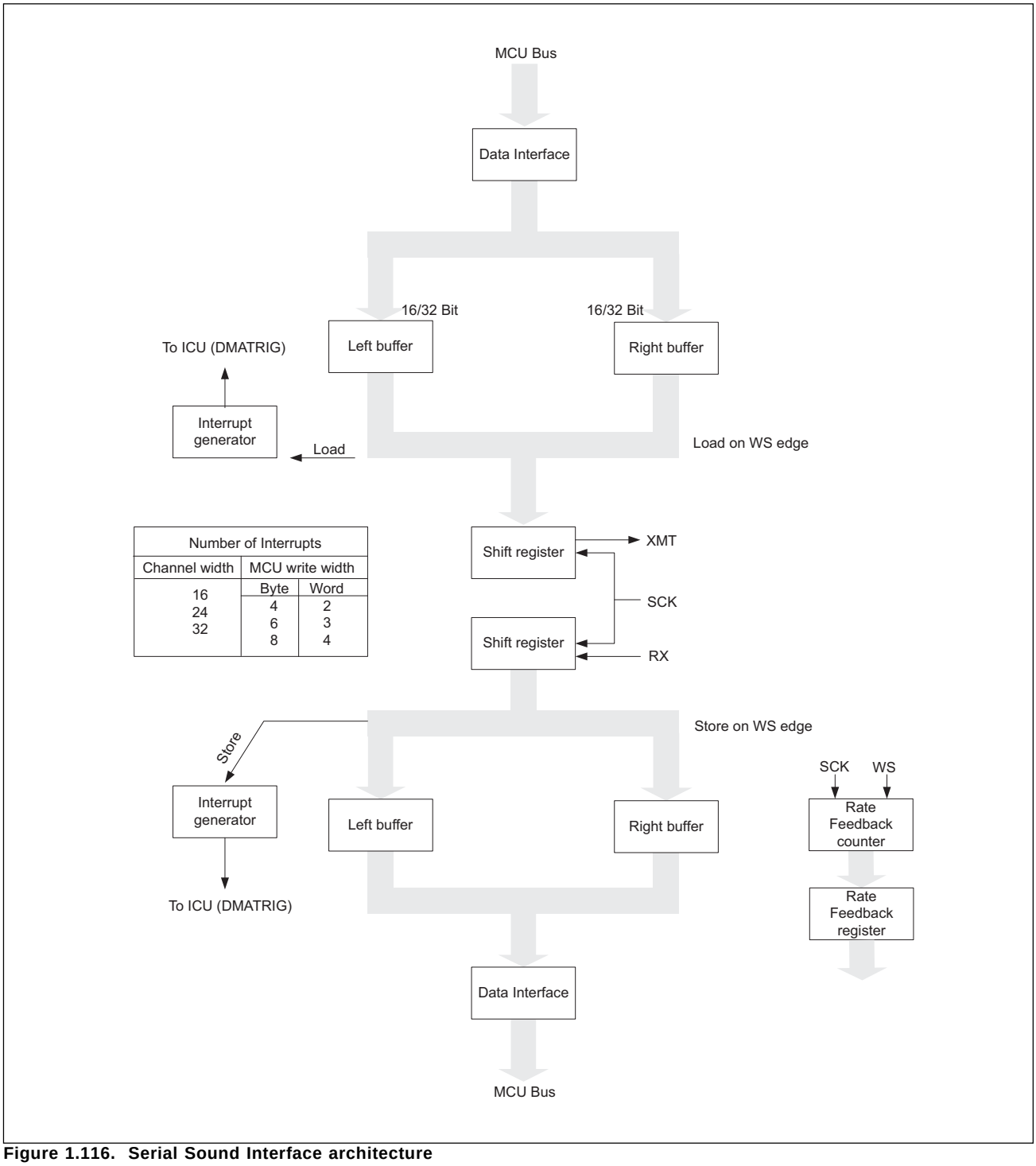



**Figure 1.116. Serial Sound Interface architecture**



Downloaded from Arrow.com.

The following features are supported via firmware controlled mode bits:

• Simultaneous transmit and receive (through separate transmit and receive pins) synchronized to the same SCK and WS signals.
• Transmit/receive data and WS synchronized to the rising edge or the falling edge of SCK as shown in Figure 1.117.
• Transmit and receive synchronized to the rising or the falling edge of WS as shown in Figure 1.118.
• Normal or delayed WS: WS transitions one SCK period before a channel change (normal mode) or concurrently with a channel change (delayed mode) as shown in Figure 1.119.
• Automatic interrupt on a channel change and on every access to the data buffer (transmit/receive) until each data buffer byte is accessed.
• Channel widths of 32, 24, and 16 bits.
• MSB or LSB first transmit and receive.
• Multiple receive formats: if the number of SCKs in a WS high/low period is less than the channel width, the data can be placed either MSB or LSB justified as shown in Figure 1.120.
• Rate feedback: when used with the USB interface, the Serial Sound Interface can count the number of WS's or SCK's per USB frame.

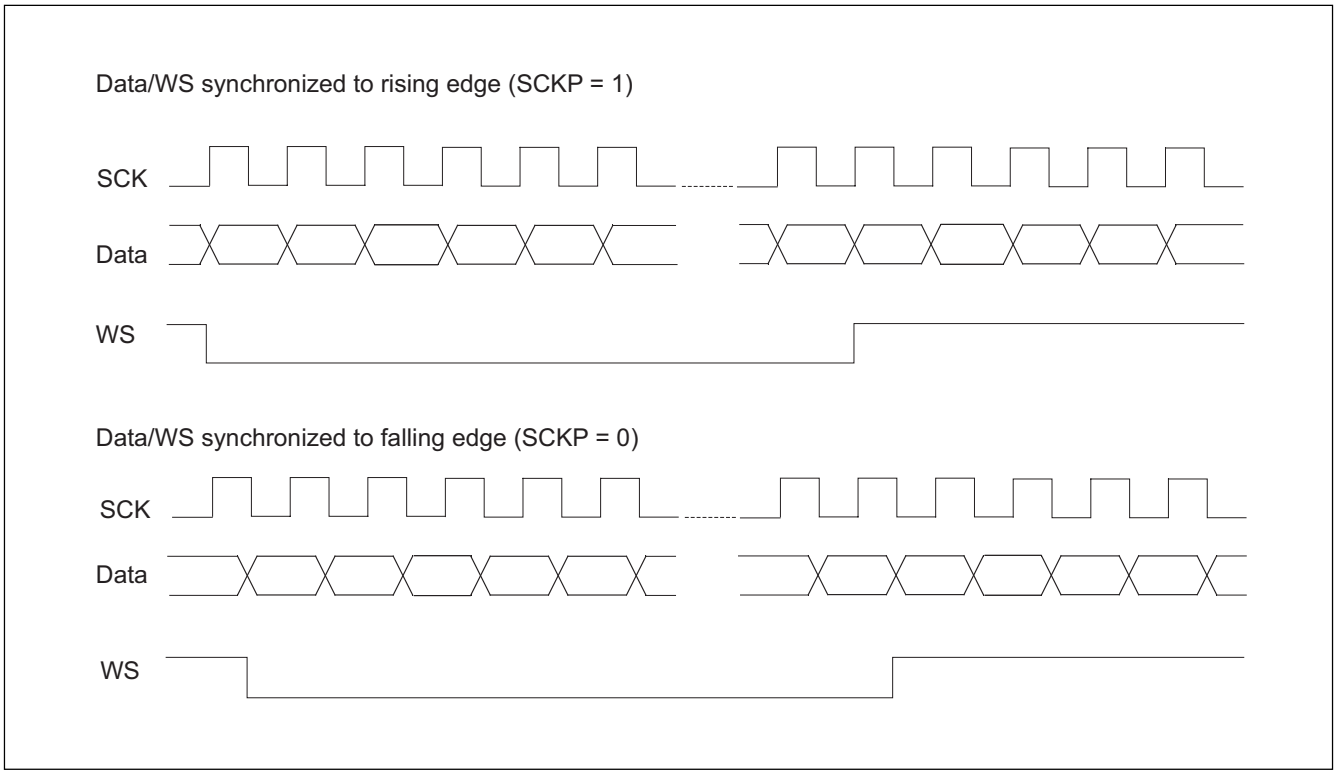
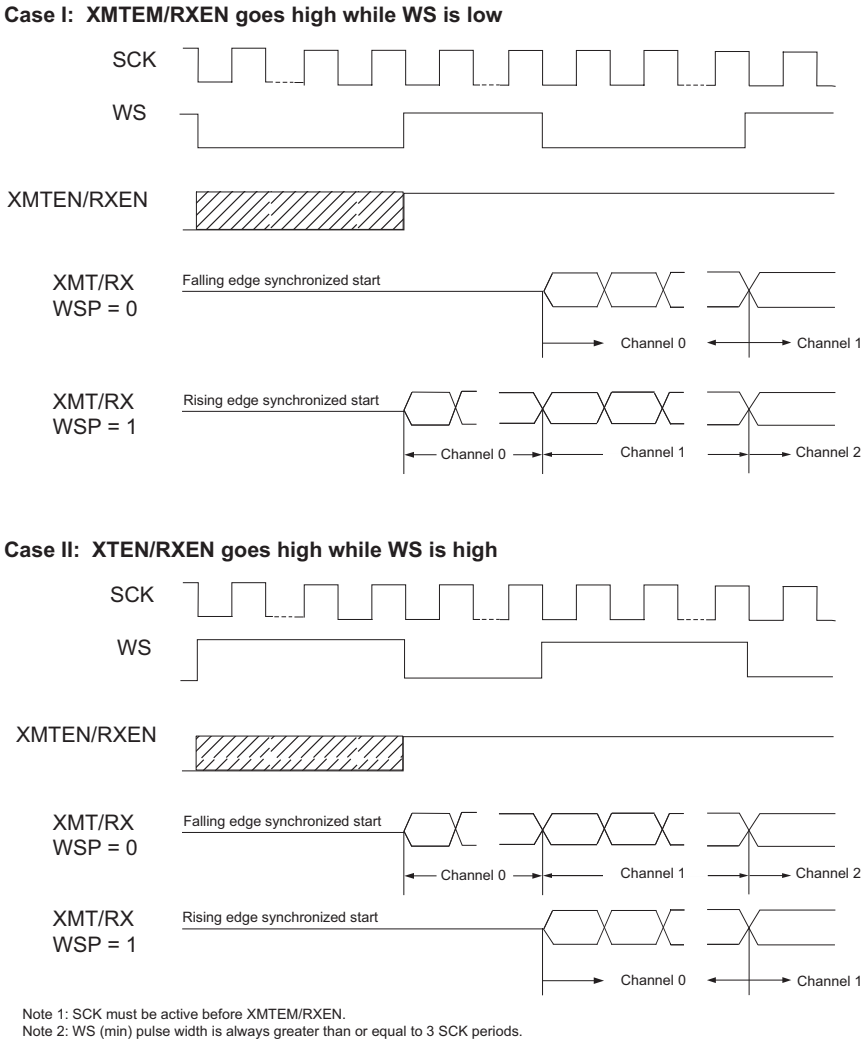



**Figure 1.117. Transmit and receive data (and WS) synchronized to SCK**

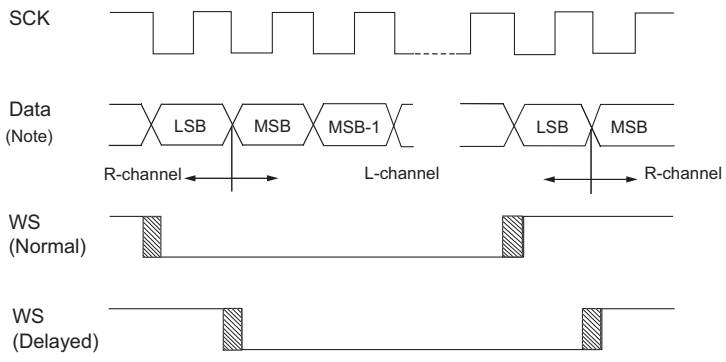


Downloaded from Arrow.com.




Note 1: SCK must be active before XMTEM/RXEN.
Note 2: WS (min) pulse width is always greater than or equal to 3 SCK periods.

**Figure 1.118. Transmit and receive data synchronized to WS**




Note: R channel and L channel are used to indicate channel change on WS edge only.

**Figure 1.119. WS normal or delayed transitions**



Downloaded from Arrow.com.

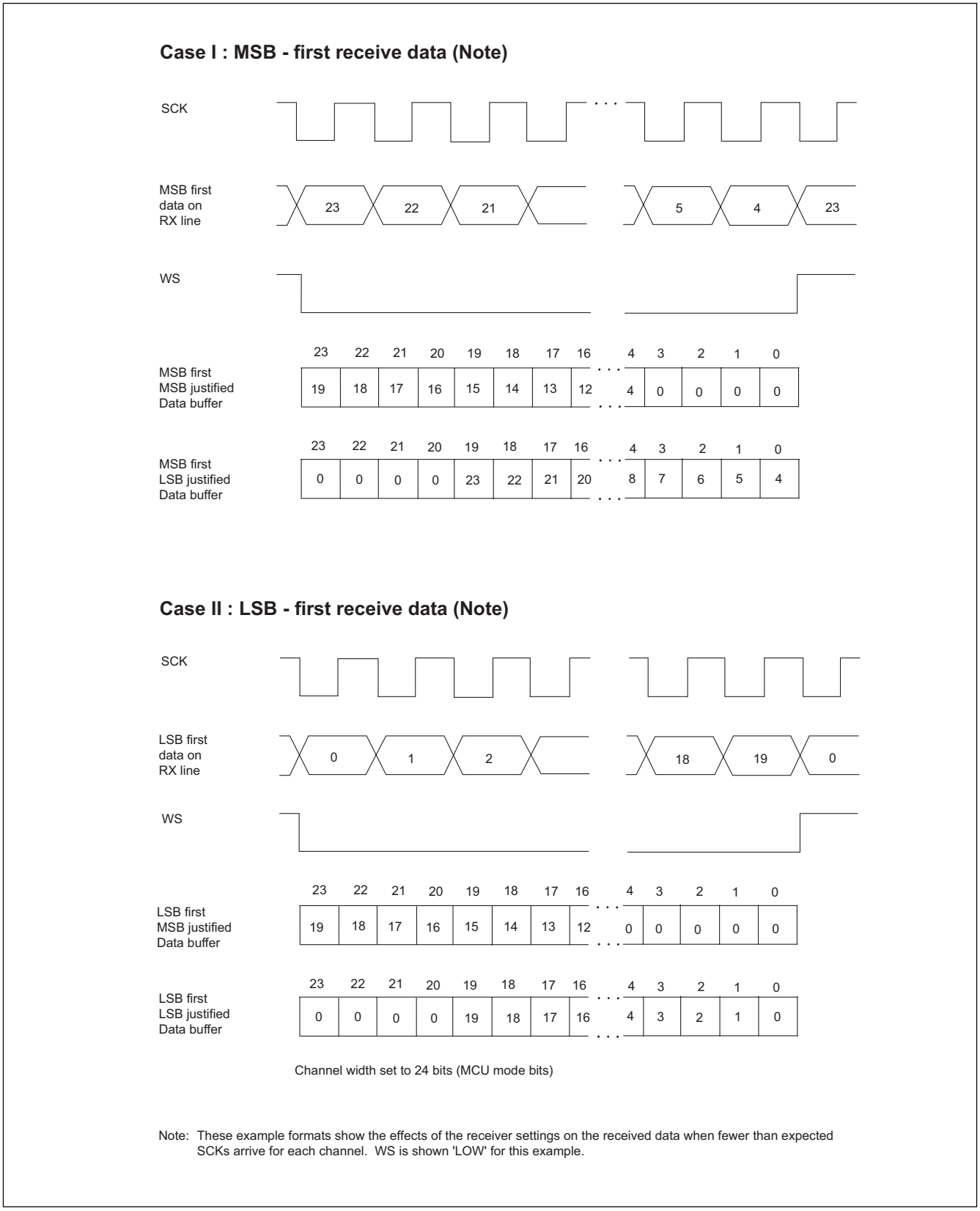



**Figure 1.120. Receiver setting effects**



Downloaded from Arrow.com.

**Overview**

The Serial Sound Interface is a serial data communication system. The parallel (MCU bus) to serial data conversion is accomplished by the shift registers. Figure 1.116 shows a description of each component of the Serial Sound Interface architecture. There are separate 32- bit shift registers for transmit and receive for full duplex operation. Each shift register can be configured for 32, 24, or 16 bits as defined by the channel width mode bits. The shift register loads (or stores for receiver) data from the data buffers on every WS edge. The first load and bit-shift begins on the first "valid" edge of WS (as defined by the mode bits) after the transmit/receive mode bits are set (see Figure 1.122). Therefore, the transmit data buffers must be loaded prior to enabling the transmitter to ensure that the first transmit contains valid data.

Both the transmitter and receiver have their own set of data buffers. There are two data buffers (left and right) so that, on special conditions when the MCU is handling a higher priority task, additional time is available (channel width X TSCK) before there is data underflow or overflow. The shift register always loads from or stores to the left buffer first and alternates between the two buffers on every edge of WS. The placement of data within the buffers is described in detail in the Data Path section.

The interrupt generator is a state machine which controls the data interface. The state machine makes the data transfer to or from the peripheral more efficient by generating interrupts until all the data needed for the data buffer has been accessed. The interrupt can be set up to be a DMA trigger for more efficient data transfer. The interrupt generator also tracks the read/write width (byte/word) so that no additional control is needed. The interrupt is first generated when a data word is loaded from the data buffer to the shift register (transmitter) or data are stored from the shift register into the data buffer (receiver). When the MCU is finished accessing (as a response to the interrupt) another interrupt is generated if the data buffer has not completely been accessed. For example, for a 24-bit transmitter, an interrupt is generated when the left buffer is loaded into the shift register. If the MCU writes a byte of data to the transmit buffer address, 8 of the 24 bits will be filled with new data. The interrupt generator triggers another interrupt causing the MCU to write more data. If this write is a 16-bit operation, no further interrupts are generated until the right buffer is loaded into the shift register. However, if the write operation is only 8 bit, then another interrupt is generated immediately.

The data interface is used to simplify the data transfer process. The data buffer address is the same regardless of the actual data buffer width. The interface places the incoming or outgoing data in the correct position according to the channel width and the number of completed reads/writes for the data buffer.

The operation of the data interface is demonstrated in the example below which is the case of 24-bit audio data with word writes. As previously explained, the state machine generates an interrupt when the left channel is loaded into the shift register for transmission. When the first word write occurs, the data interface places the data in the left buffer. Since 8 more bits are required to fully load the left buffer, another interrupt is generated. The MCU writes another word of data, of which one byte is placed in the left buffer. However, the remaining data is held in a temporary buffer within the data interface since the right channel may not be loaded into the shift register yet. If the data is not held in a temporary buffer but written to the right buffer, it would overwrite the untransmitted data in the right buffer. When the right buffer is eventually loaded into the shift register for transmission, the state machine generates an interrupt to request additional data. An MCU word write causes the data in the temporary buffer, as well as the data on the MCU data bus, to be placed in the right buffer. No further interrupts are generated because all data buffers are filled.



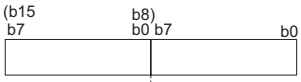
Downloaded from Arrow.com.

The data interface for the receiver behaves slightly different for the same case. When the receive shift register loads data into the left buffer, the state machine generates an interrupt. A word read from the MCU causes 16 bits to be read. The other 8 bits are latched into a temporary buffer. Latching the data into a temporary buffer empties the left buffer that provides additional time for the MCU to read the data without an overflow condition occurring. Even though there are unread bits, no interrupt is generated because a word read would read a byte from the right buffer which would be invalid data. The data in the right buffer is from the previous receive cycle and therefore invalid for the read cycle. Thus, the complete read of the left buffer is delayed until the right buffer is loaded by the receive shift register and the receive interrupts should be assigned higher priority than transmit if the Serial Sound Interface is set up for both transmit and receive.

The Serial Sound Interface also contains a rate feedback mechanism which can be used to determine the rate of data transfer via the Serial Sound Interface relative to the USB. It consists of a 16-bit counter with either SCK or WS as the count source and a 16-bit register to store the count value. The count value is loaded into the register on each negative edge of SOF pulse generated by the USB core. The counter is also reset by the SOF pulse. The SOF pulse is a frame delimiter used in USB communication. Refer to the USB section for details. The value read from the register is the count from the immediately preceding USB frame.

Figure 1.121 shows the Serial Sound Interface rate feedback registers and Serial Sound Interface transmit and receive data buffer registers. Figure 1.122. shows the Serial Sound Interface mode registers.

Serial Sound Interface transmit buffer register

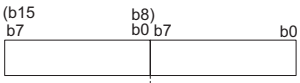



| Symbol | Address | When reset |
|---|---|---|
| SSIiTXB (i = 0, 1) | $0315_{16}$, $0314_{16}$<br>$0375_{16}$, $0374_{16}$ | $0000_{16}$ |

| Function | R | W |
|---|---|---|
| Transmit data (Note) | X | O |

Note: Write only to even byte (8 bit) or entire word (16 bit).

Serial Sound Interface receive buffer register

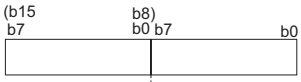



| Symbol | Address | When reset |
|---|---|---|
| SSIiRXB (i = 0, 1) | $0317_{16}$, $0316_{16}$<br>$0377_{16}$, $0376_{16}$ | $0000_{16}$ |

| Function | R | W |
|---|---|---|
| Receive data (Note) | O | X |

Note: Read only from even byte (8 bit) or entire word (16 bit)

Serial Sound Interface rate feedback register




| Symbol | Address | When reset |
|---|---|---|
| SSIiRF (i = 0, 1) | $0319_{16}$, $0318_{16}$<br>$0379_{16}$, $0378_{16}$ | $0000_{16}$ |

| Function | R | W |
|---|---|---|
| Rate feedback counter value | O | X |

**Figure 1.121. Serial Sound Interface related registers (1)**



Downloaded from Arrow.com.

Serial Sound Interface mode register 0

| Symbol | Address | When reset |
|---|---|---|
| SSIiMR0 (i = 0, 1) | $0310_{16}$, $0370_{16}$ | $00_{16}$ |

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SSIEN | Serial Sound Interface enable bit | 0 : Disable<br>1 : Enable | O | O |
| XMTEN | Transmitter enable bit | 0 : Disable<br>1 : Enable | O | O |
| RXEN | Receiver enable bit | 0 : Disable<br>1 : Enable | O | O |
| RFBEN | Rate feedback counter enable bit | 0 : Disable<br>1 : Enable | O | O |
| CWID0 | Channel width select bit 0 | b5 b4<br>0 0 : 32 bit<br>0 1 : 24 bit<br>1 0 : Reserved<br>1 1 : 16 bit | O | O |
| CWID1 | Channel width select bit 1 | | O | O |
| RFMT0 | Receiver format select bit 0 | 0 : LSB first<br>1 : MSB first | O | O |
| RFMT1 | Receiver format select bit 1 | 0 : LSB justified<br>1 : MSB justified | O | O |

Serial Sound Interface mode register 1

| Symbol | Address | When reset |
|---|---|---|
| SSIiMR1 (i = 0, 1) | $0311_{16}$, $0371_{16}$ | $00_{16}$ |

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | 0 | |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| XMTFMT | Transmit format select bit | 0 : LSB first<br>1 : MSB first | O | O |
| Reserved | | Always set to "0" | O | O |
| RFBSRC | Rate feedback counter source | 0 : SCK<br>1 : WS | O | O |
| SCKP | SCK polarity | 0 : Negative edge<br>1 : Positive edge | O | O |
| WSP | WS polarity | 0 : Negative edge<br>1 : Positive edge | O | O |
| WSDLY | WS delay | 0 : DelayedWS<br>1 : NormalWS | O | O |
| Reserved | | Always set to "0" | O | O |

**Figure 1.122. Serial Sound Interface related registers (2)**



Downloaded from Arrow.com.

### Data Path

The data path is designed to work with the USB on this device. Because the Serial Sound Interface is an audio interface, the USB audio class device specifications are used to define the data path. USB audio data can be multiple types: PCM, a-law, u-law, MPEG, AC-3, IEC 1937, etc. However, the data are always left (MSB) justified. '0's are padded to the LSB end to meet byte boundaries or packet size requirements. The USB data are transmitted as MSB first in standard formats. Therefore, for basic stereo data with 24-bit resolution ($L_{23}$ - $L_0$ and $R_{23}$ - $R_0$) should arrive or set up in the USB FIFO. Table 1.53 lists the USB FIFO data setup. Table 1.54 lists the Serial Sound Interface buffer data.

**Table 1.53. USB FIFO data setup**

| FIFO ADDRESS (offset from endpoint start address) | FIFO DATA | Comments |
|---|---|---|
| 0 | L7 - L0 | Sample 0 |
| 1 | L15 - L8 | |
| 2 | L23 - L16 | |
| 3 | R7 - R0 | |
| 4 | R15 - R8 | |
| 5 | R23 - R16 | |
| 6 | L7 - L0 | Sample 1 |
| 7 | L15 - L8 | |
| 8 | L23 - L16 | |
| 9 | R7 - R0 | |
| 10 | R15 - R8 | |
| 11 | R23 - R16 | |
| ... | ... | ... |
| ... | L7 - L0 | Sample n |
| ... | ... | ... |

**Table 1.54. Serial Sound Interface buffer data**

| Left Buffer | | | Right Buffer | | |
|---|---|---|---|---|---|
| Byte 2 | Byte 1 | Byte 0 | Byte 2 | Byte 1 | Byte 0 |
| L23 - L16 | L15 - L8 | L7 - L0 | R23 - R16 | R15 - R8 | R7 - R0 |



Downloaded from Arrow.com.

The USB FIFO is read using word accesses and each word is written to the transmit buffer. Table 1.55 lists the USB FIFO sequence operation. Note that DB refers to the MCU data bus.

**Table 1.55. USB FIFO sequence operation**

| OPERATION | Left Buffer | | | Right Bufferi | | |
|---|---|---|---|---|---|---|
| | **Byte 2** (L23-L16) | **Byte 1** (L15-L8) | **Byte 0** (L7-L0) | **Byte 2** (R23-R16) | **Byte 1** (R15-R8) | **Byte 0** (R7-R0) |
| First Word Write | | $DB_{15}$ - $DB_8$ | $DB_7$ - $DB_0$ | | | |
| Second Word Write | $DB_7$ - $DB_0$ | | | | | $DB_{15}$ - $DB_8$ |
| Third Word Write | | | | $DB_{15}$ - $DB_8$ | $DB_7$ - $DB_0$ | |

Data are placed in the buffer from the least significant byte to the most significant byte with the left buffer written first. If the write operation is a word, the lower order data bus bits (DB7-DB0) are treated as more significant than the higher order data bus bits (D15-DB8). This is compatible with the USB.
The same operation sequences occur for the receive buffer read. On a byte access, data are placed on the bus with the most significant byte first. If the access is a word read, the lower order data bus bits (DB7-DB0) are treated as more significant.

**Precautions**

- Entering wait mode with the SSI active can produce unpredictable data transfers. Make sure to disable the SSI transmitter and receiver before entering wait mode, and re-enable the transmitter and receiver after exiting wait mode.
- For flash memory version SSI transmission data must be latched as the following timing by a receiver.
  - SCKP=0 (falling edge) : within 3 BCLK cycles from the rising edge of SCK
  - SCKP=1 (rising edge) : within 3 BCLK cycles from the falling edge of SCK



Downloaded from Arrow.com.

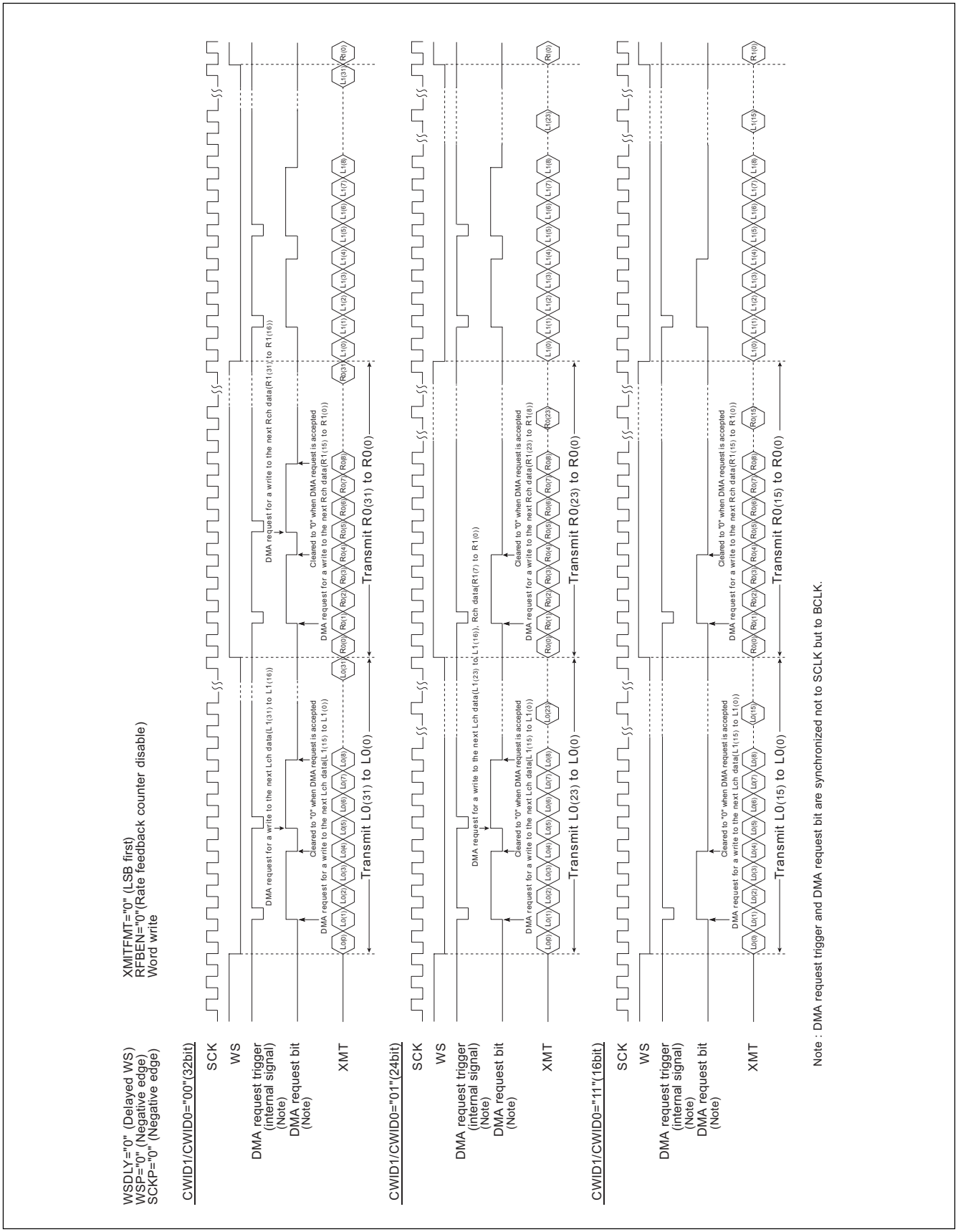



**Figure 1.123. DMA request timing in 32/24/16 bit width (transmission)**



Downloaded from Arrow.com.

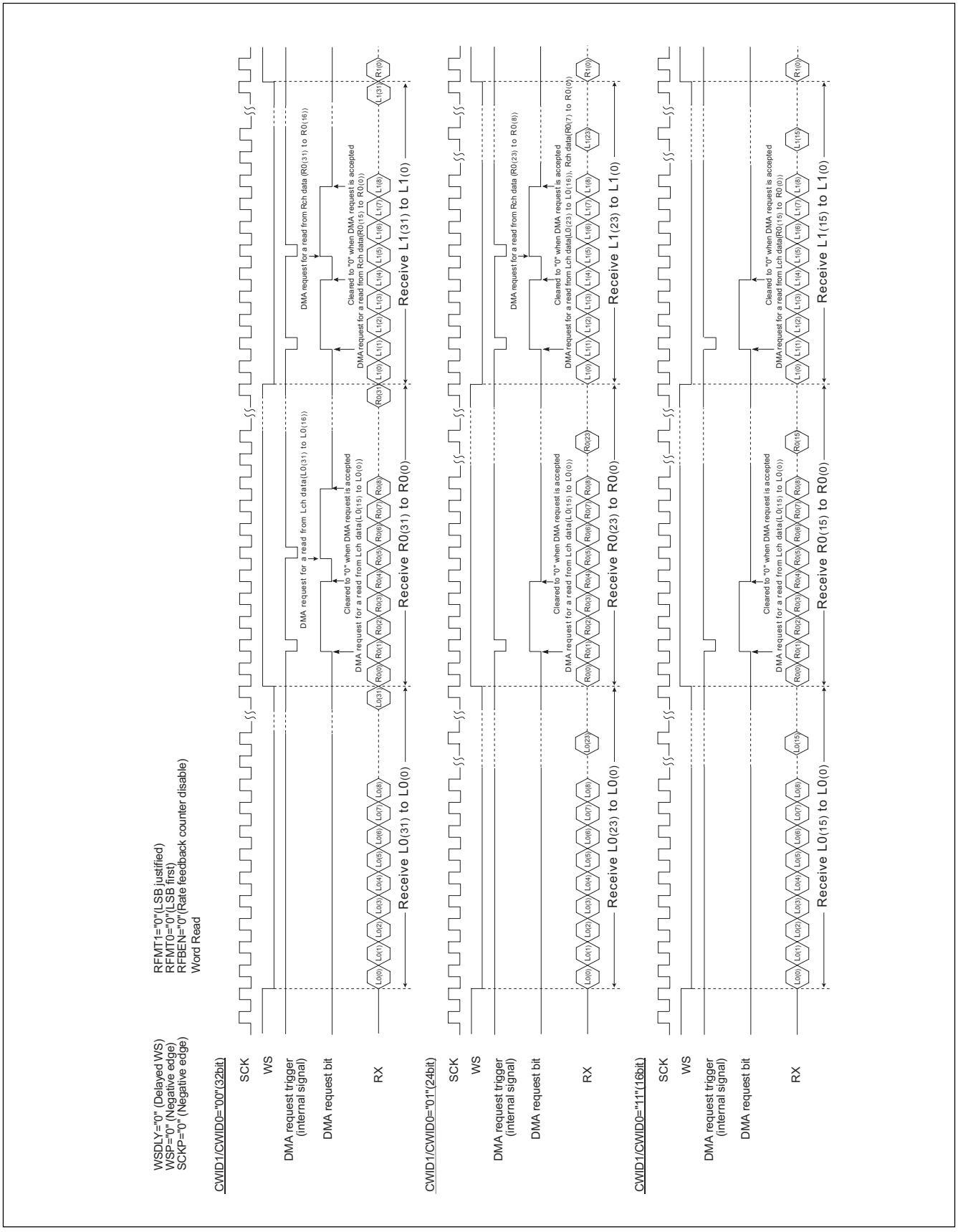



**Figure 1.124. DMA request timing in 32/24/16 bit width (reception)**



Downloaded from Arrow.com.

## A/D converter

The A/D converter consists of one 10-bit successive approximation A/D converter circuit with a capacitive coupling amplifier. Pins $P10_0$ to $P10_7$ function as the analog signal input pins. Set the direction registers corresponding to a pin with A/D conversion to input. The result of an A/D conversion is stored in the AD registers of the selected pins.
Table 1.56 shows the performance of the A/D converter. Figure 1.125 shows the block diagram of the A/D converter, and Figure 1.126 and Figure 1.127 show the A/D converter-related registers.

**Table 1.56. A/D Converter performance**

| Item | Performance |
|---|---|
| A/D conversion method | Successive approximation (capacitive coupling amplifier) |
| Analog input voltage (Note 1) | 0V to AVcc (Vcc) |
| Operating clock ΦAD (Note 2) | $f_{AD}$, $f_{AD}/2$, $f_{AD}/3$, $f_{AD}/4$ fAD=f(Xin) |
| Resolution | 8-bit or 10-bit (selectable) |
| Non-linear accuracy | |
| Operating modes | • One-shot mode<br>• Repeat mode<br>• Single sweep mode<br>• Repeat sweep mode 0<br>• Repeat sweep mode 1 |
| Analog input pins | 8 pins ($AN_0$ to $AN_7$) |
| A/D conversion start condition | • Software trigger<br>-A/D conversion startswhen the A/D conversion start flag changes to "1"<br>• External trigger (canberetriggered)<br>-A/D conversion startswhen $\overline{AD}_{\overline{TRG}}/P9_3$ input changes from "H" to "L" (Note 3) |
| Conversion speedper pin | • Without sample and hold function<br>8-bit resolution: 49 ΦAD cycles<br>10-bit resolution: 59 ΦAD cycles<br>• With sampleand hold function<br>8-bit resolution: 28 ΦAD cycles<br>10-bit resolution: 33 ΦAD cycles |

Note 1: Doesnot depend on use of sample and hold function.
Note 2: Whenf(Xin) is over 10 MHz, the ΦAD frequency must be set under 10MHz with the frequency select bits (bits 7 at $03D6_{16}$ and bit 4 at $03D7_{16}$).
Without the sample and hold function, set the ΦAD to 250kHz or higher.
With the sample and hold function, set the ΦAD frequency to 1 MHz or higher.
Note 3: Set the port direction register to input.



Downloaded from Arrow.com.

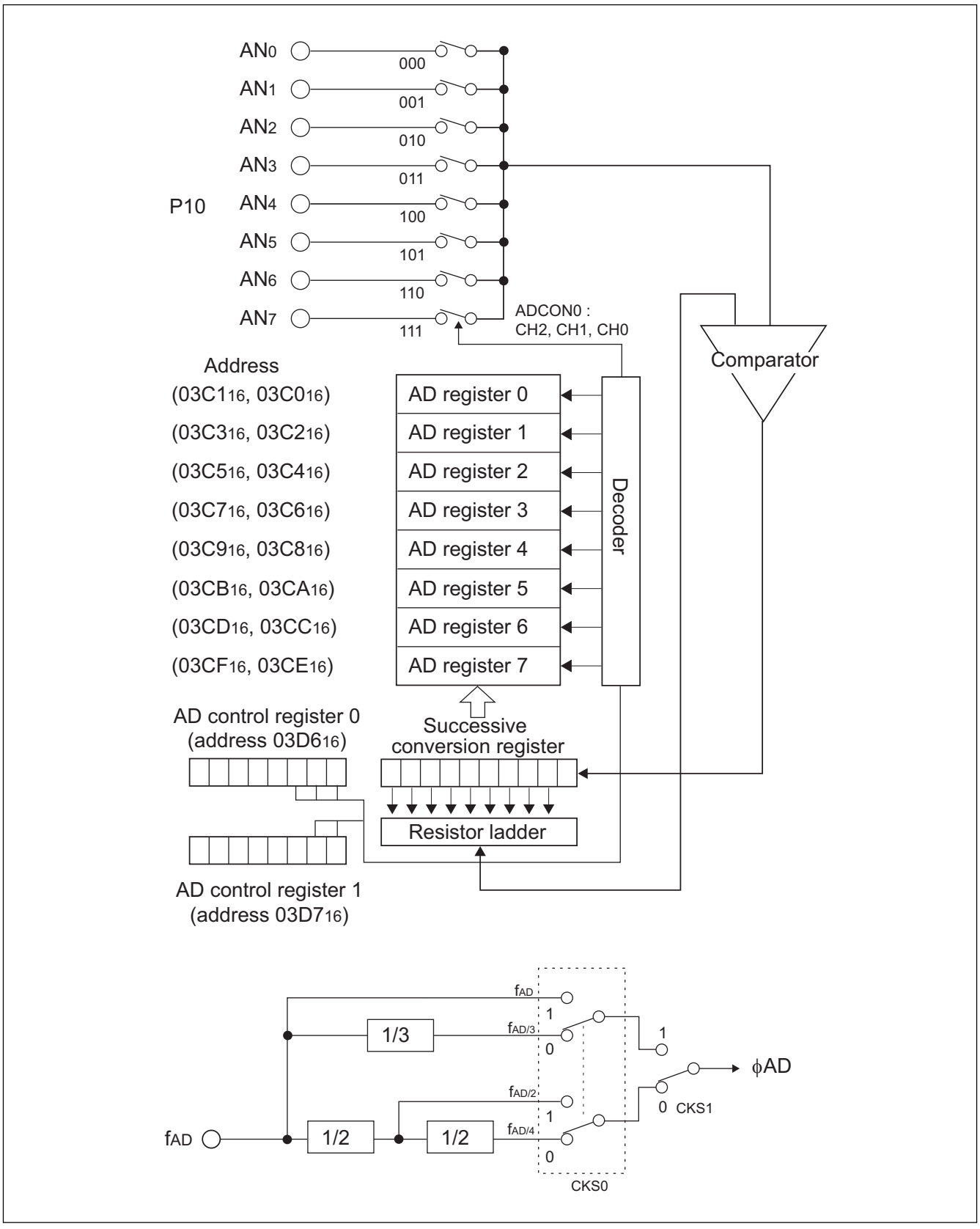



**Figure 1.125. A/D converter block diagram**



Downloaded from Arrow.com.

AD control register 0 (Note 1)




Symbol ADCON0 | Address $03D6_{16}$ | When reset $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | b2 b1 b0<br>0 0 0 : AN0<br>0 0 1 : AN1<br>0 1 0 : AN2<br>0 1 1 : AN3 (Note 2, 3)<br>1 0 0 : AN4<br>1 0 1 : AN5<br>1 1 0 : AN6<br>1 1 1 : AN7 | O | O |
| CH1 | | | O | O |
| CH2 | | | O | O |
| MD0 | A/D operation mode select bit 0 | b4 b3<br>0 0 : One-shot mode<br>0 1 : Repeat mode (Note 2)<br>1 0 : Single sweep mode<br>1 1 : Repeat sweep mode 0<br>Repeat sweep mode 1 | O | O |
| MD1 | | | O | O |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $\overline{AD_{TRG}}$ trigger | O | O |
| ADST | A/D conversion start flag | 0 : A/D conversion disabled<br>1 : A/D conversion enabled (Note 4) | O | O |
| CKS0 | Frequency select bit (Note 5) | 0 : fAD/3 or fAD/4 is selected<br>1 : fAD/1 or fAD/2 is selected | O | O |

Note 1: If the AD control regsiter 0 is rewritten during A/D conversion, the conversion result is indeterminate.
Note 2: When changing A/D operation mode, reset the analog input pin.
Note 3: This bit is disabled in single-sweep mode, repeat-sweep mode 0 and repeat-sweep mode 1.
Note 4: Set to "1" when $\overline{AD_{TRG}}$ is selected.
Note 5: When $f(X_{IN})$ exceeds 10 MHz, the AD frequency must be set less than 10 MHz by dividing.

AD control register 1 (Note 1)




Symbol ADCON1 | Address $03D7_{16}$ | When reset $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A/D sweep pin select bit | b1 b0<br>0 0 : AN0, AN1 (AN0)<br>0 1 : AN0 to AN3 (AN0, AN1) (Note 2)<br>1 0 : AN0 to AN5 (AN0 to AN2)<br>1 1 : AN0 to AN7 (AN0 to AN3) | O | O |
| SCAN1 | | | O | O |
| MD2 | A/D operation mode select bit 1 | 0 : Any mode other than repeat-sweep mode 1<br>1 : Repeat-sweep mode 1 | O | O |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | O | O |
| CKS1 | Frequency select bit (Note 3) | 0 : fAD/2 or fAD/4 is selected<br>1 : fAD/1 or fAD/3 is selected | O | O |
| VCUT | Vref connect bit | 0 : Vref not connected<br>1 : Vref connected | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate when read. | | | – | – |

Note 1: If the AD control regsiter 1 is rewritten during A/D conversion, the conversion result is indeterminate.
Note 2: This bit is invalid in one-shot mode and repeat mode. Channels shown in parentheses are valid when repeat-sweep mode 1 (bit 2 = "1") is selected.
Note 3: When $f(X_{IN})$ exceeds 10 MHz, the AD frequency must be set less than 10 MHz by dividing.

**Figure 1.126. A/D converter-related registers (1)**



Downloaded from Arrow.com.

AD control register 2 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| × | 0 | 0 | × | 0 | × | × | |

Symbol: ADCON2 — Address: $03D4_{16}$ — When reset: $X00X0XX0_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| SMP | A/D conversion method select bit | 0 :Without sample and hold<br>1 :With sample and hold | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate when read. | | | – | – |
| Reserved | | Must always be set to "0" | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate when read. | | | – | – |
| Reserved | | Must always be set to "0" | O | O |
| Nothing is assigned. Write "0" when writing to this bit. The value is indeterminate when read. | | | – | – |

Note : If the AD control register 2 is rewritten during A/D conversion, the conversion result is indeterminate.

AD register i (i = 0 to 7)

(b15) b7 (b8) b0 b7 b0

Symbol: ADi (i = 0 to 7) — Address: $03C0_{16}$ to $03CF_{16}$ — When reset: Indeterminate

| Function | R | W |
|---|---|---|
| Eight low-order bits of A/D conversion results | O | X |
| During 10-bit mode: Two high-order bits of A/D conversion results.<br>During 8-bit mode: The values are indeterminate when read. | O | X |
| Nothing is assigned.<br>Write "0" when writing to these bits. The values are indeterminate when read. | – | – |

**Figure 1.127. A/D converter-related registers (2)**



Downloaded from Arrow.com.

### One-shot mode

In one-shot mode, the pin selected using the analog input pin select bit is used for one-shot A/D conversion. Table 1.57 shows the specifications of one-shot mode.

**Table 1.57. One-shot mode specifications**

| Item | Specification |
|---|---|
| Function | The pin selected by the analog input pin select bit is used for one A/D conversion. |
| Start condition | Writing "1" to A/D conversion start flag, external trigger. |
| Stop condition | • End of A/D conversion (A/D conversion start flag changes to "0" except when external trigger is selected)<br>• Writing "0" to A/D conversion start flag. |
| Interrupt request generation timing | End of A/D conversion. |
| Input pin | One of $AN_0$ to $AN_7$, as selected. |
| A/D converter results | Read AD register corresponding to selected pin. |

### Repeat mode

In repeat mode, the pin selected using the analog input pin select bit is used for repeated A/D conversion. Table 1.58 shows the specifications of repeat mode.

**Table 1.58. Repeat mode specifications**

| Item | Specification |
|---|---|
| Function | The pin selected by the analog input pin select bit is used for repeated A/D conversion. |
| Start condition | Writing "1" to A/D conversion start flag, external trigger. |
| Stop condition | Writing "0" to A/D conversion start flag. |
| Interrupt request generation timing | None generated. |
| Input pin | One of $AN_0$ to $AN_7$, as selected. |
| A/D converter results | Read AD register corresponding to selected pin (at any time). |



Downloaded from Arrow.com.

### Single sweep mode

In single sweep mode, the pins selected using the A/D sweep pin select bit are used for one-by-one A/D conversion. Table 1.59 shows the specifications of single sweep mode.

**Table 1.59. Single sweep mode specifications**

| Item | Specification |
|---|---|
| Function | The pins selected by the A/D sweep pin select bit are used for one-by-one A/D conversion. |
| Start condition | Writing "1" to A/D converter start flag, external trigger. |
| Stop condition | • End of A/D conversion (A/D conversion start flag changes to "0" except when external trigger is selected).<br>• Writing "0" to A/D conversion start flag. |
| Interrupt request generation timing | End of Sweep. |
| Input pin | $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_3$ (4 pins), $AN_0$ to $AN_5$ (6 pins), or $AN_0$ to $AN_7$ (8 pins). |
| A/D converter results | Read AD register corresponding to selected pin. |

### Repeat sweep mode 0

In repeat sweep mode 0, the pins selected using the A/D sweep pin select bit are used for repeat sweep A/D conversion. Table 1.60 shows the specifications of repeat sweep mode 0.

**Table 1.60. Repeat sweep 0 specifications**

| Item | Specification |
|---|---|
| Function | The pins selected by the A/D sweep pin select bit are used for repeat sweep A/D conversion. |
| Start condition | Writing "1" to A/D conversion start flag. |
| Stop condition | Writing "0" to A/D conversion start flag. |
| Interrupt request generation timing | None generated. |
| Input pin | $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_3$ (4 pins), $AN_0$ to $AN_5$ (6 pins), or $AN_0$ to $AN_7$ (8 pins). |
| A/D converter results | Read AD register corresponding to selected pin (at any time). |



Downloaded from Arrow.com.

### Repeat sweep mode 1

In repeat sweep mode 1, all pins are used for A/D conversion with emphasis on the pin or pins selected using the A/D sweep pin select bit. Table 1.61 shows the specifications of repeat sweep mode 1.

**Table 1.61. Repeat sweep mode 1 specifications**

| Item | Specification |
|---|---|
| Function | All pinsperform repeat sweep A/D conversion with emphasis on the pin or pins selected by the A/D sweep pin select bit.<br>Example: $AN_0$ selected: $AN_0 \rightarrow AN_1 \rightarrow AN_0 \rightarrow AN_2 \rightarrow AN_0 \rightarrow AN_3$ etc. |
| Start condition | Writing "1" to A/D conversion start flag. |
| Stop condition | Writing "0" to A/D conversion start flag. |
| Interrupt request generation timing | None generated. |
| Input pin | $AN_0$ to $AN_7$. |
| Pin emphasis | $AN_0$ (1 pin) $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_2$ (3 pins), $AN_0$ to $AN_3$ (4 pins). |
| A/D converter results | Read AD register corresponding to selected pin (at any time). |

### Resolution select function

8 or 10-bit mode select bit of AD control register 1 (bit 3 at address $03D7_{16}$)
When set to 10-bit precision, the low 8-bits are stored in the even addresses and the high 2 bits in the odd addresses. When set to 8-bit precision, the low 8 bits are stored in the even addresses.

### Sample and hold

Sample and hold is selected by setting bit 0 of the AD control register 2 (address $03D4_{16}$) to "1". When sample and hold is selected, the rate of conversion of each pin increases. As a result, a 28 $f_{AD}$ cycle is achieved with 8-bit resolution and 33 $f_{AD}$ with 10-bit resolution. Sample and hold can be selected in all modes. However, in all modes, be sure to specify before starting A/D conversion whether sample and hold is to be used.

### Power consumption reduction function

The $V_{REF}$ connect bit (bit 5 at addresses $03D7_{16}$) can be used to isolate the resistance ladder of the A/D converter from the reference voltage pin ($V_{REF}$) when the A/D converter is not used. This stops any current from flowing into the resistance ladder from $V_{REF}$, reducing the power dissipation. When using the A/D converter, start A/D conversion only after connecting $V_{REF}$. Do not write A/D conversion start flag and $V_{REF}$ connect bit to "1" at the same time.

### Precautions

- Write to each bit (except bit 6) of AD control register 0, AD control register 1, and to bit 0 of AD control register 2 when A/D conversion is stopped (before a trigger occurs). When the $V_{REF}$ connection bit is changed from "0" to "1", wait 1 µs or longer before starting A/D conversion.

- When changing A/D operation mode, select the analog input pin again.



Downloaded from Arrow.com.

- Using one-shot mode or single sweep mode:

Read the corresponding AD register after confirming A/D conversion is finished. (Check the A/D conversion interrupt request bit.)

- Using repeat mode, repeat sweep mode 0 or repeat sweep mode 1:

Use the undivided main clock as the internal CPU clock.
When f(Xin) is faster than 10MHz, make the A/D frequency 10MHz or less by dividing.

### Output impedance of sensor at A/D conversion (Reference value).

To carry out A/D conversion properly, charging the internal capacitor C shown in Figure 1.126 has to be completed within a specified period of time T. Let the output impedance of sensor equivalent circuit be R0, the microcomputer's internal resistance be R, the precision (error) of the A/D converter be X, and the A/D converter's resolution be Y (Y is 1024 in the 10-bit mode, and 256 in the 8-bit mode).

$$\text{Vc is generally} = V_{IN}\{1 - e^{-\frac{t}{C(R0+R)}}\}$$

$$\text{And when } t = T,\quad Vc = V_{IN} - \frac{X}{Y}\ V_{IN} = V_{IN}\left(1 - \frac{X}{Y}\right)$$

$$e^{-\frac{T}{C(R0+R)}} = \frac{X}{Y}$$

$$-\frac{T}{C(R0+R)} = \ln\frac{X}{Y}$$

$$\text{Therefore, } R0 = -\frac{T}{C \cdot \ln\frac{X}{Y}} - R$$

With the model shown in Figure 1.128 as an example, when the difference between VIN and VC becomes 0.1LSB, we find impedance R0 when voltage between pins VC changes from 0 to VIN-(0.1/1024) VIN in time T. (0.1/1024) means that A/D precision drop due to insufficient capacitor charge is held to 0.1LSB at time of A/D conversion in the 10-bit mode. Actual error however is the value of absolute precision added to 0.1LSB. When f(Xin) = 10 MHz, T = 0.3 us in the A/D conversion mode with sample & hold. Output impedance R0 for sufficiently charging capacitor C within time T is determined as follows.

If T = 0.3μs, R = 7.8kΩ, C = 3pF, X = 0.1, and Y = 1024

$$\text{Then } R0 = -\frac{0.3 \times 10^{-6}}{3.0 \times 10^{-12} \cdot \ln\frac{0.1}{1024}} - 7.8 \times 10^{3} = \text{approximately } 3.0 \times 10^{3}$$

Thus, the allowable output impedance of the sensor circuit capable of thoroughly driving the A/D converter turns out to be approximately 3.0 kΩ. Table 1.62 and Table 1.63 show output impedance values based on the LSB values.

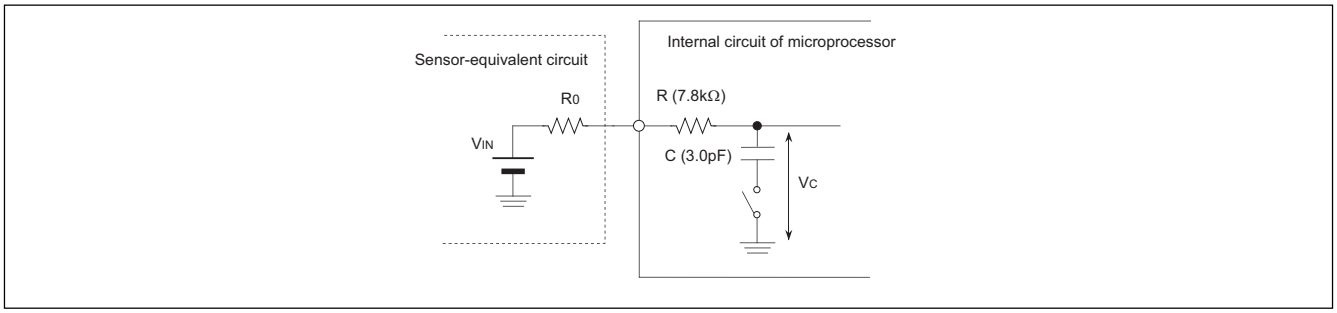



**Figure 1.128. A circuit equivalent to the A/D conversion terminal**



Downloaded from Arrow.com.

**Table 1.62. Output impedance values based on the LSB values (10-bit mode)**

| f($X_{IN}$) (MHz) | Cycle (μs) | T (Sampling time) | R (Kohm) | C(pF) | Resolution (LSB) | R0max (Kohm) |
|---|---|---|---|---|---|---|
| 10 | 0.1 | 0.3 (3 x cycle, sample and hold bit enabled | 7.8 | 3.0 | 0.1 | 3.0 |
| | | | | | 0.3 | 4.5 |
| | | | | | 0.5 | 5.3 |
| | | | | | 0.7 | 5.9 |
| | | | | | 0.9 | 6.4 |
| | | | | | 1.1 | 6.8 |
| | | | | | 1.3 | 7.2 |
| | | | | | 1.5 | 7.5 |
| | | | | | 1.7 | 7.8 |
| | | | | | 1.9 | 8.1 |
| 10 | 0.1 | 0.2 (2 x cycle, Sample and hold bit is enabled) | 7.8 | 3.0 | 0.3 | 0.4 |
| | | | | | 0.5 | 0.9 |
| | | | | | 0.7 | 1.3 |
| | | | | | 0.9 | 1.7 |
| | | | | | 1.1 | 2.0 |
| | | | | | 1.3 | 2.2 |
| | | | | | 1.5 | 2.4 |
| | | | | | 1.7 | 2.6 |
| | | | | | 1.9 | 2.8 |

**Table 1.63. Output impedance values based on the LSB values (8-bit mode)**

| f($X_{IN}$) (MHz) | Cycle (μs) | T (Sampling time) | R (Kohm) | C(pF) | Resolution (LSB) | R0max (Kohm) |
|---|---|---|---|---|---|---|
| 10 | 0.1 | 0.3 (3 x cycle, sample and hold bit enabled | 7.8 | 3.0 | 0.1 | 4.9 |
| | | | | | 0.3 | 7.0 |
| | | | | | 0.5 | 8.2 |
| | | | | | 0.7 | 9.1 |
| | | | | | 0.9 | 9.9 |
| | | | | | 1.1 | 10.5 |
| | | | | | 1.3 | 11.1 |
| | | | | | 1.5 | 11.7 |
| | | | | | 1.7 | 12.1 |
| | | | | | 1.9 | 12.6 |
| 10 | 0.1 | 0.2 (2 x cycle, Sample and hold bit is enabled) | 7.8 | 3.0 | 0.1 | 0.7 |
| | | | | | 0.3 | 2.1 |
| | | | | | 0.5 | 2.9 |
| | | | | | 0.7 | 3.5 |
| | | | | | 0.9 | 4.0 |
| | | | | | 1.1 | 4.4 |
| | | | | | 1.3 | 4.8 |
| | | | | | 1.5 | 5.2 |
| | | | | | 1.7 | 5.5 |
| | | | | | 1.9 | 5.8 |



Downloaded from Arrow.com.

## CRC calculation circuit

The Cyclic Redundancy Check (CRC) calculation circuit detects any errors in data blocks. The microcomputer uses a generator polynomial of CRC-CCITT ($x^{16}+ x^{12} + x^5 + 1$) or CRC-16 ($x^{16}+ x^{15} + x^2 + 1$) to generate CRC code.
The CRC code is a 16-bit code generated for a block of a given data length in multiples of 8 bits. It is set in a CRC data register every time one byte of data is transferred to a CRC input register after writing an initial value into the CRC data register. Generation of CRC code for one byte of data is completed in two machine cycles.
Figure 1.129 shows the block diagram of the CRC circuit. Figure 1.130 shows the CRC-related registers. Figure 1.131 shows an example of the CRC using CRC-CCITT.

### CRC Snoop

The CRC circuit includes the ability to snoop reads and writes to certain SFR addresses. This can be used to accumulate the CRC value on a stream of data without using extra bandwidth to explicitly write data into the CRCIN register. For example, it may be useful to snoop the writes to a UART TX buffer, or the reads from a UART RX buffer. This can only be used on USB, UART and SSI registers.
To snoop an SFR address, the target address is written to the CRC Snoop Address Register (CRCSAR). The two most significant bits of this register enable snooping on reads or writes to the target address. If the target SFR is written to by the CPU or DMA, and the CRC snoop write bit is set (CRCSW=1), the CRC will latch the data into the CRCIN register. The new CRC code will be set in the CRCD register.
Similarly, if the target SFR is read by the CPU or DMA, and the CRC snoop read bit is set (CRCSR=1), the CRC will latch the data from the target into the CRCIN register and calculate the CRC.
The CRC circuit can only calculate CRC codes on data one byte at a time. Therefore, if a target SFR is accessed in a word (16 bit) bus cycle, only the byte of data going to or from the target is snooped into CRCIN. The other byte of the word access is ignored.

Note: CRC Snoop can only be used to snoop USB, UART and SSI related SFR registers.

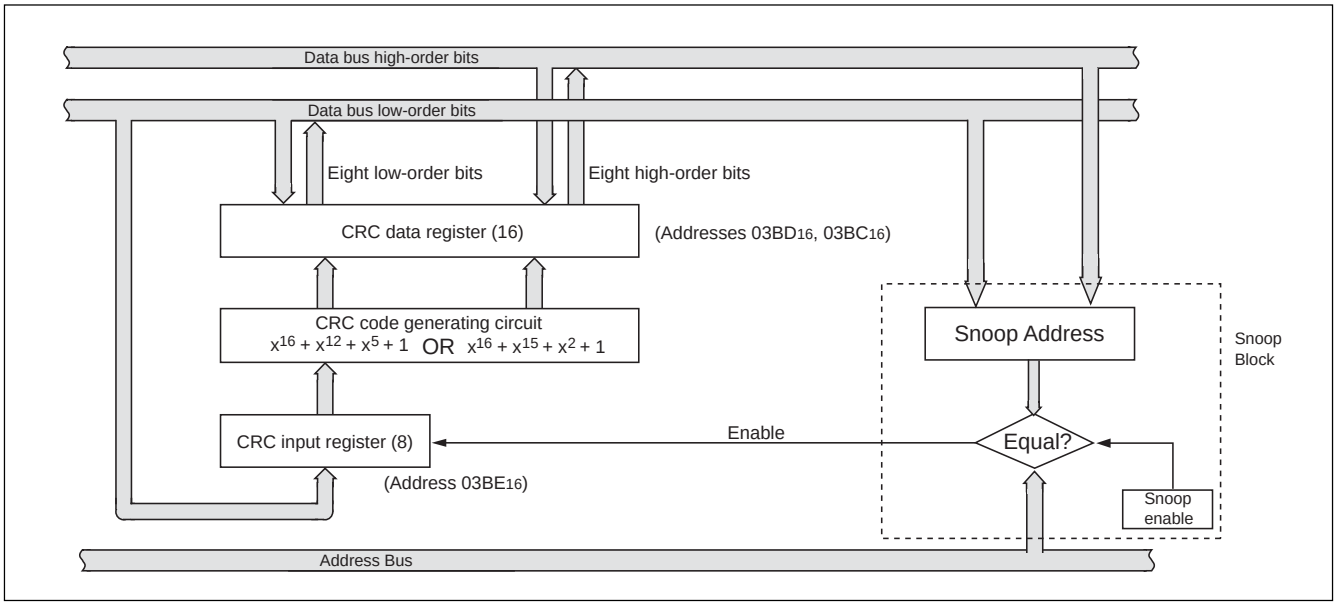
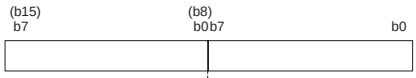



**Figure 1.129. CRC circuit block diagram**



Downloaded from Arrow.com.

CRC data register




| Symbol | Address | When reset |
|---|---|---|
| CRCD | $03BD_{16}$ to $03BC_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| CRC calculation result output | $0000_{16}$ to $FFFF_{16}$ | O | O |

CRC input register




| Symbol | Address | When reset |
|---|---|---|
| CRCIN | $03BE_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Data input | $00_{16}$ to $FF_{16}$ | O | O |

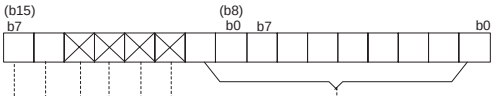
CRC mode register




| Symbol | Address | When reset |
|---|---|---|
| CRCMR | $03B6_{16}$ | 0XXXXXX0$_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CRCPS | CRC mode polynomial selection bit | 0 : $x^{16} + x^{12} + x^5 + 1$ (CRC-CCITT)<br>1 : $x^{16} + x^{15} + x^2 + 1$ (CRC-16) | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is indeterminate if read. | | | – | – |
| CRCMS | CRC mode selection bit | 0 : LSB first mode<br>1 : MSB first mode | O | O |

CRC snoop address register




| Symbol | Address | When reset |
|---|---|---|
| CRCSAR | $03B5_{16}$, $03B4_{16}$ | 00XXXX?? ????????$_2$ |

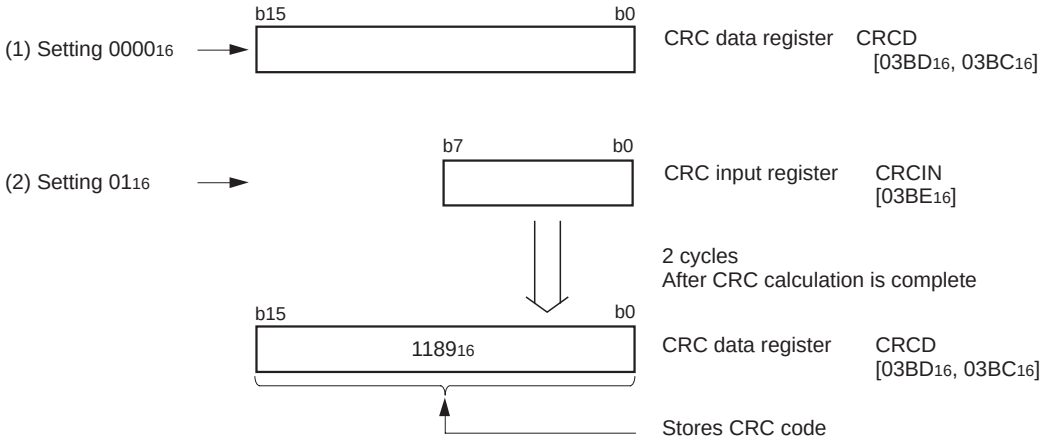
| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CRCSAR9-0 | CRC Snoop address bits | SFR address to snoop (Note) | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is indeterminate if read. | | | – | – |
| CRCSR | CRC Snoop on read enable bit | 0 : Disabled<br>1 : Enabled | O | O |
| CRCSW | CRC Snoop on write enable bit | 0 : Disabled<br>1 : Enabled | O | O |

Note: Only USB, UART and SSI related registers can be snooped

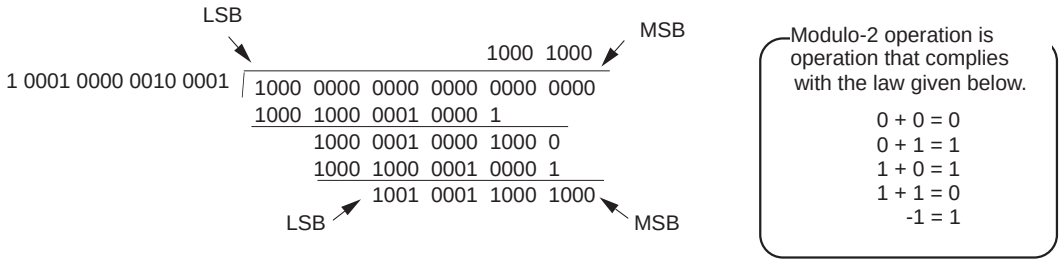
**Figure 1.130. CRC-related registers**



Downloaded from Arrow.com.




The code resulting from sending 01$_{16}$ in LSB first mode is (1000 0000). Thus the CRC code in the generating polynomial, ($X^{16} + X^{12} + X^5 + 1$), becomes the remainder resulting from dividing (1000 0000) $X^{16}$ by (1 0001 0000 0010 0001) in conformity with the modulo-2 operation.







Thus the CRC code becomes (1001 0001 1000 1000). Since the operation is in LSB first mode, the (1001 0001 1000 1000) corresponds to 1189$_{16}$ in hexadecimal notation. If the CRC operation in MSB first mode is necessary, set the CRC mode selection bit to "1". CRC data register stores CRC code for MSB first mode.

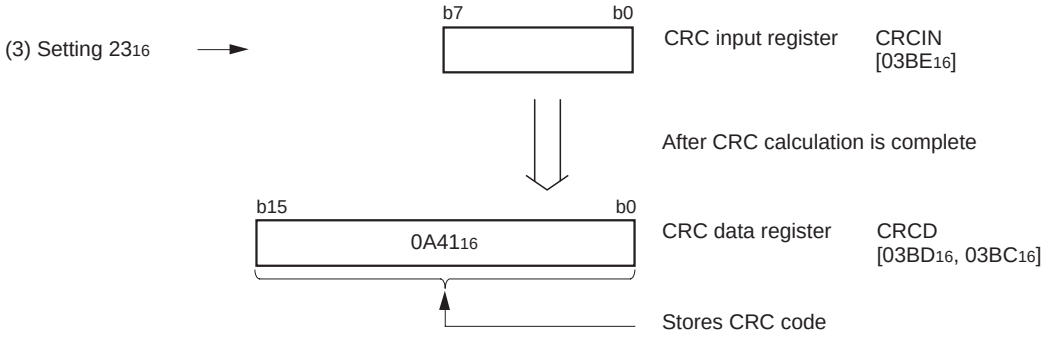



**Figure 1.131. CRC example using CRC-CCITT (LSB first mode)**



Downloaded from Arrow.com.

## Programmable I/O ports

There are 83 programmable I/O ports: P0 to P10 (excluding $P8_5$). Each port can be set independently for input or output using the direction register. A pull-up resistance for each block of 4 ports can be set. $P8_5$ is an input-only port and has no built-in pull-up resistance.
Figure 1.132 to Figure 1.135 show the programmable I/O ports. Figure 1.136 shows the I/O pins.
Each pin functions as a programmable I/O port and as the I/O for the built-in peripheral devices.
To use the pins as the inputs for the built-in peripheral devices, set the direction register of each pin to input mode.
When the pins are used as the outputs for the built-in peripheral devices, they function as outputs regardless of the contents of the direction registers. See the descriptions of the respective functions for how to set up the built-in peripheral devices.

### Direction registers

Figure 1.137 shows the direction registers.
These registers are used to choose the direction of the programmable I/O ports. Each bit in these registers corresponds one for one to each I/O pin.
Note: There is no direction register bit for $P8_5$.

### Port registers

Figure 1.138 shows the port registers.
These registers are used to write and read data for input and output to and from an external device. A port register consists of a port latch to hold output data and a circuit to read the status of a pin. Each bit in port registers corresponds one for one to each I/O pin.

### Pull-up control registers

Figure 1.139 shows the pull-up control registers.
The pull-up control register can be set to apply a pull-up resistance to each block of 4 ports. When ports are set to have a pull-up resistance, the pull-up resistance is connected only when the direction register is set for input.
However, in memory expansion mode and microprocessor mode, the pull-up control register of P0 to P3, $P4_0$ to $P4_3$, and P5 is invalid.

### High drive capacity register

Figure 1.140 shows the Port P7 drive capacity register. Port P7 can be configured to drive an LED by increasing the drive strength of the corresponding N-channel transistor bits.

### Port control register

Figure 1.141 shows the port control register.

The bit 0 of port control resister is used to read Port P1:

0: When Port P1 is an input port, the port input level is read.

When Port P1 is an output port, the contents of Port P1 register are read.

1: The contents of Port P1 register are always read.

This register is valid for the external bus width which is 8 bits in microprocessor mode or memory expansion mode.

### Unused pin connections

Table 1.64 lists an example of unused pins in single chip mode. Table 1.65 lists an example of unused pins in memory expansion mode. Figure 1.142 shows an example connection for unused pins.



Downloaded from Arrow.com.

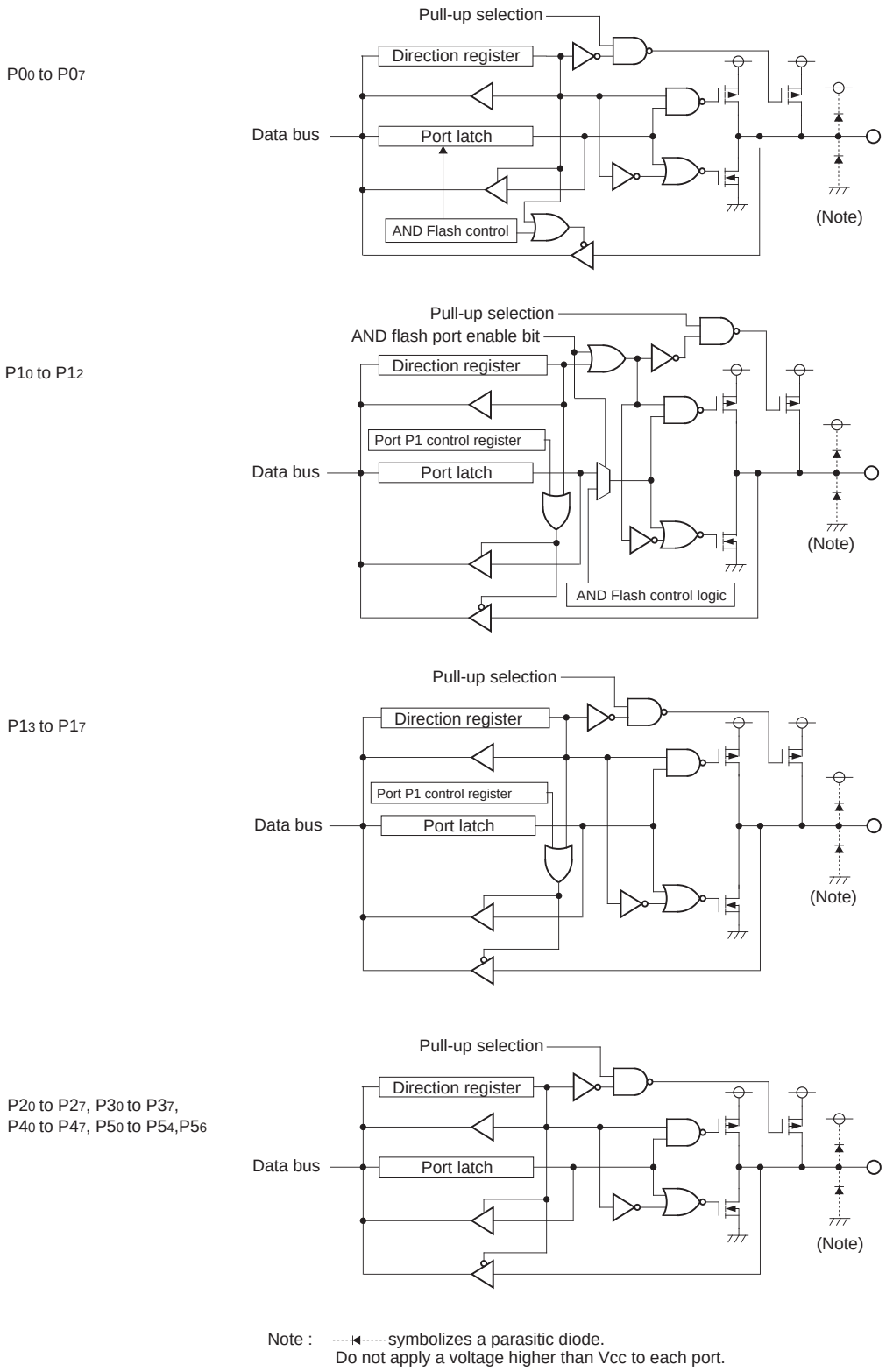
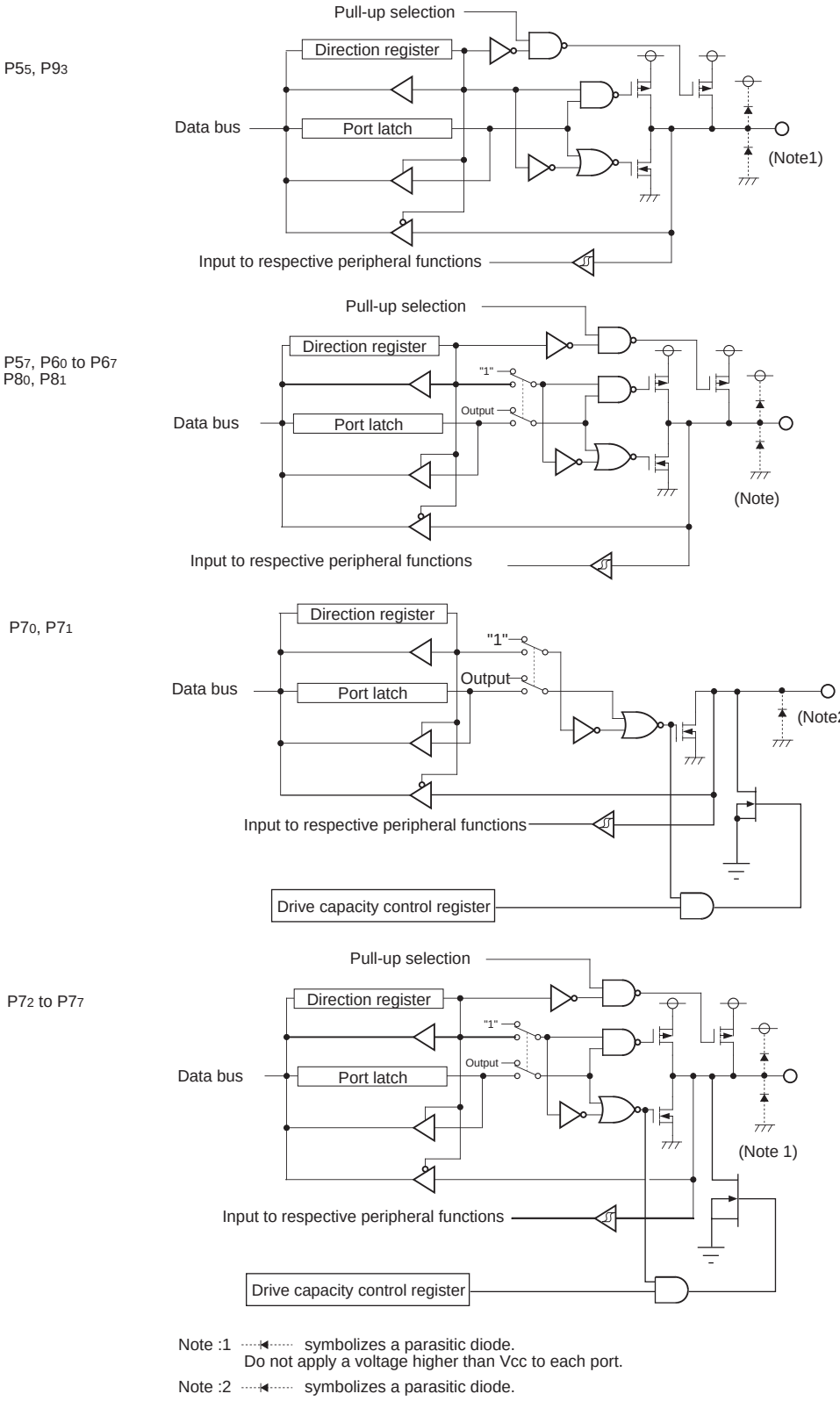



**Figure 1.132. Programmable I/O ports (1)**



Downloaded from Arrow.com.




Note :1 ····|◂···· symbolizes a parasitic diode.
Do not apply a voltage higher than Vcc to each port.

Note :2 ····|◂···· symbolizes a parasitic diode.

**Figure 1.133. Programmable I/O ports (2)**



Downloaded from Arrow.com.

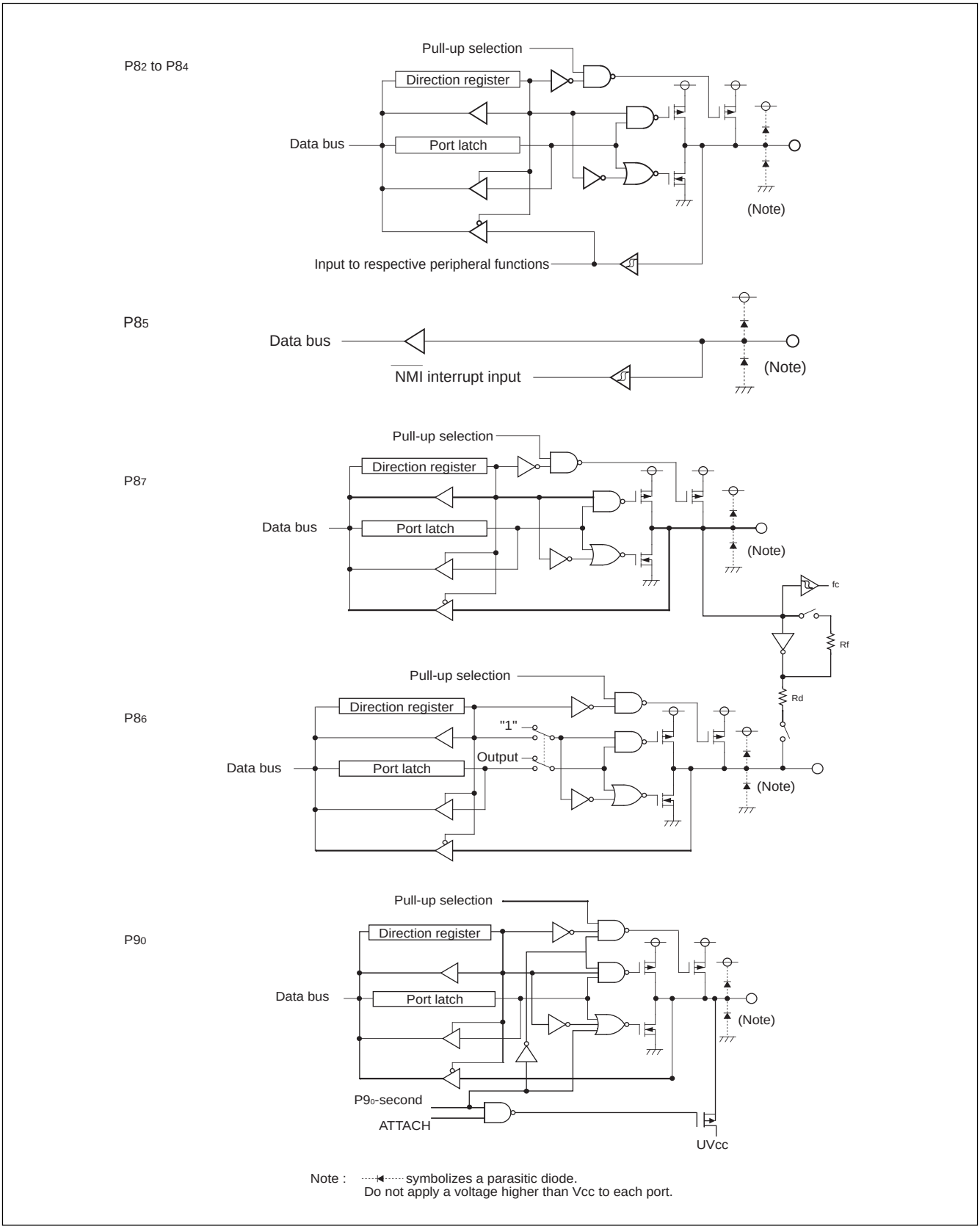



**Figure 1.134. Programmable I/O ports (3)**



Downloaded from Arrow.com.

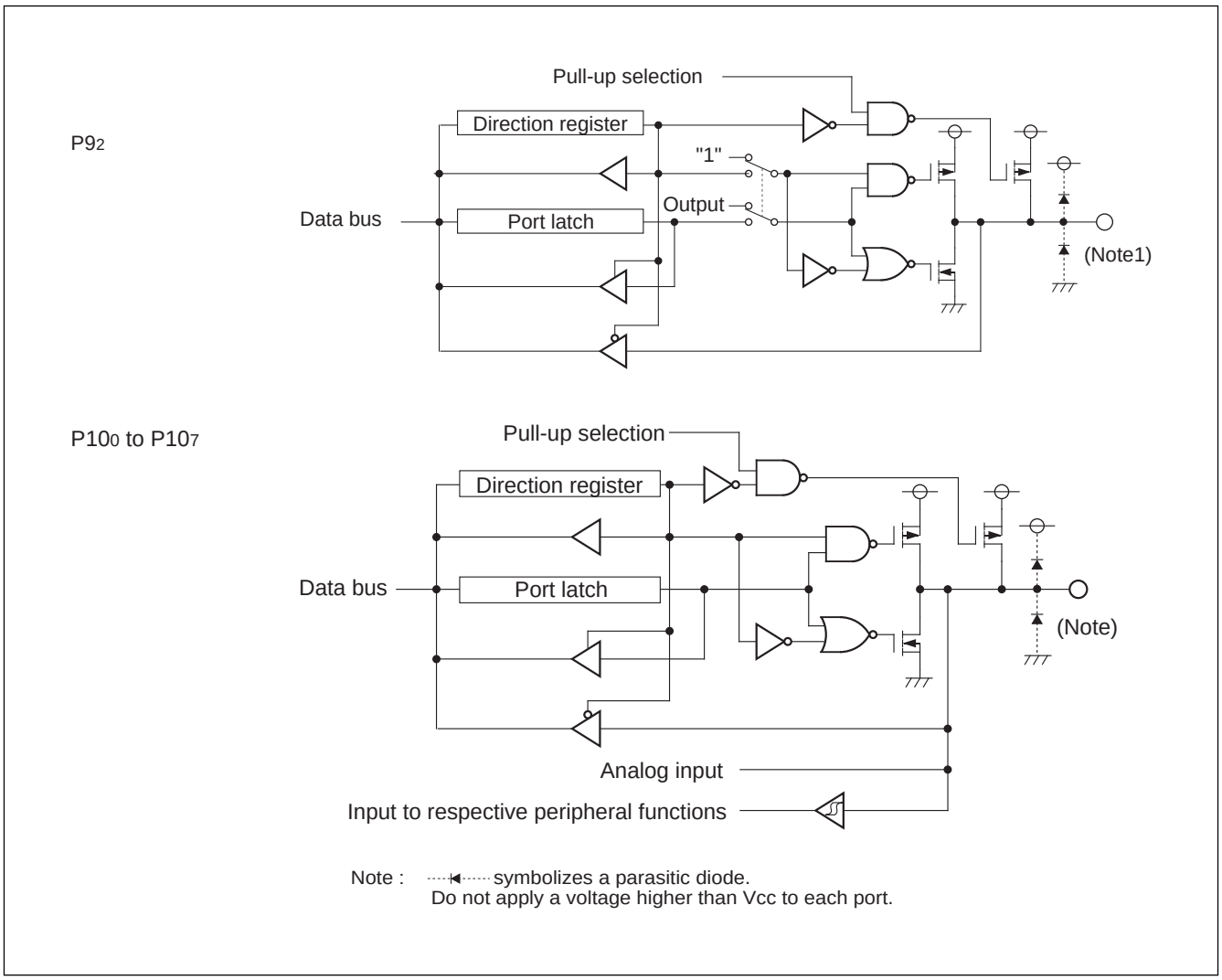



**Figure 1.135. Programmable I/O ports (4)**

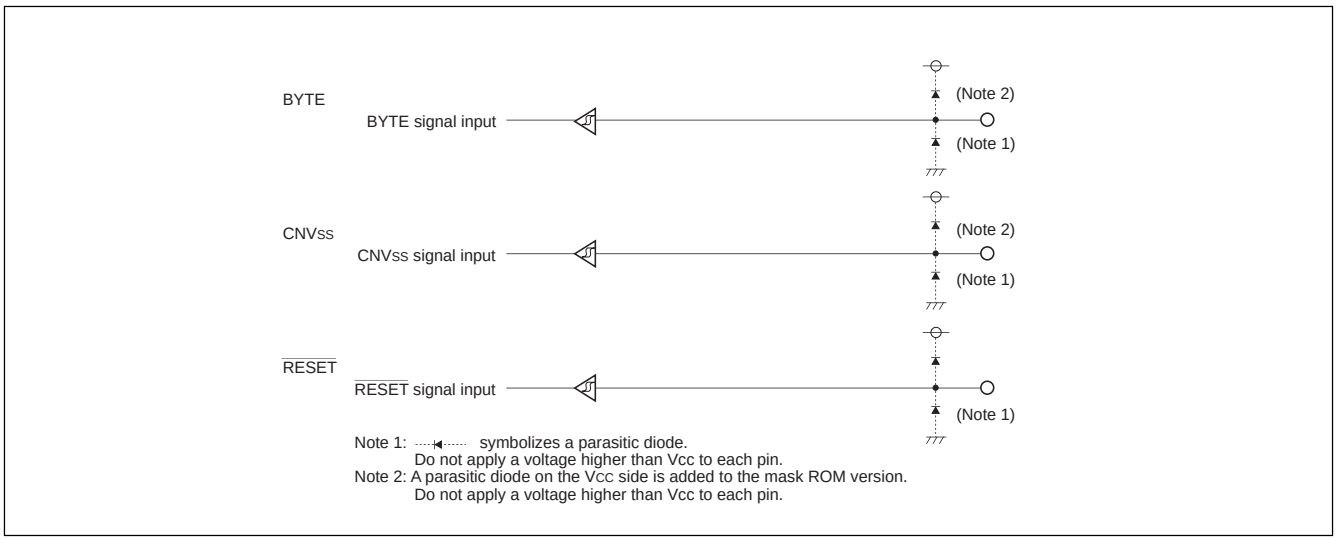



**Figure 1.136. I/O pins**



Downloaded from Arrow.com.

Port Pi direction register (Note)

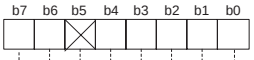



Symbol: PDi (i = 0 to 7, 10)
Address: $03E2_{16}$, $03E3_{16}$, $03E6_{16}$, $03E7_{16}$, $03EA_{16}$, $03EB_{16}$, $03EE_{16}$, $03EF_{16}$, $03F6_{16}$
When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PDi_0 | Port $Pi_0$ direction register | 0 : Input mode (Functions as an input port)<br>1 : Output mode (Functions as an output port) | O | O |
| PDi_1 | Port $Pi_1$ direction register | | O | O |
| PDi_2 | Port $Pi_2$ direction register | | O | O |
| PDi_3 | Port $Pi_3$ direction register | | O | O |
| PDi_4 | Port $Pi_4$ direction register | | O | O |
| PDi_5 | Port $Pi_5$ direction register | | O | O |
| PDi_6 | Port $Pi_6$ direction register | | O | O |
| PDi_7 | Port $Pi_7$ direction register | | O | O |

Note: In memory expansion mode and microprocessor mode, the contents of corresponding Port Pi direction register of pins $A_0$ to $A_{19}$, $D_0$ to $D_{15}$, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$ $\overline{HLDA}$ and BCLK cannot be modified.

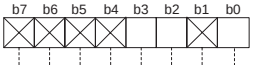
Port P8 direction register




Symbol: PD8
Address: $03F2_{16}$
When reset: $00X00000_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PD8_0 | Port $P8_0$ direction register | 0 : Input mode (Functions as an input port)<br>1 : Output mode (Functions as an output port) | O | O |
| PD8_1 | Port $P8_1$ direction register | | O | O |
| PD8_2 | Port $P8_2$ direction register | | O | O |
| PD8_3 | Port $P8_3$ direction register | | O | O |
| PD8_4 | Port $P8_4$ direction register | | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is indeterminate if read. | | | – | – |
| PD8_6 | Port $P8_6$ direction register | 0 : Input mode (Functions as an input port)<br>1 : Output mode (Functions as an output port) | O | O |
| PD8_7 | Port $P8_7$ direction register | | O | O |

Port P9 direction register




Symbol: PD9
Address: $03F3_{16}$
When reset: $XXXX00XX0_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PD9_0 | Port $P9_0$ direction register | 0 : Input mode (Functions as an input port)<br>1 : Output mode (Functions as an output port) | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is "0" if read. | | | – | – |
| PD9_2 | Port $P9_2$ direction register | 0 : Input mode (Functions as an input port)<br>1 : Output mode (Functions as an output port) | O | O |
| PD9_3 | Port $P9_3$ direction register | | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is "0" if read. | | | – | – |

**Figure 1.137. Direction registers**



Downloaded from Arrow.com.

Port Pi register (Note 2)




Symbol: Pi (i = 0 to 7, 10)
Address: $03E0_{16}$, $03E1_{16}$, $03E4_{16}$, $03E5_{16}$, $03E8_{16}$, $03E9_{16}$, $03EC_{16}$, $03ED_{16}$, $03F4_{16}$
When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| Pi_0 | Port $Pi_0$ register | Data is input and output to and from each pin by reading the writing to and from each corresponding bit.<br>0 : "L" level data<br>1 : "H" level data (Note 1) | O | O |
| Pi_1 | Port $Pi_1$ register | | O | O |
| Pi_2 | Port $Pi_2$ register | | O | O |
| Pi_3 | Port $Pi_3$ register | | O | O |
| Pi_4 | Port $Pi_4$ register | | O | O |
| Pi_5 | Port $Pi_5$ register | | O | O |
| Pi_6 | Port $Pi_6$ register | | O | O |
| Pi_7 | Port $Pi_7$ register | | O | O |

Note 1: Because $P7_0$ and $P7_1$ are N-channel open drain ports, the data are high-impedance.
Note 2: In memory expansion mode and microprocessor mode, the contents of corresponding port Pi register of pins $A_0$ to $A_{19}$, $D_0$ to $D_{15}$, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$ $\overline{HLDA}$ and BCLK cannot be modified.

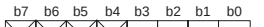
Port P8 register




Symbol: P8
Address: $03F0_{16}$
When reset: $00X00000_2$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| P8_0 | Port $P8_0$ register | Data is input and output to and from each pin by reading the writing to and from each corresponding bit.<br>0 : "L" level data<br>1 : "H" level data<br><br>(except $P8_5$) | O | O |
| P8_1 | Port $P8_1$ register | | O | O |
| P8_2 | Port $P8_2$ register | | O | O |
| P8_3 | Port $P8_3$ register | | O | O |
| P8_4 | Port $P8_4$ register | | O | O |
| P8_5 | Port $P8_5$ register | | O | X |
| P8_6 | Port $P8_6$ register | | O | O |
| P8_7 | Port $P8_7$ register | | O | O |

Port P9 register




Symbol: P9
Address: $03F1_{16}$
When reset: Indeterminate

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| P9_0 | Port $P9_0$ register | 0 : "L" level data<br>1 : "H" level data | O | O |
| VBDS | Vbus detect state bit | 0 : Not powered<br>1 : Powered (Note) | O | X |
| P9_2 | Port $P9_2$ register | 0 : "L" level data<br>1 : "H" level data | O | O |
| P9_3 | Port $P9_3$ register | 0 : "L" level data<br>1 : "H" level data | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is "0" if read. | | | – | – |

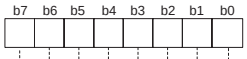
Note: This pin cannot be used for GPI/O. This bit reads "0" when Vbus detect is disabled.

**Figure 1.138. Port registers**



Downloaded from Arrow.com.

Pull-up control register 0 (Note)

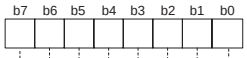



Symbol: PUR0 | Address: $03FC_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PU00 | $P0_0$ to $P0_3$ pull up | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | O | O |
| PU01 | $P0_4$ to $P0_7$ pull up | | O | O |
| PU02 | $P1_0$ to $P1_3$ pull up | | O | O |
| PU03 | $P1_4$ to $P1_7$ pull up | | O | O |
| PU04 | $P2_0$ to $P2_3$ pull up | | O | O |
| PU05 | $P2_4$ to $P2_7$ pull up | | O | O |
| PU06 | $P3_0$ to $P3_3$ pull up | | O | O |
| PU07 | $P3_4$ to $P3_7$ pull up | | O | O |

Note: In memory expansion and microprocessor mode, the contents of this register can be changed but the pull-up resistor is not connected.

Pull-up control register 1

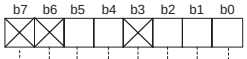



Symbol: PUR1 | Address: $03FD_{16}$ | When reset (Note 2): $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PU10 | $P4_0$ to $P4_3$ pull up (Note 3) | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | O | O |
| PU11 | $P4_4$ to $P4_7$ pull up | | O | O |
| PU12 | $P5_0$ to $P5_3$ pull up (Note 3) | | O | O |
| PU13 | $P5_4$ to $P5_7$ pull up | | O | O |
| PU14 | $P6_0$ to $P6_3$ pull up | | O | O |
| PU15 | $P6_4$ to $P6_7$ pull up | | O | O |
| PU16 | $P7_0$ to $P7_3$ pull up (Note 1) | | O | O |
| PU17 | $P7_4$ to $P7_7$ pull up | | O | O |

Note 1: Pull-up is not available for $P7_0$ and $P7_1$ because they are N-channel open drain ports.
Note 2: This register becomes $02_{16}$ when reset under the following conditions:
a) Hardware reset: when Vcc is applied to the CNVss pin.
b) Software reset: if bit 1 and bit 0 of processor mode register 0 (address $0004_{16}$) are $10_2$ or $11_2$ before reset.
Note 3: In memory expansion and microprocessor mode, the contents of this register can be changed but the pull-up resistor is not connected.

Pull-up control register 2




Symbol: PUR2 | Address: $03FE_{16}$ | When reset: $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PU20 | $P8_0$ to $P8_3$ pull up | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | O | O |
| PU21 | $P8_4$ to $P8_7$ pull up (except $P8_5$) | | O | O |
| PU22 | $P9_0$ to $P9_3$ pull up (except $P9_1$) | | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is "0" if read. | | | – | – |
| PU24 | $P10_0$ to $P10_3$ pull up | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | O | O |
| PU25 | $P10_4$ to $P10_7$ pull up | | O | O |
| Nothing is assigned.<br>Write "0" when writing to these bits. The value is "0" if read. | | | – | – |

**Figure 1.139. Pull-up control registers**



Downloaded from Arrow.com.

Port 7 drive capacity register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol P7DR | Address $03FA_{16}$ | When reset $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| P7DR0 | $P7_0$ LED drive capacity | The N-channel high drive capacity is activated for the corresponding bit.<br>0 : Normal drive<br>1 : N-channel high drive | O | O |
| P7DR1 | $P7_1$ LED drive capacity | | O | O |
| P7DR2 | $P7_2$ LED drive capacity | | O | O |
| P7DR3 | $P7_3$ LED drive capacity | | O | O |
| P7DR4 | $P7_4$ LED drive capacity | | O | O |
| P7DR5 | $P7_5$ LED drive capacity | | O | O |
| P7DR6 | $P7_6$ LED drive capacity | | O | O |
| P7DR7 | $P7_7$ LED drive capacity | | O | O |

**Figure 1.140. High drive capacity register**

Port control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Symbol PCR | Address $03FF_{16}$ | When reset $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PCR0 | Port P1 control register | 0 : When input port, read port input level. When output port, read the contents of Port P1 register.<br>1 : Read the contents of Port P1 register through input/output port. | O | O |
| OECTRL | AND Flash OE control bit | 0 : Data read mode enabled<br>1 : Output disabled | O | O |
| WECTRL | AND Flash WE control bit | 0 : Input disabled<br>1 : Command/Address mode enabled | O | O |
| AFPE | AND Flash port enable bit | 0 : P0 & P1(0-2) GPI/O function<br>1 : P0 & P1(0-2) AND Flash control function | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is"0" when read. | | | – | – |

**Figure 1.141. Port control register**

**Table 1.64. Example connection of unused pins in single-chip mode**

| Pin name | Connection |
|---|---|
| P0 to P10 (excluding P85) | After setting to input mode, connect every pin to Vss or Vcc using a resistor.<br>OR<br>Leave these pins open after setting to output mode. |
| Xout (Note 1) | Open |
| $\overline{\text{NMI}}$ | Connect using resistor to Vcc (pull-up) |
| UVcc, AVcc | Connect to Vcc |
| AVss, VREF, BYTE | Connect to Vss |
| USB D+, USB D-, LPF, VbusDTCT (Note 2) | Open |

Note 1: With external clock input to XIN pin.
Note 2: VbusDTCT pin is pulled down internaly.

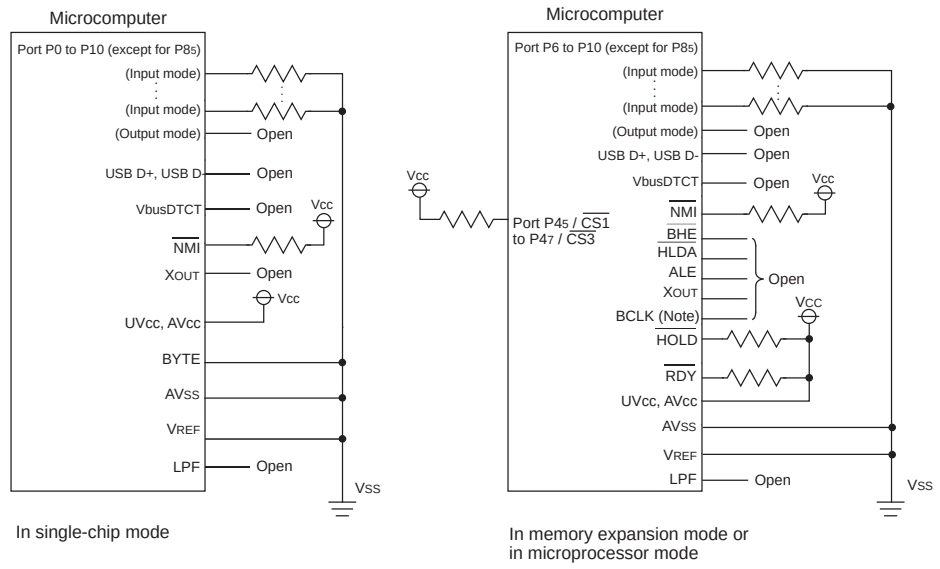


Downloaded from Arrow.com.

**Table 1.65. Example connection of unused pins in memory expansion mode**

| Pin name | Connection |
|---|---|
| P6 to P10 (excluding $P8_5$) | After setting to input mode, connect every pin to Vss or Vcc using a resistor.<br>OR<br>Leave these pins open after setting to output mode. |
| $P4_5/\overline{CS1}$ to $P4_7/\overline{CS3}$ | Set ports to input mode, set bits $\overline{CS1}$ to $\overline{CS3}$ to "0" (chip select disabled), connect to Vcc using resistors (pull-up) |
| $\overline{BHE}$, ALE, $\overline{HLDA}$, XOUT (Note 1), BCLK | Open |
| $\overline{HOLD}$, $\overline{RDY}$, $\overline{NMI}$ | Connect using a resistor to Vcc (pull-up) |
| UVcc, AVcc | Connect to Vcc |
| AVss, VREF, BYTE | Connect to Vss |
| USB D+, USB D-, LPF, VbusDTCT (Note 2) | Open |

Note 1: With external clock input to XIN pin.
Note 2: VbusDTCT pin is pulled down internaly.




Note : When the BCLK output disable bit (bit 7 at address 000416) is set to "1", connect to VCC using a pull-up resistor.

**Figure 1.142. Example connection of unused pins**

## Precautions

### Dedicated Input Pins

If a dedicated input pin is connected to a power supply different from the supply that Vcc is connected to, a resistor (approximately 1k ohm) should be added between the input pin and the connected power supply. However, if the dedicated input pin voltage is higher than Vcc, latch up could occur. A resistor is not required when using a Vcc voltage equal or greater than the voltage of the dedicated input pin.



Downloaded from Arrow.com.

## AND Flash Control Circuit

The AND flash control circuit is used for communicating with external AND type flash memory devices. The AND flash control circuit can be used only in single-chip mode. This circuit cannot be emulated by ICE. The Port Control Register (PCR), described by Figure 1.143, is used for overall control of this circuit. Setting bit AFPE to '1' assigns port pins P0$_0$-P0$_7$ and P1$_0$-P1$_2$ to function as signals necessary to interface with external flash memory. Along with their basic function, these activated signals are listed in Table 1.66, and described as follows:

**AND_DATA(7:0)** - These signals comprise the bus for input/output communication of data between the CPU and external flash memory. Upon circuit activation, the port P0 pins function as these signals. The port P0 direction register must be used to setup the direction of the AND_DATA(7:0) bus for input/output operation.

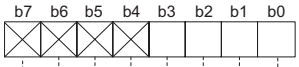
**AND_OE** - This signal is assigned to pin P1$_2$. Setting bit OECTRL to '1' will output a "L" pulse on this signal during each read from flash memory. When OECTRL is '0', AND_OE remains set "L".

**AND_WE** - This signal is assigned to pin P1$_1$. Setting bit WECTRL to '1' will output a "L" pulse on this signal during each write to flash memory. When WECTRL is '0', AND_WE remains set "H".

**AND_SC** - This signal is assigned to pin P1$_0$. With OECTRL set to '1' and WECTRL set to '0', a "H" pulse will be output on this signal during each flash memory write. If OECTRL is set to '0' and WECTRL set to '1', every read from flash memory will cause a "H" pulse to be output. The condition whereby both OECTRL and WECTRL are set to '1' results in AND_SC remaining set "L".

Figure 1.132 in the Programmable I/O section shows how the AND flash control circuitry is integrated with the port control logic for pins P0$_0$-P0$_7$ and P1$_0$-P1$_2$.

Port control register




Symbol PCR | Address $03FF_{16}$ | When reset $00_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| PCR0 | Port P1 control register | 0 : When input port, read port input level. When output port, read the contents of Port P1 register.<br>1 : Read the contents of Port P1 register through input/output port. | O | O |
| OECTRL | AND Flash OE control bit | 0 : Data read mode enabled<br>1 : Output disabled | O | O |
| WECTRL | AND Flash WE control bit | 0 : Input disabled<br>1 : Command/Address mode enabled | O | O |
| AFPE | AND Flash port enable bit | 0 : P0 & P1(0-2) GPI/O function<br>1 : P0 & P1(0-2) AND Flash control function | O | O |
| Nothing is assigned.<br>Write "0" when writing to this bit. The value is"0" when read. | | | – | – |

**Figure 1.143. Port control register**



Downloaded from Arrow.com.

Figure 1.144 shows an example of how to connect an AND type flash memory to the M30245 AND Flash Conntrol circuit.

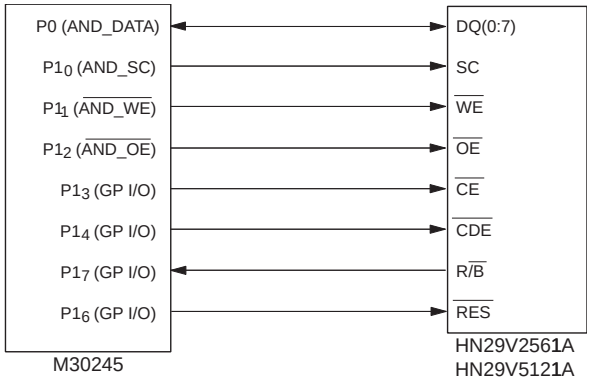



**Figure 1.144. Example connections to AND flash memory**

**Table 1.66. AND flash function table**

| WECTL, OECTL | $\overline{\text{AND_OE}}$ | $\overline{\text{AND_WE}}$ | AND_SC |
|---|---|---|---|
| 00 | Inhibited | | |
| 01 | "L" pulse during AND_DATA read cycle | "H" | "H" pulse during AND_DATA write cycle |
| 10 | "L" | "L" pulse during AND_DATA write cycle | "H" pulse during AND_DATA read cycle |
| 11 | "L" pulse during AND_DATA read cycle | "L" pulse during AND_DATA write cycle | "L" |

### Sample AND Flash Control Algorithms

Figures 1.145 and 1.146 show flow charts describing sample read and write (program) operations of AND flash memory. Please consult the M5M29F5611VP AND flash memory product specification for a detailed description of it's design and control.



Downloaded from Arrow.com.

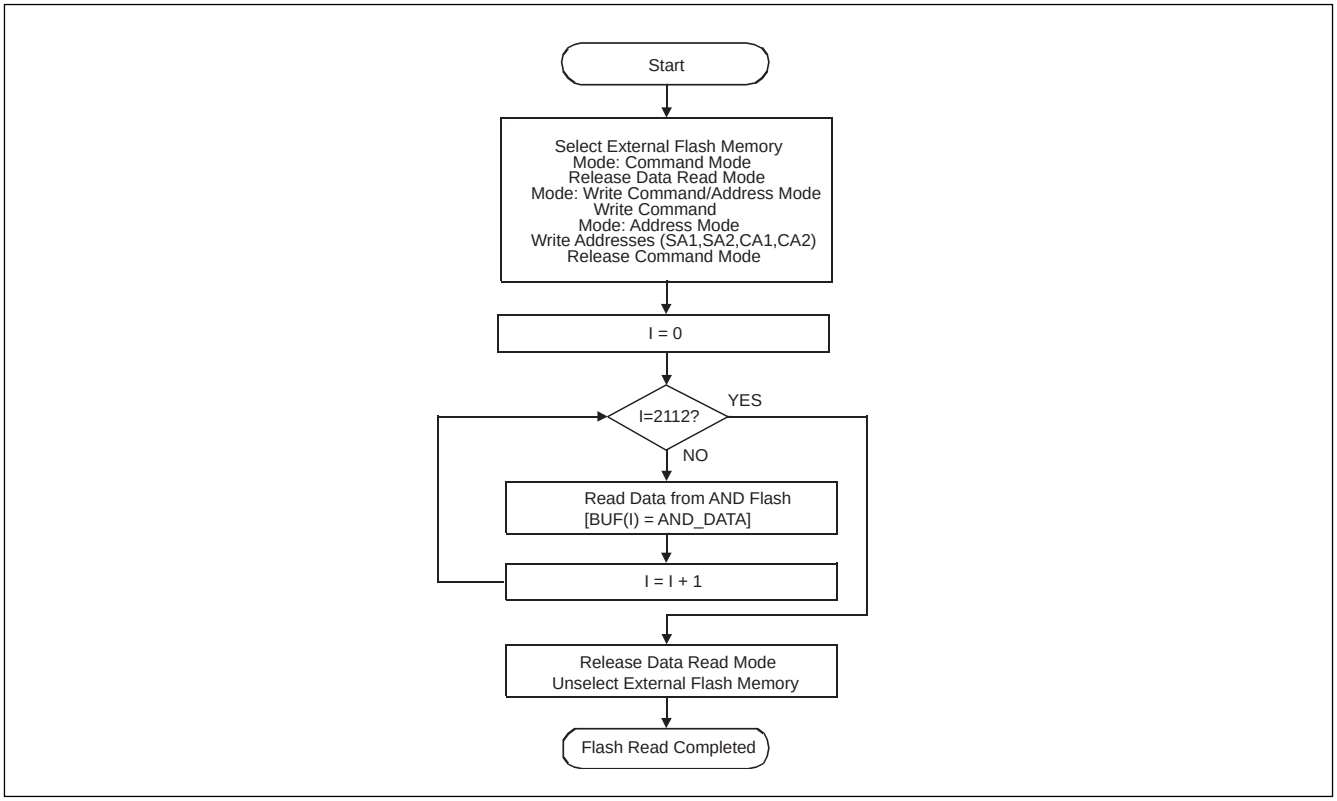



**Figure 1.145. AND flash read algorithm**

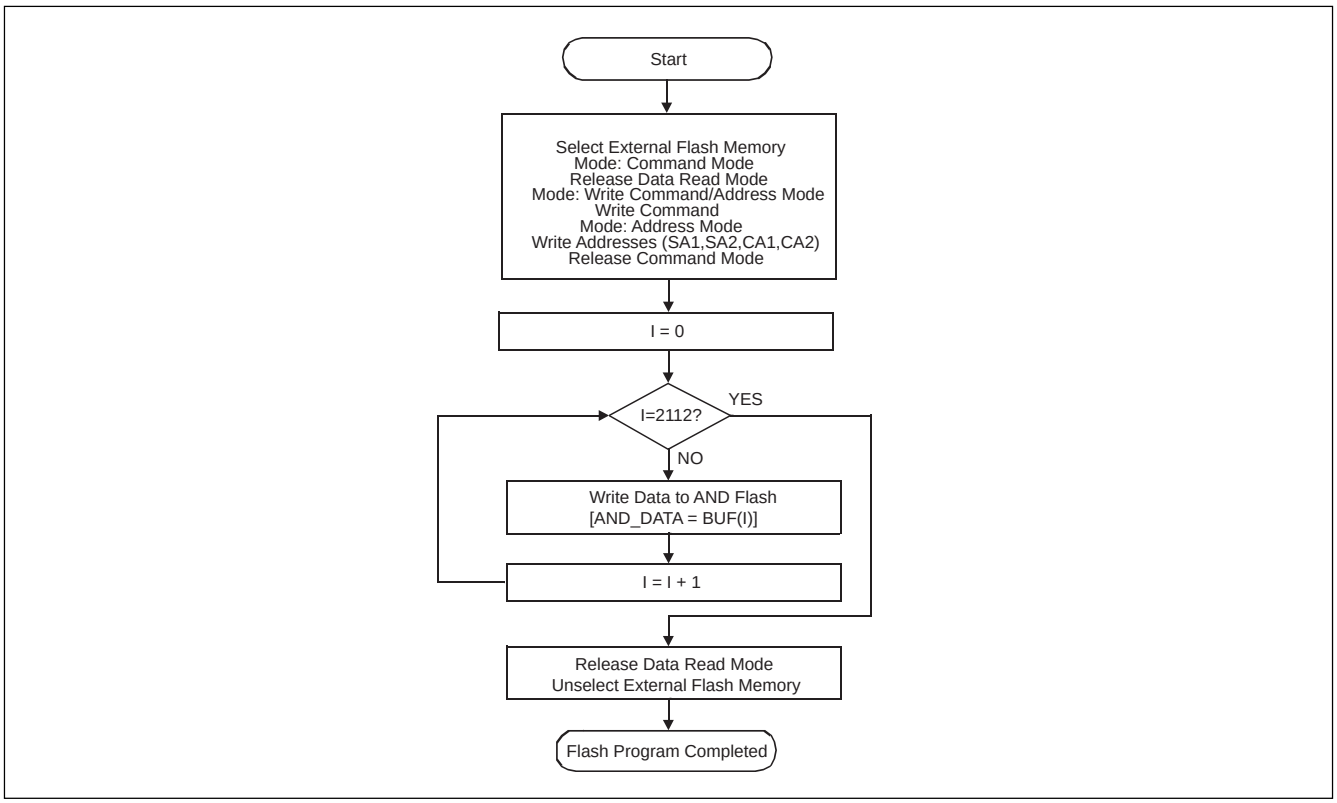



**Figure 1.146. AND flash write algorithm**



Downloaded from Arrow.com.

### Sample AND Flash Code

Figures 1.147 and 1.148 show sample code segments of AND flash read and write (program) assembly routines.

```
;    Test Read Access to AND Flash
;
          MOV.B   #03EH, P1          ; Initialize Port 1
          MOV.B   #07FH, PD1         ; Initialize Port 1
          MOV.B   #00AH, PCR         ; Initialize AND_Flash Control Register
          MOV.B   #040H, P1          ; Release AND_Flash RESET
          BCLR    3, P1              ; Select External Flash memory
          BCLR    4, P1              ; Mode : Command Mode
          BSET    1, PCR             ; Release Data Read Mode
          BSET    2, PCR             ; Mode : Write Command / Address Mode
          MOV.B   #000H, P0          ; Write Command 00 = Read
          BSET    4, P1              ; Mode : Address Mode
          MOV.B   #034H, P0          ; Sector Address 1
          MOV.B   #012H, P0          ; Sector Address 2
          MOV.B   #000H, P0          ; Column Address 1
          MOV.B   #000H, P0          ; Column Address 2
          BCLR    1, PCR             ; Mode : Data Read Mode
RYBY02:
          BTST    7, P1              ; Wait to RY/BYB = 0
          JNE     RYBY02
RYBY12:
          BTST    7, P1              ; Wait to RY/BYB = 1
          JEQ     RYBY12
          MOV.W   #$0000H, A0        ; Initialize A0
READDATA:
          MOV.B   P0, RAMAD[A0]      ; Read Data
          CMP.W   #0083FH, A0        ; I = 2112?
          JEQ     READEND
          INC.W   A0                 ; I = I + 1
          JMP     READDATA
READEND:
          BSET    1, PCR             ; Release Data Read Mode
          BSET    3, P1              ; Unselect External Flash memory
```

**Figure 1.147. AND Flash read example program**



Downloaded from Arrow.com.

```
;       Test Write Access to AND Flash
;
            MOV.B   #03EH, P1               ; Initialize Port 1
            MOV.B   #07FH, PD1              ; Initialize Port 1
            MOV.B   #00AH, PCR              ; Initialize AND_Flash Control Register
            MOV.B   #040H, P1               ; Release AND_Flash RESET
            BCLR    3, P1                   ; Select External Flash memory
            BCLR    4, P1                   ; Mode : Command Mode
            BSET    1, PCR                  ; Release Data Read Mode
            BSET    2, PCR                  ; Mode : Write Command / Address Mode
            MOV.B   #011H, P0               ; Write Command 11 = Program 4
            BSET    4, P1                   ; Mode : Address Mode
            MOV.B   #034H, P0               ; Sector Address 1
            MOV.B   #012H, P0               ; Sector Address 2
            MOV.B   #000H, P0               ; Column Address 1
            MOV.B    #000H, P0              ; Column Address 2
            BCLR    2, PCR                  ; Release Command Mode
RYBY03:
            BTST    7, P1                   ; Wait to RY/BYB = 0
            JNE     RYBY03
RYBY13:
            BTST    7, P1                   ; Wait to RY/BYB = 1
            JEQ     RYBY13
            BCLR    4, P1                   ; Mode : Data Entry Mode
            MOV.W   #$0000H, A0
TRANSDATA:
            MOV.B   RAMAD[A0], P0
            CMP.W   #0083FH, A0
            JEQ     TRANSEND
            INC.W   A0
            JMP     TRANSDATA
TRANSEND:
            BSET    2, PCR                  ; Mode : Write Command / Address Mode
            MOV.B   #040H, P0               ; Auto Program Data
            BSET    4, P1                   ; Mode : CDE=H
RYBY13B:
            BTST    7, P1                   ; Wait to RY/BYB = 1
            JEQ     RYBY13B
            BSET    1, PCR                  ; Release Data Read Mode
            BSET    3, P1                   ; Unselect External Flash memory
```

**Figure 1.148 AND Flash write example program**



Downloaded from Arrow.com.

## Flash memory

The M30245FC contains flash memory that can be rewritten with a single voltage of 3.3 V. Three flash memory modes are available to read, program, and erase:

• CPU rewrite mode in which the flash memory can be manipulated by the Central Processing Unit (CPU).

• Parallel I/O and standard serial I/O modes can be manipulated using a programmer

The flash memory is divided into several blocks as shown in Figure 1.149.
Memory can be erased one block at a time. Each block has a lock bit to enable or disable execution of an erase or program operation. This allows data in each block to be protected. Table 1.67 shows an overview of the M30245 (flash memory version).
In addition to the ordinary user ROM area that stores the microcomputer operation program, the flash memory has a boot ROM area that stores a program to control rewriting in CPU rewrite and standard serial I/O modes. The boot ROM area has a standard serial I/O mode control program stored in it when it is shipped from the factory. However, the user can write a CPU rewrite control program in this area specific to the user's application system. The boot ROM area can only be rewritten in parallel I/O mode.

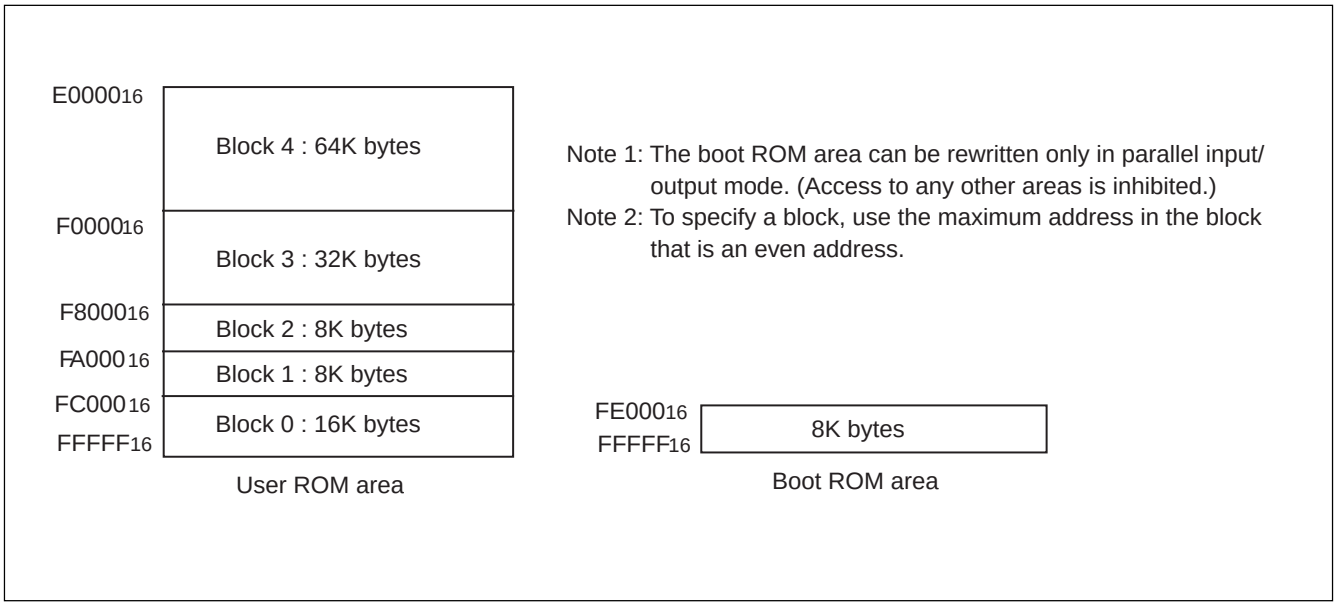



**Figure 1.149. Flash memory version user ROM memory map**



Downloaded from Arrow.com.

**Table 1.67. M30245 Flash Memory Overview**

| Item | | Performance |
|---|---|---|
| Power supply voltage | | 3.0V to 3.6V |
| Program/erase voltage | | 3.0V to 3.6V |
| Flash memory operation mode | | 3 modes:<br>• CPUrewrite<br>• Parallel I/O<br>• Standard serial I/O |
| Erase block division | User ROM area | See Figure 1.141 |
| | Boot ROM area | One division (8 Kbytes) Note |
| Program method | | In page units (256 bytes) |
| Erase method | | Collective erase/block erase |
| Program/erase control method | | Program/erase control by software command |
| Protect method | | Protection for each block by lock bit |
| Number of commands | | 8 |
| Program/erase count | | 100 times |
| Data holding | | 10 years |
| ROM code protect | | Parallel I/O and Standard serial I/O modes are supported |

Note: The boot ROM contains a stored standard serial I/O control program when it is shipped from the factory. This area can be erased and programmed in parallel I/O mode only.



Downloaded from Arrow.com.

## CPU Rewrite Mode

In CPU rewrite mode, the on-chip flash memory can be read, programmed, or erased under control of the Central Processing Unit (CPU). Only the user ROM area, shown in Figure 1.149, can be rewritten. The boot of the user ROM area.ROM area cannot be rewritten. Make sure the program and block erase commands are issued only for each block The control program for CPU rewrite mode can be stored in either the user ROM or the boot ROM area. Because the flash memory cannot be read from the CPU, the rewrite control program must be transferred to an area other than the internal flash memory before it can be executed.

### Overview

In the CPU rewrite mode, the CPU erases, programs and reads the internal flash memory as instructed by software commands. Operations are executed from a memory other than the internal flash memory, such as the internal RAM. When the CPU rewrite mode select bit (bit 1 at address $02F7_{16}$) is set to "1", transition to CPU rewrite mode occurs and software commands can be accepted. Read and write software commands and data to even-numbered addresses ("0" for address $A_0$) in 16-bit units. For 8-bit mode, always write 8-bit software commands to even-numbered addresses. Commands are ignored with odd-numbered addresses. Use software commands to control program and erase operations. The status register can verify if a program or erase operation has terminated normally or in error.
Figure 1.150 shows the flash memory control register 0. Figure 1.151 shows a flowchart for enabling/disabling the CPU rewrite mode. Always follow the operation as indicated in these flowcharts.

Bit 0 is the RY/BY status flag used exclusively to read the operating status of the flash memory. During programming and erase operations, it is "0". Otherwise, it is "1".

Bit 1 is the CPU rewrite mode select bit. The CPU rewrite mode is entered by setting this bit to "1" to make software commands accepted. In CPU rewrite mode, the CPU becomes unable to access the internal flash memory directly so, write bit 1 in an area other than the internal flash memory. To set this bit to "1", it is necessary to write "0" and then write "1" in succession when the NMI pin is "H" level. The bit can be set to "0" by only writing "0".

Bit 2 is the lock bit disable bit. By setting this bit to "1", it is possible to disable erase and write protect (block lock) effected by the lock bit data. The lock bit disable select bit only disables the lock bit function; it does not change the lock data bit value. However, if an erase operation is performed when this bit = "1", the lock bit data that is "0" (locked) is set to "1" (unlocked) after being erased. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. This bit can be controlled only when the CPU rewrite mode select bit = "1".

Bit 3 is the flash memory reset bit used to reset the control circuit of the internal flash memory. This bit is used when exiting CPU rewrite mode and when flash memory access has failed. When the CPU rewrite mode select bit is "1", writing "1" to this bit resets the control circuit. To release the reset, set this bit to "0".

Bit 5 is the user ROM area select bit that is effective only in boot mode. If this bit is set to "1", the accessed area is switched from the boot ROM area to the user ROM area. When the CPU rewrite mode is used in boot mode, set this bit to "1". If the microcomputer is booted from the user ROM area, the user ROM area is always accessed and this bit has no effect. When in boot mode, the function of this bit is effective regardless of whether the CPU rewrite mode is on or off. Use a control program that is not running in the internal flash memory to rewrite this bit.



Downloaded from Arrow.com.

Flash memory control register 0

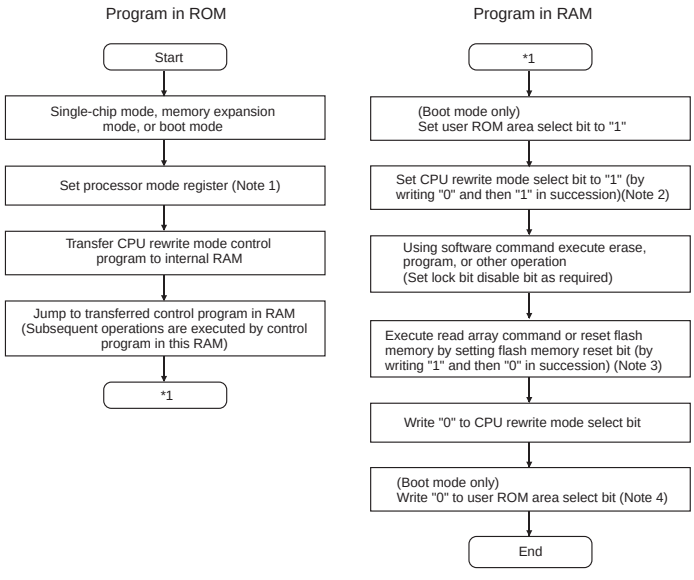
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | | 0 | | | | |

Symbol: FMR0 | Address: $02F7_{16}$ | When reset: $XX000001_{16}$

| Bit Symbol | Bit Name | Function | R | W |
|---|---|---|---|---|
| FMR00 | RY/$\overline{BY}$ status bit | 0 : Busy (overwrite or erase)<br>1 : Ready | O | X |
| FMR01 | CPU rewrite mode select bit (Note 1) | 0 : Normal mode (invalid software commands)<br>1 : CPU rewrite mode (software command accepted) | O | O |
| FMR02 | Lock bit disable bit (Note 2) | 0 : Enabled<br>1 : Disabled | O | O |
| FMR03 | Flash memory reset bit (Note 3) | 0 : Normal operation<br>1 : Reset | O | O |
| Reserved | | Must always be "0" | O | O |
| FMR05 | User ROM area select bit (Note 4). | 0 : Boot ROM area accessed<br>1 : User ROM area accessed | O | O |
| Nothing is assigned. Write "0" when writing to these bits. The value is indeterminate if read. | | | — | — |

Note 1: To set this bit to "1", the user must write "0" and then "1" to it in succession. This bit is not set to "1" unless this sequence has been performed. This is necessary to ensure that no interrupt or DMA transfer are executed during the interval. Use the control program except in the internal flash memory for writing to this bit. Also, write to this bit when the NMI pin is "H" level.
Note 2: To set this bit to "1", the user must write "0" and then "1" to it in succession when the CPU rewrite mode select bit = "1". This bit is not set to "1" unless this sequence has been performed. This is necessary to ensure that no interrupt or DMA transfer are executed during the interval.
Note 3: Effective only when CPU rewrite mode select bit "1". Set this bit to "1" and then "0" in succession.
Note 4: Effective only in boot mode. Use a control program that is not in the internal flash memory when writing to this bit.

**Figure 1.150. Flash memory control register**




Note 1: During CPU rewrite mode, set the main clock frequency to 6.25MHz or less using the main clock division register (addresses $0006_{16}$ and $0007_{16}$).
Note 2: For CPU rewrite mode select bit to be set to "1", the user needs to write a "0" and then a "1" to it in succession. When it is not this procedure, it is not enacted in "1". This is necessary to ensure that no interrupt or DMA transfer will be executed during the interval. Use the program except in the internal flash memory for write to this bit. Also write to this bit when NMI pin is "H" level.
Note 3: Before exiting the CPU rewrite mode after completing erase or program operation, always be sure to execute a read array command or reset the flash memory
Note 4: "1" can be set. However, when this bit is "1", user ROM area is accessed.

**Figure 1.151. CPU rewrite mode set/reset flowchart**



Downloaded from Arrow.com.

### Microcomputer Mode and Boot Mode

The control program for CPU rewrite mode must be written into the user ROM or boot ROM area in parallel I/O mode. If the control program is written into the boot ROM area, the standard serial I/O mode becomes unusable.

Normal microcomputer mode is entered when the microcomputer is reset when pulling CNVSS pin low. In this case, the CPU starts operating using the control program in the user ROM area.

When the microcomputer is reset by pulling the P55 pin low, and the CNVSS pin and P50 pin high, the CPU starts operating using the control program in the boot ROM area. This mode is called the "boot" mode. The control program in the boot ROM area can also be used to rewrite the user ROM area.

### Block Address

Block addresses refer to the maximum even address of each block. These addresses are used in the block erase command, lock bit program command, and read lock status command.

### Software Commands

Table 1.68 lists the software commands available with the M30245 (flash memory version). After setting the CPU rewrite mode select bit to 1, write a software command to specify an erase or program operation. When entering a software command, the upper byte ($D_8$ to $D_{15}$) is ignored.

**Table 1.68. List of software commands**

| Command | | First bus cycle | | | Second bus cycle | | | Third bus cycle | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Mode | Address | Data ($D_0$ to $D_7$) | Mode | Address | Data ($D_0$ to $D_7$) | Mode | Address | Data ($D_0$ to $D_7$) |
| 1 | Read array | Write | X (Note 6) | $FF_{16}$ | | | | | | |
| 2 | Read status register | Write | X | $70_{16}$ | Read | X | SRD (Note 2) | | | |
| 3 | Clear status register | Write | X | $50_{16}$ | | | | | | |
| 4 | Page program (Note 3) | Write | X | $41_{16}$ | Write | WA0 (Note 3) | WD0 (Note 3) | Write | WA1 | WD1 |
| 5 | Block erase | Write | X | $20_{16}$ | Write | BA (Note 4) | $D0_{16}$ | | | |
| 6 | Erase all unlocked blocks | Write | X | $A7_{16}$ | Write | X | $D0_{16}$ | | | |
| 7 | Lock bit program | Write | X | $77_{16}$ | Write | BA | $D0_{16}$ | | | |
| 8 | Read lock bit status | Write | X | $71_{16}$ | Read | BA | $D_6$ (Note 5) | | | |

Note 1: When a software command is input, the data high-order byte ($D_8$ to $D_{15}$) is ignored.
Note 2: SRD= Status register data
Note 3: WA = Write address, WD = Write data.
WA and WD must be set sequentially from $00_{16}$ to $FE_{16}$ (even byte address). The page size is 256 bytes.
Note 4: BA = Block address. Enter the maximum address of each block that is an even address.
Note 5: $D_6$ corresponds to the block lock status. When $D_6$ = "1", the unlocked blocks are "0".
Note 6: X denotes a given even address in the user ROM area.



Downloaded from Arrow.com.

**1. Read Array Command ($FF_{16}$)**

The read array mode is entered by writing the command code "$FF_{16}$" in the first bus cycle. When an even address that is to be read is input in one of the bus cycles that follow, the content of the specified address is read out at the data bus ($D_0$-$D_{15}$), 16 bits at a time.
The read array mode is retained until another command is written.

**2. Read Status Register Command ($70_{16}$)**

When the command code "$70_{16}$" is written in the first bus cycle, the content of the status register is read out at the data bus ($D_0$-$D_7$) by a read in the second bus cycle.

**3. Clear Status Register Command ($50_{16}$)**

This command clears the bits SR3 to SR5 of the status register after being set. These bits indicate that operation has ended in an error. To use this command, write the command code "$50_{16}$" in the first bus cycle.

**4. Page Program Command ($41_{16}$)**

Page program allows for high-speed programming in units of 256 bytes. Page program operation starts when the command code "$41_{16}$" is written in the first bus cycle. In the second bus cycle through the 129th bus cycle, the write data is sequentially written 16 bits at a time. At this time, the addresses $A_0$-$A_7$ need to be incremented by 2 from "$00_{16}$" to "$FE_{16}$." When the system finishes loading the data, it starts an auto write operation (data program and verify operation). Figure 1.152 shows an example of a page program flowchart.
The completed auto write operation can be confirmed by reading the status register or the flash memory control register 0. At the same time the auto write operation starts, the read status register mode is automatically entered, so the content of the status register can be read out. The status register bit 7 (SR7) is set to 0 at the same time the auto write operation starts and is returned to 1 when the auto write operation has been completed. In this case, the read status register mode remains active until the Read Array command ($FF_{16}$) or Read Lock Bit Status command ($71_{16}$) is written or the flash memory is reset using its reset bit.
The RY/BY status flag of the flash memory control register 0 is "0" during auto write operation and "1" when the auto write operation and status register bit 7 have been completed.
After the auto write operation is completed, the status register can read out the results of the auto write operation. Refer to the status register section for more details.
Each block of the flash memory can be write protected by using a lock bit. Refer to the data protect function section for more details. Additional writes to the pages previously programmed are prohibited.

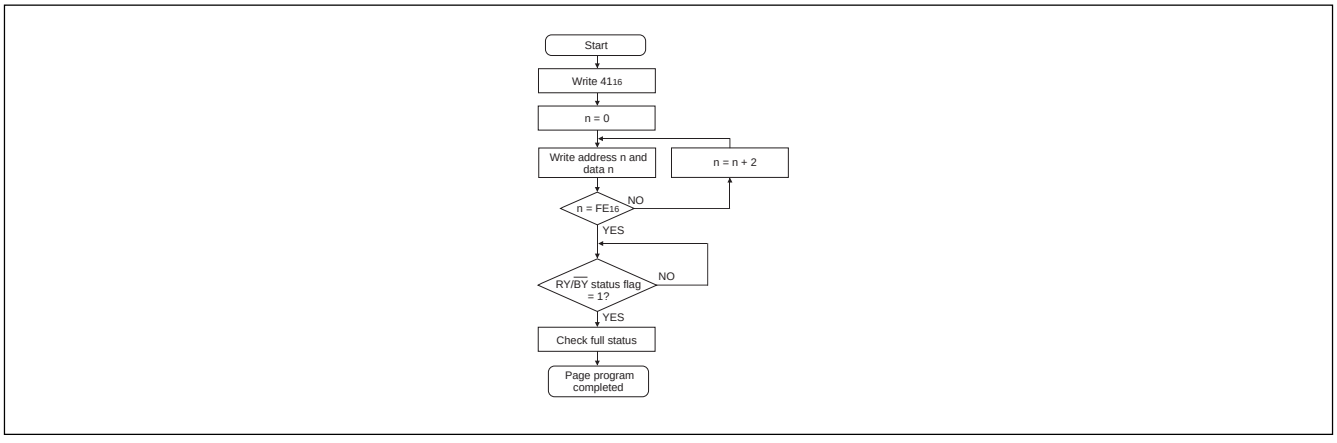



**Figure 1.152. Page program flowchart**



Downloaded from Arrow.com.

### 5. Block Erase Command ($20_{16}$/$D0_{16}$)

By writing the command code "$20_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" in the second bus cycle to the block address of a flash memory block, the system initiates an auto erase (erase and erase verify) operation. Figure 1.153 is an example of a block erase flowchart.

Read the status register or the flash memory control register 0 to confirm the completion of the auto erase operation. At the same time the auto erase operation starts, the read status register mode is automatically entered, so the contents of the status register can be read out. The status register bit 7 (SR7) is set to "0" at the same time the auto erase operation starts and is returned to "1" when the auto erase operation is completed. The read status register mode remains active until the Read Array command ($FF_{16}$) or Read Lock Bit Status command ($71_{16}$) is written or the flash memory is reset using its reset bit.

The RY/BY status flag of the flash memory control register 0 is "0" during auto erase operation and "1" when the auto erase operation and status register bit 7 is completed.

After the auto erase operation is completed, the status register can read for the results of the auto erase operation. Refer to the status register for more details.

A lock bit protects each block of the flash memory against erasure. Refer to the data protect function section for more details.

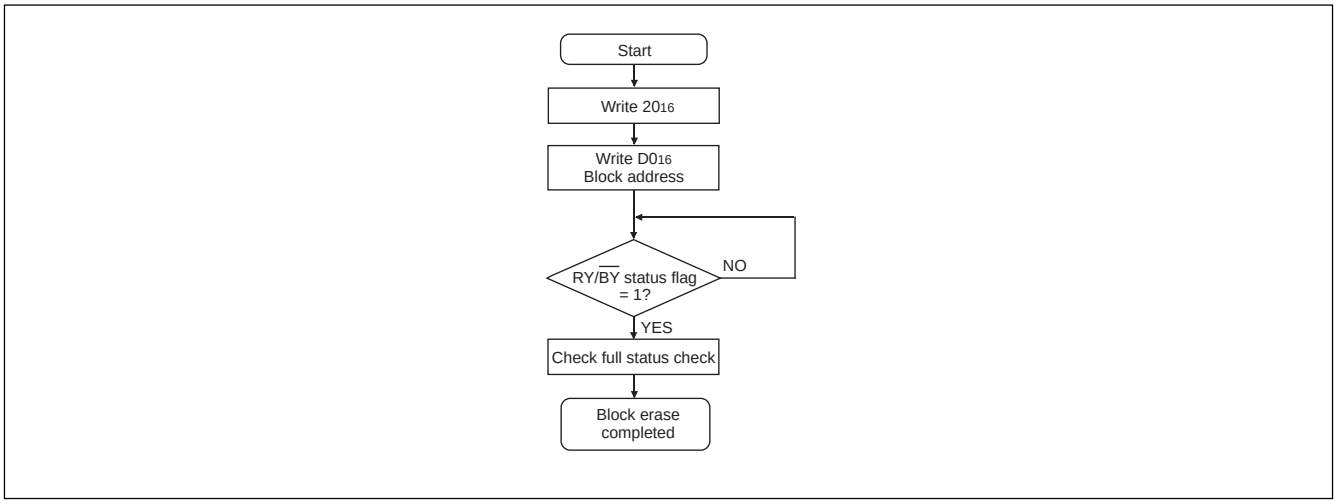



**Figure 1.153. Block erase flowchart**

### 6. Erase All Unlock Blocks Command ($A7_{16}$/$D0_{16}$)

By writing the command code "$A7_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" in the second bus cycle that follows, the system starts erasing blocks successively.

Reading the status register or the flash memory control register 0 confirms whether the erase all unlock blocks command was terminated in the same way as for block erase. Also, the status register can read out the results of the auto erase operation.

When the lock bit disable bit of the flash memory control register 0 = "1", all blocks are erased regardless of how the lock bit is set. On the other hand, when the lock bit disable bit = "0", the function of the lock bit is effective and only unlocked blocks (where lock bit data = "1") are erased.



Downloaded from Arrow.com.

### 7. Lock Bit Program Command ($77_{16}$/$D0_{16}$)

By writing the command code "$77_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" in the second bus cycle to the block address of a flash memory block, the system sets the lock bit for the specified block to "0" (locked).
Figure 1.154 is an example of a lock bit program flowchart. The lock bit status (lock bit data) can be read out by a read lock bit status command.
Reading the status register or the flash memory control register 0 confirms whether the lock bit program command has terminated the same way as in the page program.
Refer to the data protect function section for more details.

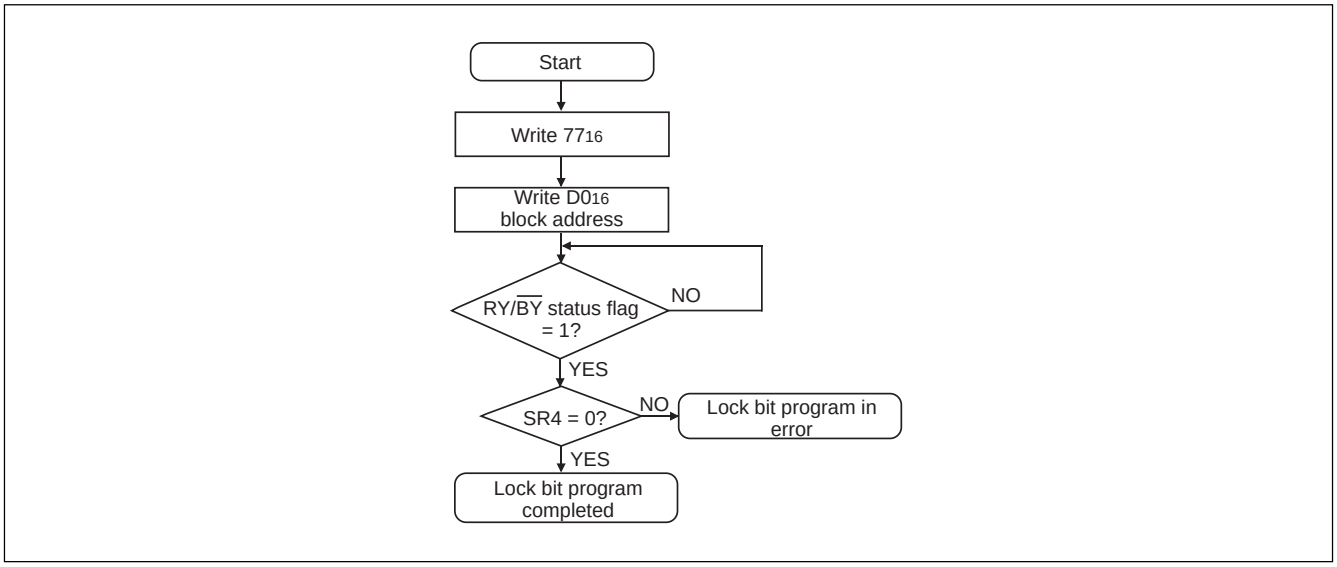



**Figure 1.154. Lock bit program flowchart**

### 8. Read Lock Bit Status Command ($71_{16}$)

By writing the command code "$71_{16}$" in the first bus cycle and then the block address of a flash memory block in the second bus cycle that follows, the system reads out the status of the lock bit of the specified block to the data bit ($D_6$).
Figure 1.155 is an example of a read lock bit program flowchart.

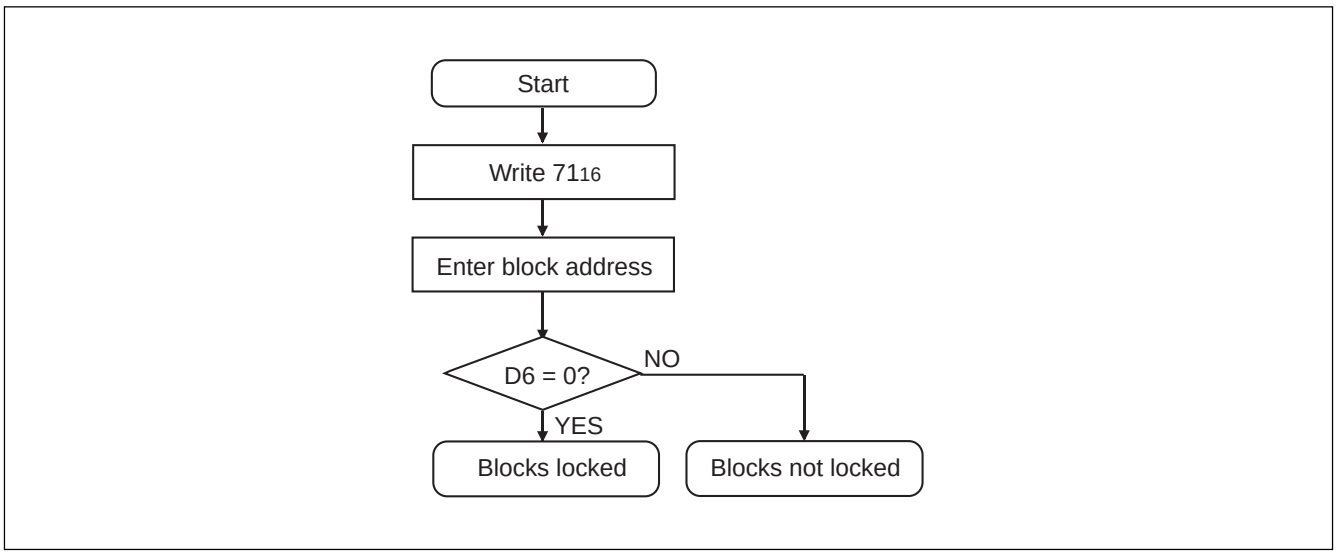



**Figure 1.155. Read lock bit status flowchart**



Downloaded from Arrow.com.

#### Data Protect Function (Block Lock)

Each block in Figure 1.149 has a nonvolatile lock bit to specify that the block is protected (locked) against erase/write. The lock bit program command is used to set the lock bit to 0 (locked). The lock bit of each block can be read out using the read lock bit status command.

Whether block lock is enabled or disabled is determined by the status of the lock bit and the lock bit disable bit in flash memory control register 0.

(1) When the lock bit disable bit = "0", a specified block can be locked or unlocked by the lock bit status (lock bit data). If lock bit data = "0" (locked), they are disabled against erase/write. On the other hand, if lock bit data = "1" (unlocked) they are enabled for erase/write.

(2) When the lock bit disable bit = "1", all blocks are unlocked regardless of the lock bit data, and enabled for erase/write. In this case, the lock bit data is set to "1" (unlocked) after erasure, so that the lock bit is disabled.

### Status Register

The status register indicates the flash memory operating status and whether an erase or program operation has terminated normally or in error. Table 1.69 details the status register. The contents of this register can be read out only by writing the read status register command ($70_{16}$). Writing the Clear Status Register command ($50_{16}$) clears the status register. After a reset, the status register is set to "$80_{16}$."

**Table 1.69. Status register bit definition**

| Each SRD bit | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR7 (Bit 7) | Write state machine (WSM) | Ready | Busy |
| SR6 (Bit 6) | Reserved | _ | _ |
| SR5 (Bit 5) | Erase status | Terminated in error | Terminated normally |
| SR4 (Bit 4) | Program status | Terminated in error | Terminated normally |
| SR4 (Bit 3) | Block status after program | Terminated in error | Terminated normally |
| SR2 (Bit 2) | Reserved | _ | _ |
| SR1 (Bit 1) | Reserved | _ | _ |
| SR0 (Bit 0) | Reserved | _ | _ |

#### Write state machine (WSM) status (SR7)

After power-on, the write state machine (WSM) status is set to "1".

The write state machine (WSM) status indicates the operating status of the RY/BY pin output. This status bit is set to "0" during an auto write or auto erase operation and is set to "1" when the operation is completed.



Downloaded from Arrow.com.

**Erase status (SR5)**

The erase status indicates the operating status of an auto erase to the CPU. It is set to "1" when an erase error occurs. The erase status is reset to "0" when cleared.

**Program status (SR4)**

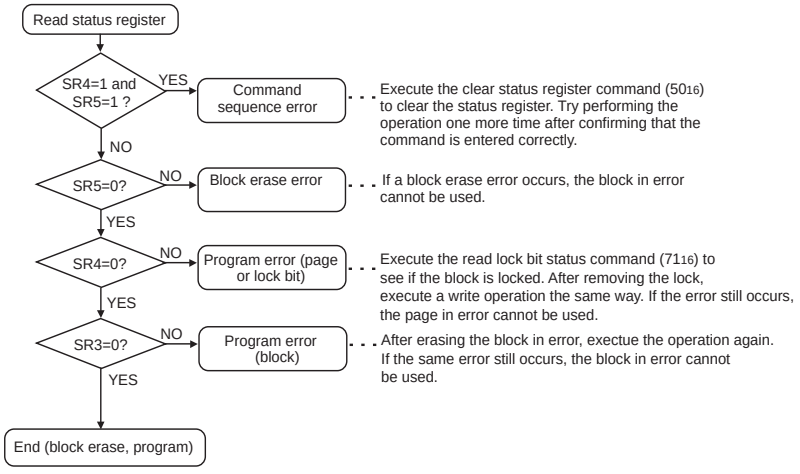
The program status indicates the operating status of an auto write to the CPU. It is set to "1" when a write error occurs. The program status is reset to "0" when cleared.
When an erase command is in error, which occurs if the command entered after the block erase command ($20_{16}$) is not the confirmation command ($D0_{16}$), both the program status and erase status (SR5) are set to "1".
If the program status or erase status = "1", the following commands entered by command write are not accepted and SR4 and SR5 are set to "1" (command sequence error):
(1) A valid command is not entered correctly
(2) The data entered in the second bus cycle of lock bit program ($77_{16}$/$D0_{16}$), block erase ($20_{16}$/$D0_{16}$), or erase all unlocked blocks ($A7_{16}$/$D0_{16}$) is not the $D0_{16}$ or $FF_{16}$. However, if $FF_{16}$ is entered, read array is assumed and the command that has been set up in the first bus cycle is canceled.

**Block status after program (SR3)**

If data is overwritten (this occurs when a memory cell becomes overcharged and data incorrectly read), "1" is set for the program status after the program at the end of the page write operation. In other words:
• When writing ends successfully, "$80_{16}$" is output;
• When writing fails, "$90_{16}$" is output;
• When excessive data is written, "$88_{16}$" is output.

## Full-Status Check

A full-status check allows the user to review the erase and program operations. Figure 1.156 shows a full-status check flowchart and the action to take when an error occurs.




Note: When one of SR5 to SR3 is set to 1, none of the page program, block erase, erase all unlocked blocks and lock bit program commands are accepted. Execute the clear status register command ($50_{16}$) before executing these commands.

**Figure 1.156. Full-status check flowchart**



Downloaded from Arrow.com.

## Precautions

### Operation speed

During CPU rewrite mode, set the main clock frequency to 6.25MHz or less using the main clock division select bits (bit 6 at address $0006_{16}$, and bits 6 and 7 at address $0007_{16}$).

### Prohibited Instructions

The UND, INTO, JMPS, JSRS, and BRK instructions cannot be used during CPU rewrite mode because they refer to the internal data of the flash memory.

### Prohibited Interrupts

The address match interrupt cannot be used during CPU rewrite mode because it refers to the internal data of the flash memory. If the interrupt's vector is in the variable vector table, it can be used by transferring the vector into the RAM area. The $\overline{\text{NMI}}$ and watchdog timer interrupts can be used to change the CPU rewrite mode select bit forcibly to normal mode (FMR01="0") when the interrupt occurs. If the rewrite operation is stopped when the $\overline{\text{NMI}}$ or watchdog timer interrupts occurs, the CPU rewrite mode select bit should be set to "1" and the erase/program operation should be repeated.

### Reset

Reset input is always accepted.

### Access

To set the CPU rewrite mode select bit, and the lock bit disable bit to "1", the user must write a "0" and then a "1". This sequence must be followed to set this bit to "1". This is necessary to ensure that no interrupt or DMA transfer will be executed during the interval.
Write to the CPU rewrite mode select bit when $\overline{\text{NMI}}$ pin is a "H" level.

### Access disable

Write the CPU rewrite mode select bit, and the user ROM area select bit in an area other than the internal flash memory.

### Writing in the user ROM area

If power is lost while rewriting blocks that contain the flash rewrite program with the CPU rewrite mode, the blocks may not be correctly rewritten. Afterwards, it is possible that the flash memory can not be rewritten. Therefore, use the standard serial I/O mode or parallel I/O mode to rewrite these blocks.

### Using the lock bit

In CPU rewrite mode, use a program that can set and clear the lock bit disable bit (FMR02).



Downloaded from Arrow.com.

## Parallel I/O Mode

The parallel I/O mode can be used to input and output the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. In this mode, the M30245 (flash memory version) operates in a manner similar to other flash memory from Renesas. Because there are some differences with some functions not available with the microcomputer and the memory capacity, the M30245 cannot be programmed by a programmer for other Renesas flash memory. Use an exclusive programmer that supports the M30245 (flash memory version). Refer to the instruction manual of each programmer manufacturer for usage details.

### User ROM and Boot ROM Areas

In parallel I/O mode, the user ROM and boot ROM areas shown in Figure 1.149 can be rewritten. Both areas of flash memory can be operated on in the same way.

Program and block erase operations can be performed in the user ROM area. The user ROM area and it's blocks are shown in Figure 1.149.

The boot ROM area is 8 Kbytes in size. In parallel I/O mode, it is located at addresses $0FE000_{16}$ through $0FFFFF_{16}$. Ensure that the program and block erase operations are always performed within this address range. Access to any location outside this address range is prohibited.

In the boot ROM area, an erase block operation is applied to only one 8 Kbyte block. The boot ROM area has a standard serial I/O mode control program installed at the Renesas factory, therefore, it is unnecessary to write to the boot ROM area when using standard serial I/O mode.



Downloaded from Arrow.com.

### ROM code protect function

To prevent the contents of the flash memory from being read out or rewritten too easily, the device incorporates a ROM code protect function for use in parallel I/O mode. The ROM code protect function prevents reading out or modifying the contents of the flash memory by using the ROM code protect control register ($0FFFFF_{16}$) during parallel I/O mode. Figure 1.157 shows the ROM code protect control address. (This address exists in the user ROM area.)

If one pair of ROM code protect bits is set to "0", ROM code protect is turned on so that the contents of the flash memory are protected against being read out or modified. The ROM code protect function is implemented in two levels. If level 2 is selected, the flash memory is protected even against readout by a shipment inspection LSI tester, etc. If both level 1 and level 2 are selected, level 2 is selected by default.

If both of the two ROM code protect reset bits are set to "00," the ROM code protect function is turned off so that the contents of the flash memory can be read out or modified. Once ROM code protect is turned on, the contents of the ROM code protect reset bits cannot be modified in parallel I/O mode. Use the serial I/O mode or another mode to rewrite the contents of the ROM code protect reset bits.

ROM code protect control register




Symbol: ROMCP
Address: $0FFFFF_{16}$
When reset: $FF_{16}$

| Bit Symbol | Bit Name | Function |
|---|---|---|
| Reserved | | Always set to "1" |
| ROMCP2 | ROM code protect level 2 set bit (Note 1, 2) | b3 b2<br>0 0 : Protect enable<br>0 1 : Protect enable<br>1 0 : Protect enable<br>1 1 : Protect enable |
| ROMCR | ROM code protect reset bit (Note 3) | b5 b4<br>0 0 : No protect set bit<br>0 1 : Protect set bit active<br>1 0 : Protect set bit active<br>1 1 : Protect set bit active |
| ROMCP1 | ROM code protect level 1 set bit (Note 1) | b7 b6<br>0 0 : Protect enable<br>0 1 : Protect enable<br>1 0 : Protect enable<br>1 1 : Protect enable |

Note 1: When ROM code protect is turned on, the on-chip flash memory is protected against readout or modification in parallel input/output mode.
Note 2: When ROM code protect level 2 is turned on, ROM code readout by a shipment inspection LSI tester, etc, is inhibited.
Note 3: The ROM code protect reset bits can be used to turn off ROM code protect levels 1 and 2. However, because these bits cannot be changed in parallel input/output mode, they need to be rewritten in serial input/output or some other mode.

**Figure 1.157. ROM code protect control register**



Downloaded from Arrow.com.

## Standard Serial I/O Mode

The standard serial I/O mode serially inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. It uses a specific serial programmer to accomplish this. It is different from the parallel I/O mode because the CPU controls operations like rewriting the flash memory (using the CPU rewrite mode) and serially inputting data.

The standard serial I/O mode is entered by clearing the reset with the $P5_0$ ($\overline{CE}$) pin set to a "H" level, the $P5_5$ ($\overline{EPM}$) pin set to a "L" level and the CNVss pin set to a "H" level. (For normal microprocessor mode, set the CNVss pin to "L" level.) A control program is written in the boot ROM area when the product is shipped from Renesas. The standard serial I/O mode cannot be used if the boot ROM area is rewritten in the parallel I/O mode. Figure 1.158 shows the pin connections for the standard serial I/O mode. Table 1.70 lists the pin functions for standard serial IO mode.

There are two standard serial I/O modes that both require a purpose-specific serial programmer: clock synchronous and clock asynchronous. Standard serial I/O switches between mode 1 (clock synchronous) and mode 2 (clock asynchronous) according to the level of the CLK1 pin when the reset is released. Serial data I/O uses UART1 and transfers the data serially in 8-bit units.

To use standard serial I/O mode 1 (clock synchronous):

- Set the CLK1 pin to "H" level and release the reset
- This mode uses the four UART1 pins CLK1, RxD1, TxD1 and $\overline{RTS1}$ (BUSY).
- The CLK1 pin is the transfer clock input pin through which an external transfer clock is input.
- The TxD1 pin is for CMOS output.
- The $\overline{RTS1}$ (BUSY) pin outputs an "L" level when ready for reception and an "H" level when reception starts.

To use standard serial I/O mode 2 (clock asynchronous):

- Set the CLK1 pin to "L" level and release the reset.
- This mode uses the two UART1 pins RxD1 and TxD1.

In standard serial I/O mode, only the user ROM area indicated in Figure 1.149 can be rewritten. The boot ROM cannot.

The standard serial I/O mode uses a 7-byte ID code. When there is data in the flash memory, commands sent from the programmer are not accepted unless the ID codes are identical.

### ID Code Check Function

The ID code check function can be used in serial I/O mode to protect the contents of the flash memory from being read out or rewritten. If the contents of the flash memory are not blank, the ID code sent from the serial programmer is compared with the ID code written in the flash. If the ID codes are not identical, the commands sent from the serial programmer are not accepted. Figure 1.159 shows the ID code store addresses. The ID code consists of 8-bit data: (beginning with the first byte) $0FFFDF_{16}$, $0FFFE3_{16}$, $0FFFEB_{16}$, $0FFFEF_{16}$, $0FFFF3_{16}$, $0FFFF7_{16}$, and $0FFFFB_{16}$. Write a program that has the ID code preset at these addresses.



Downloaded from Arrow.com.

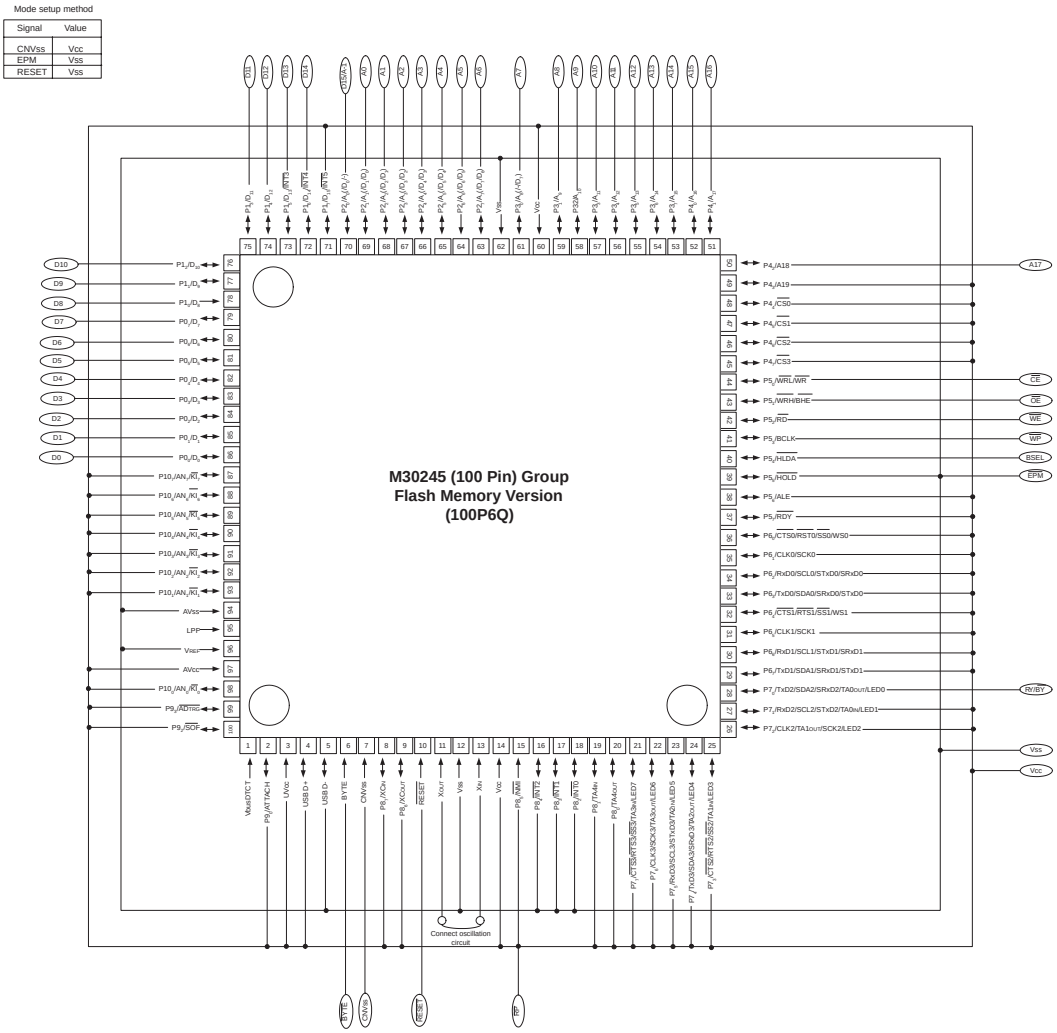



**Figure 1.158. Pin descriptions for standard serial I/O mode**

| Address | | |
|---|---|---|
| 0FFFDF16 to 0FFFDC16 | ID1 | Undefined instruction vector |
| 0FFFE316 to 0FFFE016 | ID2 | Overflow vector |
| 0FFFE716 to 0FFFE416 | | BRK instruction vector |
| 0FFFEB16 to 0FFFE816 | ID3 | Address match vector |
| 0FFFEF16 to 0FFFEC16 | ID4 | |
| 0FFFF316 to 0FFFF016 | ID5 | Watchdog timer vector |
| 0FFFF716 to 0FFFF416 | ID6 | |
| 0FFFFB16 to 0FFFF816 | ID7 | NMI vector |
| 0FFFFF16 to 0FFFFC16 | | Reset vector |

4 bytes

**Figure 1.159. ID code storage addresses**



Downloaded from Arrow.com.

**Table 1.70. Flash memory standard serial I/O mode pin functions**

| Pin | Name | IO | Description |
|---|---|---|---|
| Vcc, Vss | Power input | | Apply program/erase voltage to Vcc pin and 0V to Vss pin |
| CNVss | CNVss | I | Connect to Vcc pin. |
| $\overline{RESET}$ | Reset input | I | Reset input pin. While reset is "L" level, a 20 cycle or longer clock must be input to $X_{IN}$ pin. |
| $X_{IN}$ | Clock input | I | Connect a ceramic resonator or crystal oscillator between $X_{IN}$ and $X_{OUT}$ pins. To input an externally generated clock, input it to $X_{IN}$ pin and open $X_{OUT}$ pin. |
| $X_{OUT}$ | Clock output | O | |
| BYTE | BYTE | I | Connect this pin to Vcc or Vss. |
| AVcc, AVss | Analog power supply input | I | Connect AVss to Vss and AVcc to Vcc respectively. |
| $V_{REF}$ | Reference voltage input | I | Enter the reference voltage for A/D converter from this pin. |
| $P0_0$ to $P0_7$ | Input Port P0 | I | Input "H" or "L" level signal or open. |
| $P1_0$ to $P1_7$ | Input Port P1 | I | Input "H" or "L" level signal or open. |
| $P2_0$ to $P2_7$ | Input Port P2 | I | Input "H" or "L" level signal or open. |
| $P3_0$ to $P3_7$ | Input Port P3 | I | Input "H" or "L" level signal or open. |
| $P4_0$ to $P4_7$ | Input Port P4 | I | Input "H" or "L" level signal or open. |
| $P5_1$ to $P5_4$, $P5_6$, $P5_7$ | Input Port P5 | I | Input "H" or "L" level signal or open. |
| $P5_0$ | $\overline{CE}$ input | I | Input "H" level signal. |
| $P5_5$ | $\overline{EPM}$ input | I | Input "L" level signal. |
| $P6_0$ to $P6_3$ | Input Port P6 | I | Input "H" or "L" level signal or open. |
| $P6_4$ | BUSY output | O | Standard serial mode 1: BUSY signal output pin<br>Standard serial mode 2: Monitors the program operation check. |
| $P6_5$ | SCLK input | I | Standard serial mode 1: Serial clock input pin<br>Standard serial mode 2: Input "L" level signal |
| $P6_6$ | RxD input | I | Serial data input pin |
| $P6_7$ | TxD output | O | Serial data output pin |
| $P7_0$ to $P7_7$ | Input Port P7 | I | Input "H" or "L" level signal or open. |
| $P8_0$ to $P8_4$, $P8_6$, $P8_7$ | Input Port P8 | I | Input "H" or "L" level signal or open. |
| $P8_5$ | $\overline{NMI}$ input | I | Connect this pin to Vcc |
| $P9_0$, $P9_2$, $P9_3$ | Input Port P9 | I | Input "H" or "L" level signal or open. |
| $P10_0$ to $P10_7$ | Input Port P10 | I | Input "H" or "L" level signal or open. |



Downloaded from Arrow.com.

### Standard serial I/O mode 1

In standard serial I/O mode 1, software commands, addresses, and data are input and output between the MCU and a serial programmer using 4-wire clock-synchronized serial I/O (UART1). Standard serial I/O mode 1 is initiated by releasing the reset with the P6$_5$ (CLK1) pin at a "H" level.

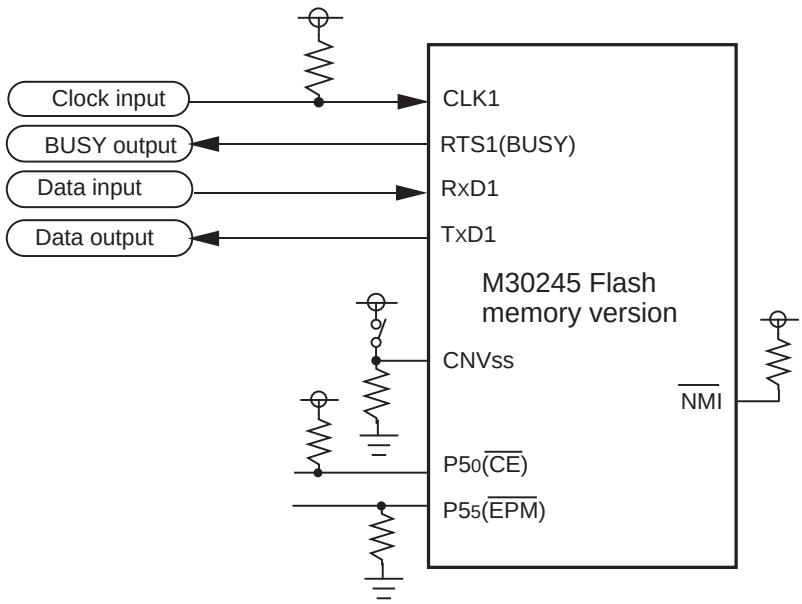
In reception, the software commands, addresses, and program data are synchronized with the rise of the transfer clock (input to the CLK1 pin) and input to the MCU on the RxD1 pin.

In transmission, the read data and status are synchronized with the fall of the transfer clock, and output from the TxD1 pin. The TxD1 pin is a CMOS output. Transfer is in 8-bit units with LSB first.

When busy, such as during transmission, reception, erasing, or program execution, the $\overline{\text{RTS1}}$ (BUSY) pin is at a "H" level. Accordingly, always start the next transfer after the $\overline{\text{RST1}}$ (BUSY) pin is at a "L" level.

#### Example Circuit Application

Figure 1.160 shows a circuit application for the standard serial I/O mode 1. Control pins will vary according to peripheral unit (programmer), therefore check the peripheral unit (programmer) manual for more information.




(1) Control pins and external circuitry will vary according to peripheral unit (programmer). For more information, see the peripheral unit (programmer) manual.
(2) In this example, the microprocessor mode and standard serial I/O mode are switched via a switch.

**Figure 1.160. Example circuit application for the standard serial I/O mode 1**



Downloaded from Arrow.com.

### Software Commands

In the standard serial I/O mode 1, erase , program and read operations are controlled by transferring software commands using the RxD1 pin.

Data and status registers in memory can be read after inputting software commands. Reading the status register can check the status of the flash memory operating state or successful completion of a program or erase operation. Table 1.71 lists the software commands.

**Table 1.71. Software commands**

| | Control command | | 2nd byte | 3rd byte | 4th byte | 5th byte | 6th byte | | When ID is not verified |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Page read | $FF_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 2 | Page program | $41_{16}$ | Address (middle) | Address (high) | Data input | Data input | Data input | Data input to 259th byte | Not acceptable |
| 3 | Block erase | $20_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 4 | Erase all unlocked blocks | $A7_{16}$ | $D0_{16}$ | | | | | | Not acceptable |
| 5 | Read status register | $70_{16}$ | SRD output | SRD1 output | | | | | Acceptable |
| 6 | Clear status register | $50_{16}$ | | | | | | | Not acceptable |
| 7 | Read lock bit status | $71_{16}$ | Address (middle) | Address (high) | Lock bit data output | | | | Not acceptable |
| 8 | Lock bit program | $77_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 9 | Lock bit enable | $7A_{16}$ | | | | | | | Not acceptable |
| 10 | Lock bit disable | $75_{16}$ | | | | | | | Not acceptable |
| 11 | ID check function | $F5_{16}$ | Address (low) | Address (middle) | Address (high) | ID size | ID1 | To ID7 | Acceptable |
| 12 | Download function | $FA_{16}$ | Size (low) | Size (high) | Check sum | Data input | As required | | Not acceptable |
| 13 | Version data output function | $FB_{16}$ | Version data output | Version data output | Version data output | Version data output | Version data output | Version data output to 9th byte | Acceptable |
| 14 | Boot ROM area output function | $FC_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 15 | Read check data | $FD_{16}$ | Check data (low) | Check data (high) | | | | | Not acceptable |

Note 1: The shaded areas indicate a transfer from flash memory MCU to peripheral unit. All other data is transferred from the peripheral unit to the flash memory MCU.
Note 2: SRD refers to Status Register Data. SRD1 refers to Status Register Data 1.
Note 3: All commands are accepted if the flash memory is blank.



Downloaded from Arrow.com.

### 1. Page Read Command

The page read command reads the specified page (256 bytes) in the flash memory sequentially one byte at a time. Figure 1.161 shows the timing for the page read.

To execute the page read command:

(1) Transfer the "FF16" command code with the 1st byte.

(2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.

(3) From the 4th byte on, data (D0-D7) for the page (256 bytes) specified by addresses A8 to A23, will be output sequentially from the smallest address first in sync with the fall of the clock.

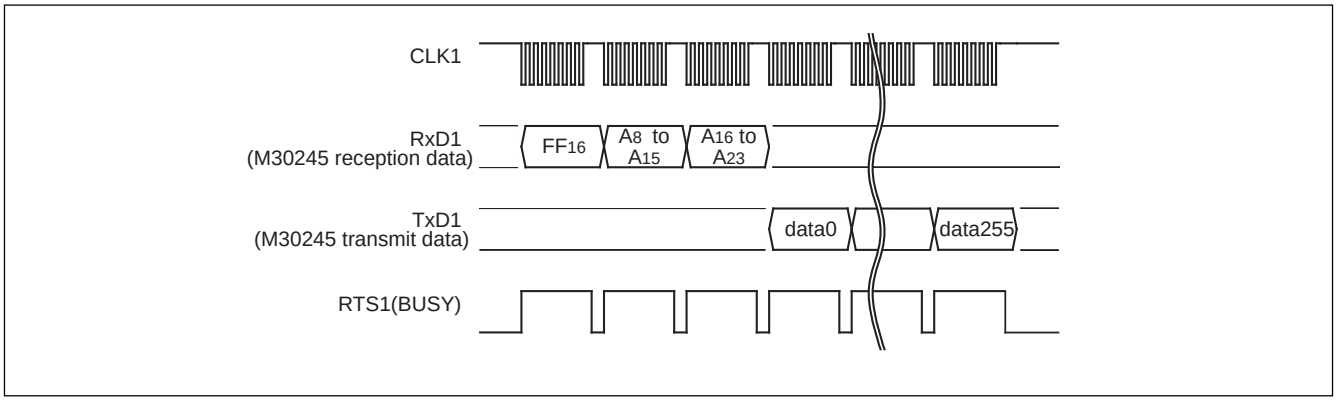



**Figure 1.161. Timing for page read**

### 2. Page Program Command

This command writes the specified page (256 bytes) in the flash memory sequentially one byte at a time. Figure 1.162 shows the page program timing.

To execute the page program command:

(1) Transfer the "4116" command code with the 1st byte.

(2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.

(3) From the 4th byte on, write data (D0-D7) for the page (256 bytes) specified by addresses A8 to A23, will be input sequentially from the smallest address first. The page is automatically written.

When reception of the page (256 bytes) ends, the $\overline{\text{RTS1}}$ (BUSY) signal changes from the "H" to the "L" level. The status register shows the results of the page program. Refer to the status register section for more details.

Each block is write-protected with the lock bit. Refer to the data protection function section for more details. Additional writing of previously programmed pages is not allowed.

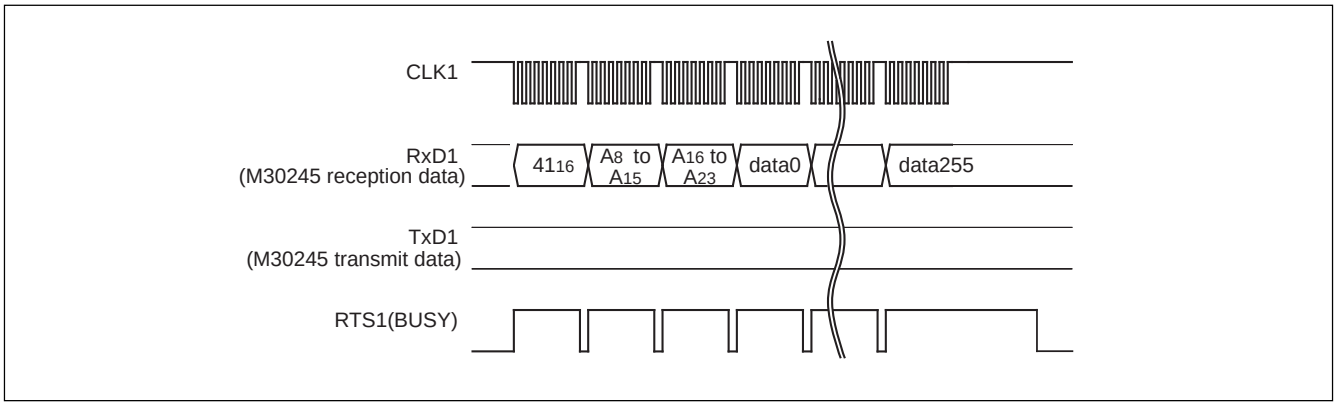



**Figure 1.162. Timing for the page program**



Downloaded from Arrow.com.

### 3. Block Erase Command

This command erases the data in the specified block. Figure 1.163 shows the block erase timing.
To execute the block erase command:
(1) Transfer the "$20_{16}$" command code with the 1st byte.
(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively. Write the highest address of the specified block for addresses $A_{16}$ to $A_{23}$.
(3) Transfer the verify command code "$D0_{16}$" with the 4th byte. The verify command code allows the erase operation to start for the specified block in the flash memory.
When the block erase is finished, the $\overline{RTS1}$ (BUSY) signal changes from the "H" to the "L" level. The status register shows the results of the block erase command. Refer to the status register section for more details.
Each block is erase-protected with the lock bit. Refer to the data protection section for more details.

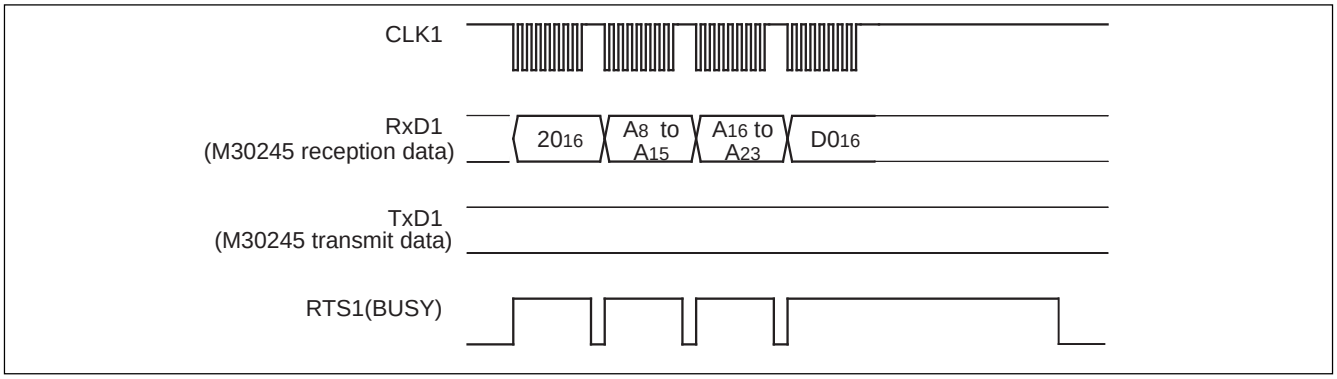



**Figure 1.163. Timing for block erasing**

### 4. Erase All Unlocked Blocks Command

This command erases the contents of all blocks. Figure 1.164 shows the timing for erasing all unlocked blocks. To execute the erase all unlocked blocks command:
(1) Transfer the "$A7_{16}$" command code with the 1st byte.
(2) Transfer the verify command code "$D0_{16}$" with the 2nd byte. The verify command code allows the erase operation to continue for all of the flash memory.
When block erasing ends, the $\overline{RTS1}$ (BUSY) signal changes from the "H" to the "L" level The status register shows the results of the erase all unlocked blocks command. Refer to the status register section for more details.
Each block is erase-protected with the lock bit. Refer to the data protection section for more details.

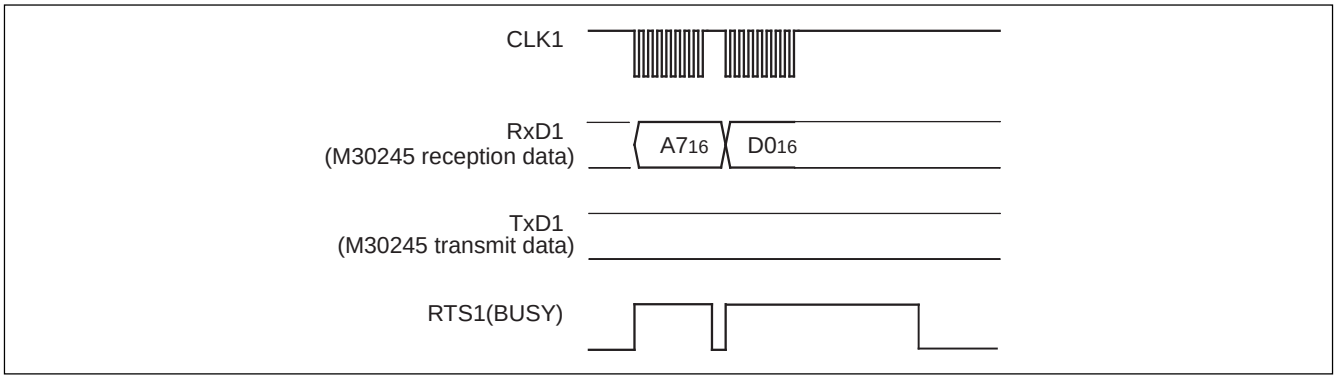



**Figure 1.164. Timing for erasing all unlocked blocks**



Downloaded from Arrow.com.

### 5. Read Status Register Command

This command reads the status information. Figure 1.165 shows the read status command timing.
When the "70$_{16}$" command code is sent with the 1st byte, the contents of the status register (SRD) are read with the 2nd byte and the contents of status register 1 (SRD1) are read with the 3rd byte.

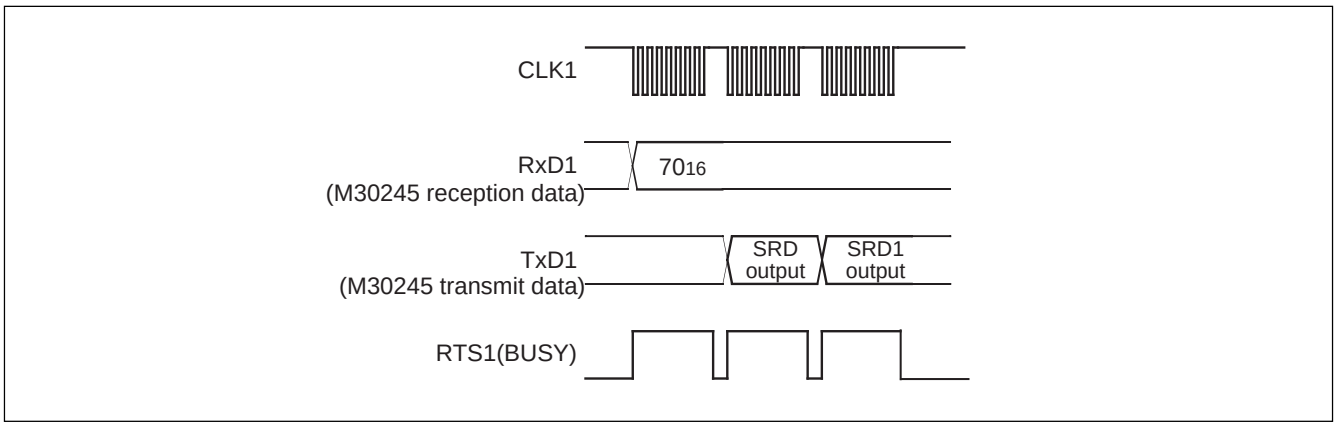



**Figure 1.165. Timing for reading the status register**

### 6. Clear Status Register Command

This command clears the bits (SR3-SR5) that are set when an erase, program, or status operation ends in error. Figure 1.166 shows the clear status register timing.
When the "50$_{16}$" command code is sent with the 1st byte, bits SR3-SR5 are cleared. When the clear status register operation has ended, the $\overline{\text{RTS1}}$ (BUSY) signal changes from the "H" to the "L" level.

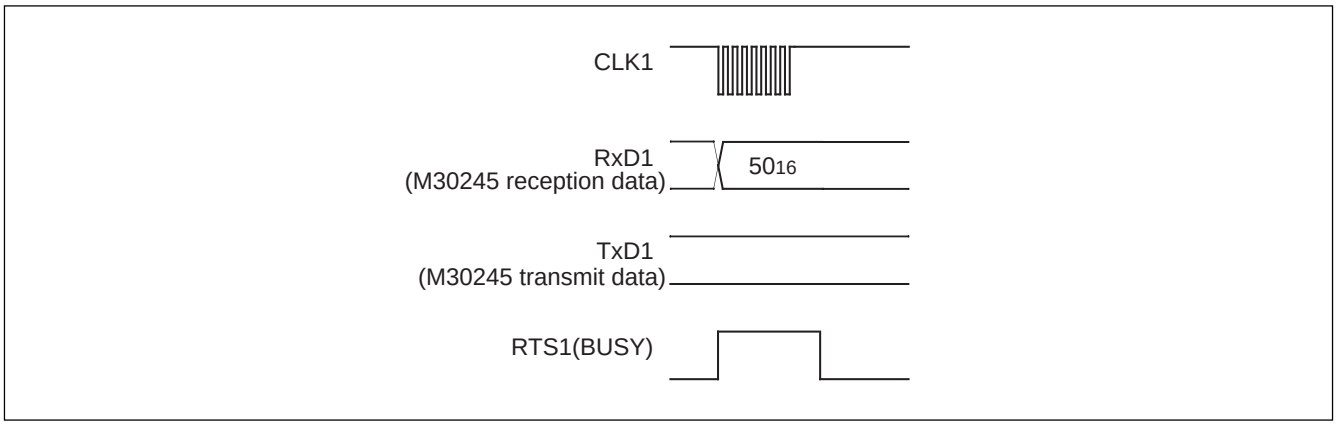



**Figure 1.166. Timing for clearing the status register**

### 7. Read Lock Bit Status Command

This command reads the lock bit status of the specified block. Figure 1.167 shows the lock bit status timing. To execute the read lock bit status command:
(1) Transfer the "71$_{16}$" command code with the 1st byte.
(2) Transfer addresses A$_8$ to A$_{15}$ and A$_{16}$ to A$_{23}$ with the 2nd and 3rd bytes respectively.
(3) The lock bit data of the specified block is output with the 4th byte. The 6th bit (D$_6$) of the output data is the lock bit data.
Write the highest address of the specified block for addresses A$_8$ to A$_{23}$.



Downloaded from Arrow.com.

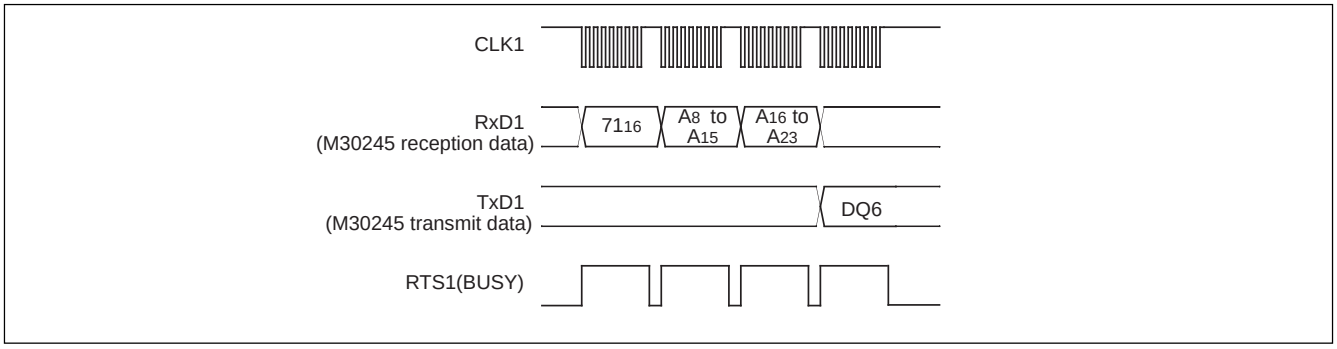



**Figure 1.167. Timing for reading lock bit status**

### 8. Lock Bit Program Command

This command writes "0" (lock) for the lock bit of the specified block. Figure 1.168 shows the lock bit program timing. To execute the lock bit program command:

(1) Transfer the "$77_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code "$D0_{16}$" with the 4th byte. With the verify command code, "0" is written to the lock bit of the specified block. Write the highest address of the specified block for addresses $A_8$ to $A_{23}$.

When writing ends, the $\overline{RTS1}$ (BUSY) signal changes from the "H" to the "L" level. Lock bit status can be read with the read lock bit status command. Refer to the data protection function for more details.

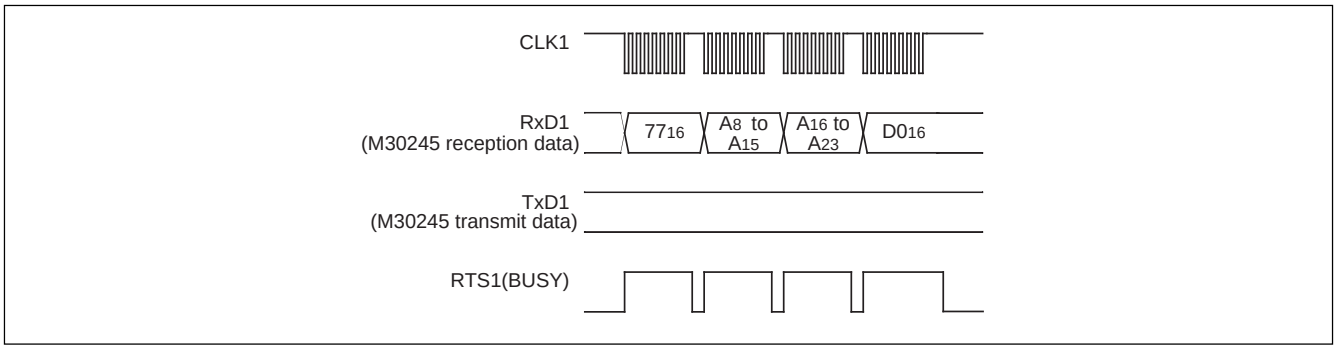



**Figure 1.168. Timing for the lock bit program**

### 9. Lock Bit Enable Command

This command enables the lock bit for all blocks. Figure 1.169 shows the lock bit enable timing. The command code "$7A_{16}$" is sent with the 1st byte of the serial transmission. This command only enables the lock bit function; it does not set the lock bit itself.

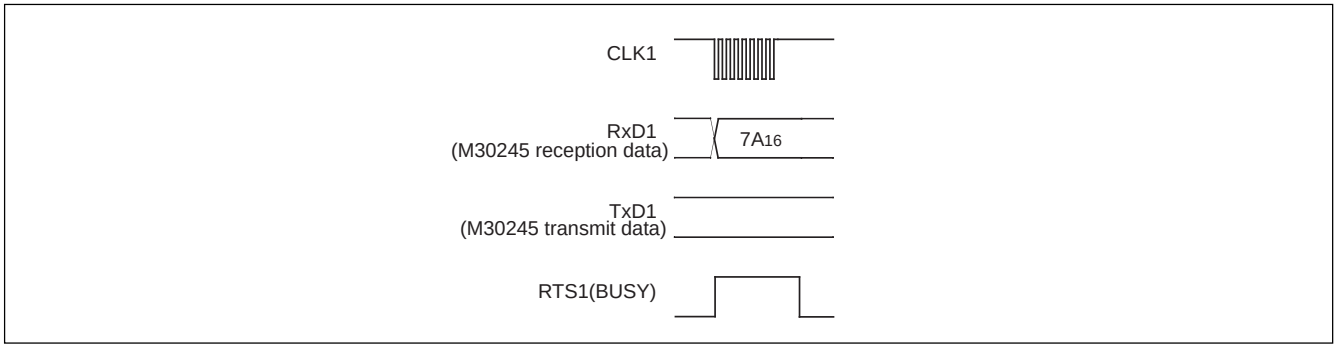



**Figure 1.169. Timing for enabling the lock bit**



Downloaded from Arrow.com.

### 10. Lock Bit Disable Command

This command disables the lock bit for all blocks. Figure 1.170 shows the lock bit disable command timing. The command code "75$_{16}$" is sent with the 1st byte of the serial transmission. This command only disables the lock bit function; it does not set the lock bit itself. However, if an erase command is executed after executing the lock bit disable command, all "0" (locked) lock bit data is set to "1" (unlocked) after the erase operation ends. After reset the lock bit is always enabled.

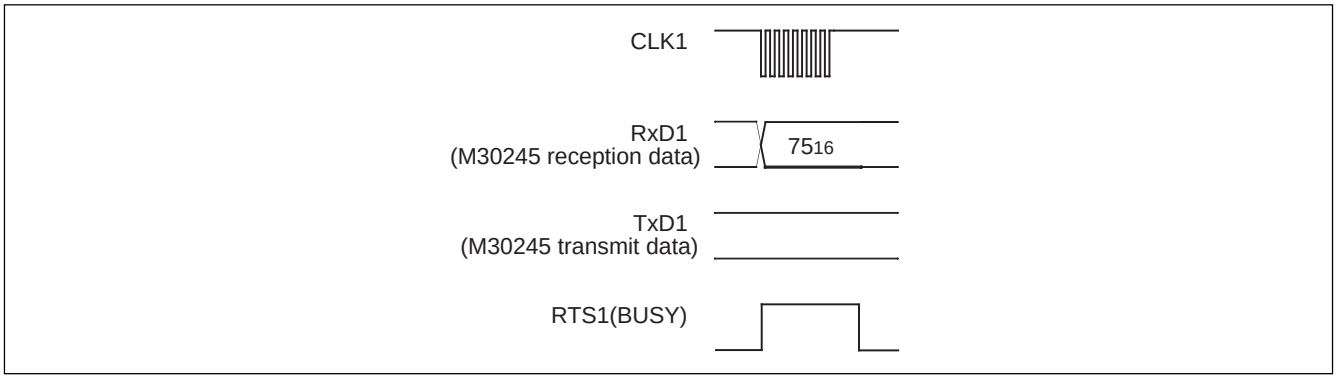



**Figure 1.170. Timing for disabling the lock bit**

### 11. ID Check Command

This command checks the ID code. Figure 1.171 shows the ID check command timing. To execute the boot ID check command:

(1) Transfer the "F5$_{16}$" command code with the 1st byte.

(2) Transfer addresses A$_0$ to A$_7$, A$_8$ to A$_{15}$ and A$_{16}$ to A$_{23}$ of the 1st byte of the ID code with the 2nd, 3rd and 4th bytes respectively.

(3) Transfer the number of data sets of the ID code with the 5th byte.

(4) The ID code is sent with the 6th byte on, starting with the 1st byte of the code.

See the ID code section for more information.

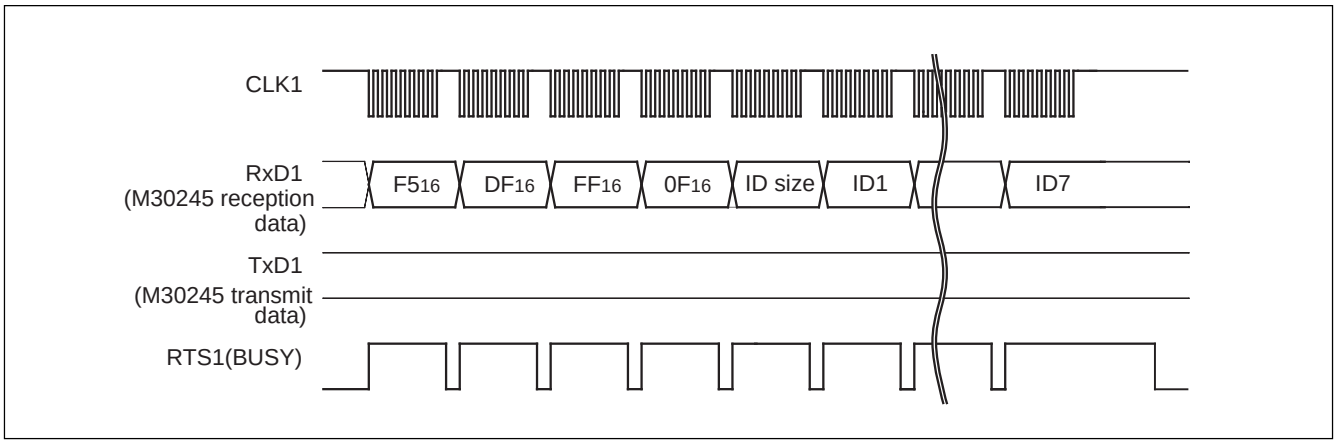



**Figure 1.171. Timing for the ID check**



Downloaded from Arrow.com.

### 12. Download Command

This command downloads a program to the RAM for execution. Figure 1.172 shows the download command timing. To execute the download command:

(1) Transfer the "FA16" command code with the 1st byte.

(2) Transfer the program size with the 2nd and 3rd bytes.

(3) Transfer the check sum with the 4th byte. The check sum is added to all data sent starting with the 5th byte.

(4) The program to execute is sent starting with the 5th byte.

After all data has been transmitted, if the check sum matches, the downloaded program is executed. The size of the program will vary according to the internal RAM.

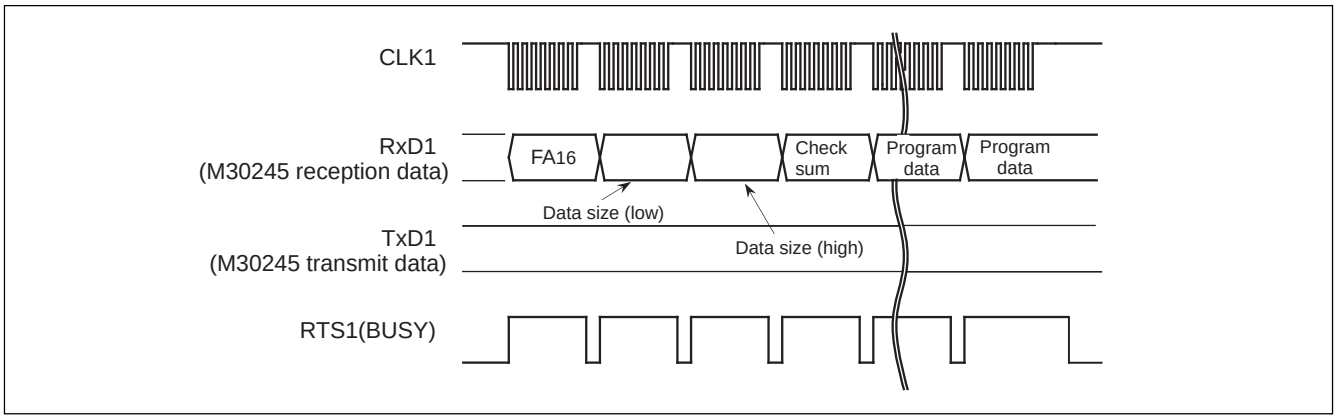



**Figure 1.172. Timing for download**

### 13. Version Information Output Command

This command outputs the version information of the control program stored in the boot area. Figure 1.173 shows the version information output timing. To execute the version information output command:

(1) Transfer the "FB16" command code with the 1st byte.

(2) The version information will be output from the 2nd byte on. The version data is composed of 8 ASCII code characters.

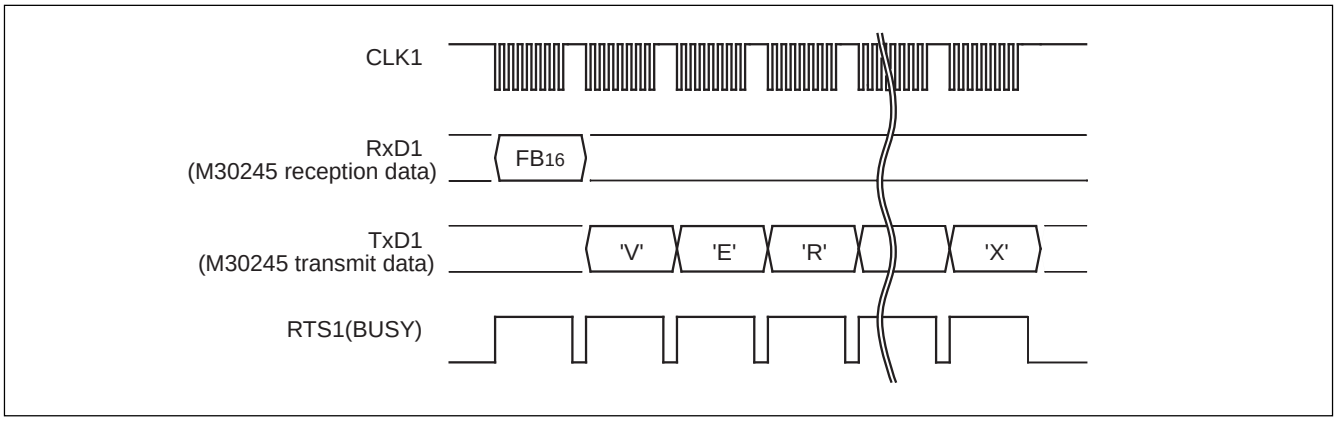



**Figure 1.173. Timing for version information output**



Downloaded from Arrow.com.

### 14. Boot ROM Area Output Command

This command outputs the control program stored in the boot ROM area in one page blocks (256 bytes). Figure 1.174 shows the boot ROM area output timing. To execute the boot ROM area output command:
(1) Transfer the "FC16" command code with the 1st byte.
(2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.
(3) Starting with the 4th byte, data (D0-D7) for the page (256 bytes) specified by addresses A8 to A23 will be output sequentially from the smallest address first, in sync with the falling edge of the transfer clock.

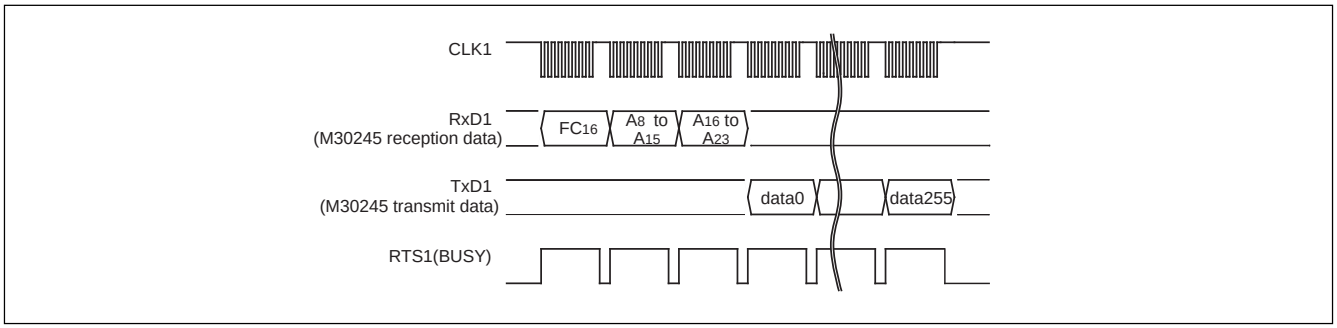



**Figure 1.174. Timing for boot ROM area output**

### 15. Read Check Data command

This command reads the check data that confirms that the write data, sent with the page program command, has been successfully received. Figure 1.175 shows the read check data timing. To execute the read check data command:
(1) Transfer the "FD16" command code with the 1st byte.
(2) The check data (low) is received with the 2nd byte and the check data (high) with the 3rd byte.
To use this read check data command, first execute the command and then initialize the check data. Then execute the page program command the required number of times. Afterwards, when the read check command is executed again, the check data (for all of the read data that was sent with the page program command during this time) is read. The check data is the result of a CRC operation of write data.

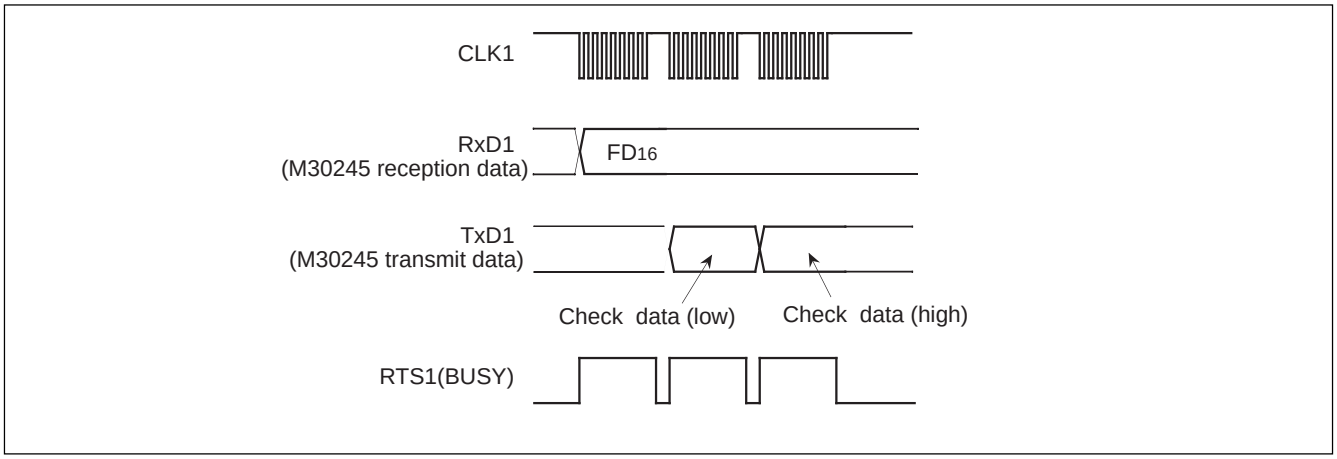



**Figure 1.175. Timing for the read check data**



Downloaded from Arrow.com.

### Data Protection (Block Lock)

Each block in Figure 1.176 has a nonvolatile lock bit that indicates protection (block lock) against erasing/writing. A block is locked (writing "0" for the lock bit) with the lock bit program command. Any lock bit can be read with the read lock bit status command.
Block lock disable/enable is determined by the status of the lock bit and execution status of the lock bit disable and lock bit enable commands.
(1) After reset and the lock bit enable command is executed, the specified block can be locked/unlocked using the lock bit (lock bit data). Blocks with a "0" lock bit data are locked and cannot be erased or written to. Blocks with a "1" lock bit data are unlocked and can be erased or written to.
(2) After the lock bit disable command has been executed, all blocks are unlocked regardless of the lock bit data status and can be erased or written to. In this case, any lock bit data that was "0" before the block was erased is set to "1" (unlocked) after erasing.

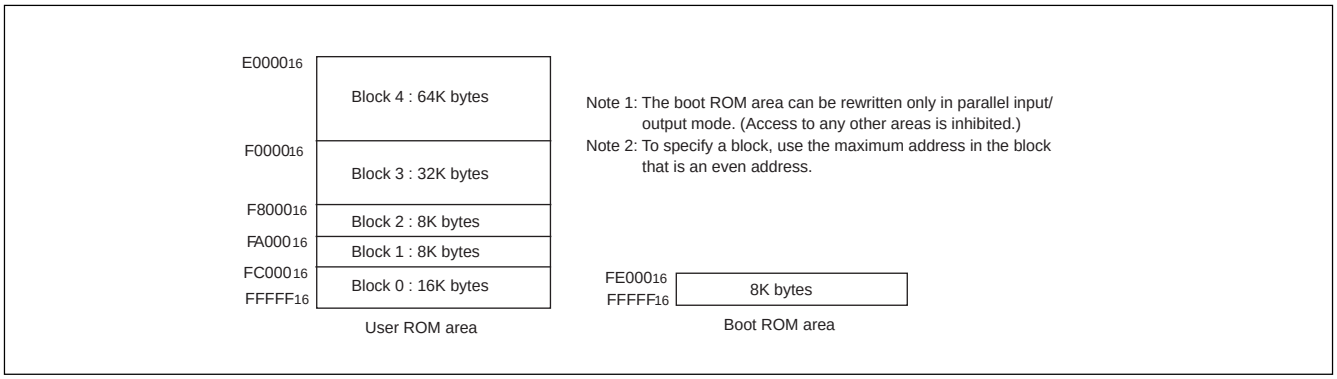



**Figure 1.176. Block diagram of the flash memory version**

### Status Register (SRD)

The status register indicates the flash memory operating status and whether an erase or program operation has terminated normally or in error. It can be read by using the read status register command ($70_{16}$). Writing the clear status register command ($50_{16}$) clears the status register. Table 1.72 defines each status register bit. After reset, the status register outputs "$80_{16}$".

**Table 1.72. Status register (SRD)**

| Each SRD bit | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR7 (Bit 7) | Write state machine (WSM) | Ready | Busy |
| SR6 (Bit 6) | Reserved | _ | _ |
| SR5 (Bit 5) | Erase status | Terminated in error | Terminated normally |
| SR4 (Bit 4) | Program status | Terminated in error | Terminated normally |
| SR4 (Bit 3) | Block status after program | Terminated in error | Terminated normally |
| SR2 (Bit 2) | Reserved | _ | _ |
| SR1 (Bit 1) | Reserved | _ | _ |
| SR0 (Bit 0) | Reserved | _ | _ |



Downloaded from Arrow.com.

**Write State Machine (WSM) Status (SR7)**

The write state machine (WSM) status indicates the operating status of the flash memory. When power is turned on, "1" (ready) is set. The bit is set to "0" (busy) during an auto-write or auto-erase operation, but returns to "1" when the operation ends.

**Erase Status (SR5)**

The erase status reports the operating status of the auto-erase operation. If an erase error occurs, it is set to "1". When the erase status is cleared, it is set to "0".

**Program Status (SR4)**

The program status reports the operating status of the auto-write operation. If a write error occurs, it is set to "1". When the program status is cleared, it is set to "0".

**Block Status After Program (SR3)**

If data is overwritten (this occurs when a memory cell becomes overcharged and data is incorrectly read), a "1" is set for the block status after program at the end of the page write operation.
In other words:
• When writing ends successfully "$80_{16}$" is output
• When writing fails, "$90_{16}$" is output
• When excessive data is written, "$88_{16}$" is output.
If "1" is written to any SR5, SR4 or SR3 bits, the page program, block erase, erase all unlocked blocks and lock bit program commands are not accepted. Before executing these commands, execute the clear status register command ($50_{16}$).

**Status Register 1 (SRD1)**

Status register 1 indicates the status of serial communications, ID check results, and check sum comparisons. It can be read after the SRD by writing the read status register command ($70_{16}$). Status register 1 can be cleared by writing the clear status register command ($50_{16}$).
Table 1.73 defines each status register 1 bit. "$00_{16}$" is output when power is turned ON and the flag status is maintained even after the reset.

**Table 1.73. Status register 1 (SRD1)**

| SRD1 bits | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR15 (Bit 7) | Boot update complete bit | Completed | Not updated |
| SR14 (Bit 6) | Reserved | _ | _ |
| SR13 (Bit 5) | Reserved | _ | _ |
| SR12 (Bit 4) | Checksum match bit | Match | No match |
| SR11 (Bit 3)<br>SR10 (Bit 2) | ID check completed bits | 0 0 Not verified<br>0 1 Verified no match<br>1 0 Reserved<br>1 1 Verified | |
| SR9 (Bit 1) | Data receive time out | Time out | Normal operation |
| SR8 (Bit 0) | Reserved | _ | _ |



Downloaded from Arrow.com.

**Boot Update Completed Bit (SR15)**

This flag indicates if the control program was properly downloaded (using the download function) to RAM.

**Check Sum Consistency Bit (SR12)**

This flag indicates if the check sum matches after a program is downloaded for execution (using the download function).

**ID Check Completed Bits (SR11 and SR10)**

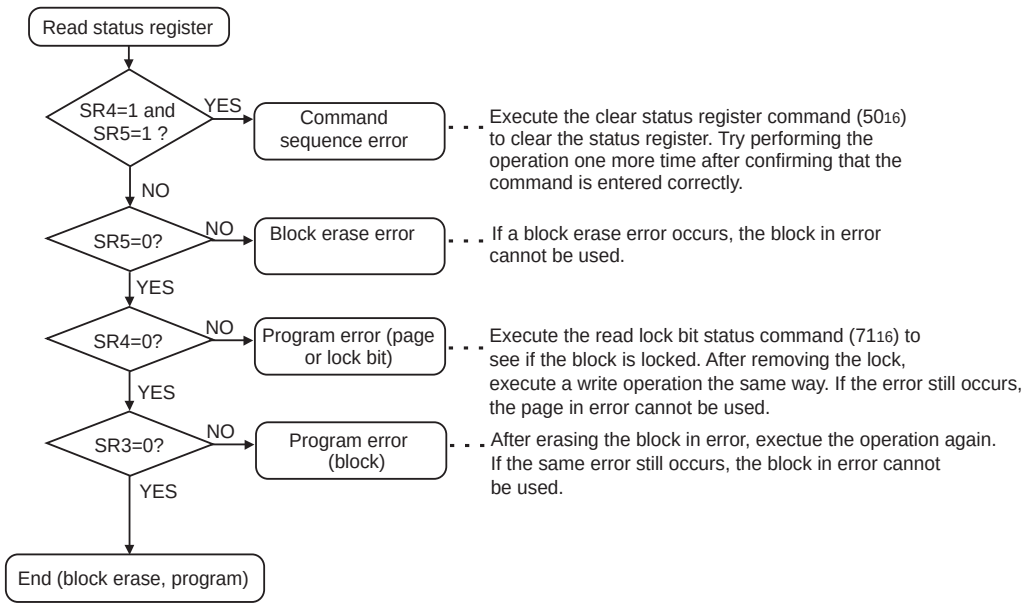
These flags indicate the result of ID checks. Some commands cannot be accepted without an ID check.

**Data Reception Time Out (SR9)**

This flag indicates when a time out error is generated during data reception. If this flag is set during data reception, the received data is discarded and the microcomputer returns to the command wait state.

**Full Status Check**

A full-status check allows the user to review the erase and program operations. Figure 1.177 shows a full-status check flowchart and the action to take when an error occurs.




Note: When one of SR5 to SR3 is set to 1, none of the page program, block erase, erase all unlocked blocks and lock bit program commands are accepted. Execute the clear status register command (5016) before executing these commands.

**Figure 1.177. Full status check flowchart**



Downloaded from Arrow.com.

### Standard serial I/O mode 2

In standard serial I/O mode 2 (clock asynchronous), software commands, addresses and data are input and output between the MCU and peripheral units (serial programmer, etc.) using 2-wire clock-asynchronous serial I/O (UART1). Standard serial I/O mode is entered by releasing the reset with the $P6_5$ (CLK1) pin at a "L" level. The TxD1 pin is set to CMOS output. Data transfer is in 8-bit units with LSB first, 1 stop bit and parity OFF.

After reset, connections can be established at 9,600 bps when initial communications are made with a peripheral unit. This requires a main clock with a minimum 2 MHz input oscillation frequency. The baud rate can also be changed from 9,600 bps to 19,200, 38,400, or 57,600 bps by executing software commands. Communication errors may occur because of the main clock oscillation frequency. If errors occur, change the main clock's oscillation frequency and the baud rate.

After executing commands from a peripheral unit that require time to erase and write data, as with the erase and program commands, allow a sufficient time interval or execute the read status command and check how the processing ended before executing the next command.

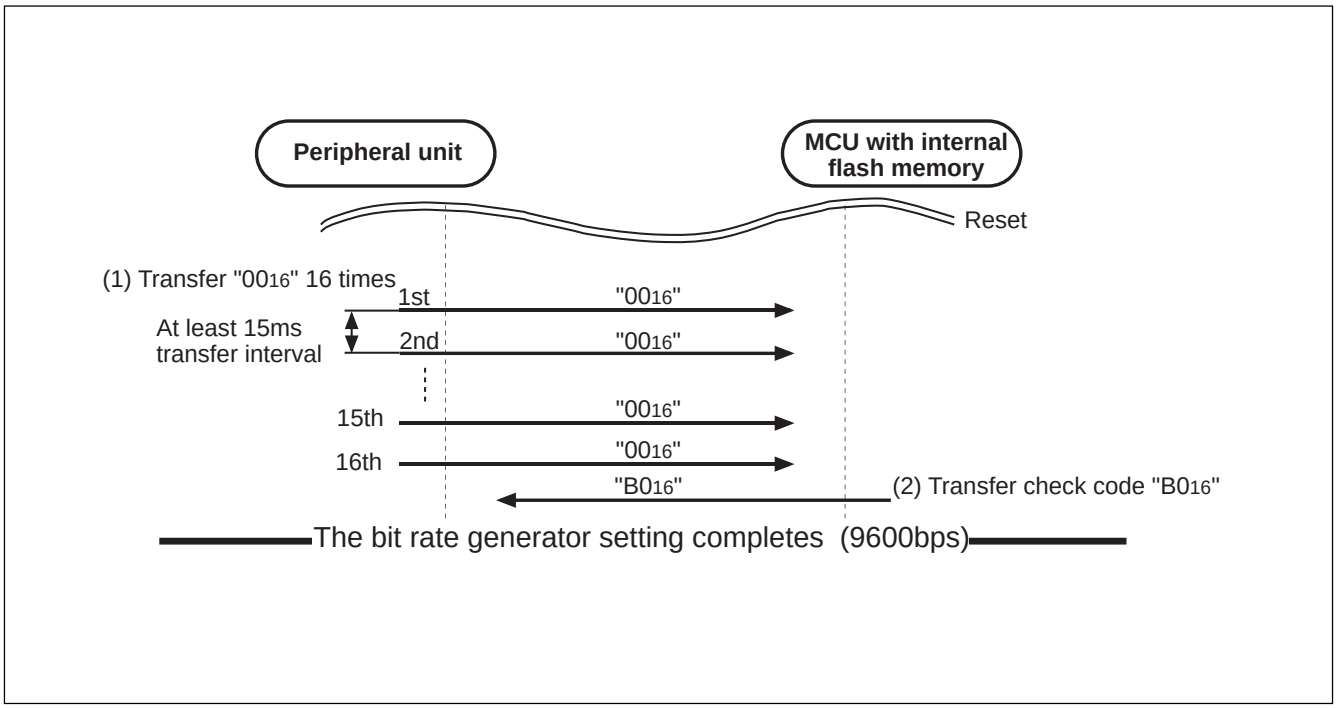
Data and status registers can be read after transmitting software commands. Reading the status register can check status of the flash memory operating state or successful completion of a program or erase operation.

Initial communications with peripheral units

After reset, the bit rate generator is adjusted to 9,600 bps to match the main clock's oscillation frequency, by sending the code as prescribed by the protocol for initial communications with peripheral units. Figure 1.178 shows the initial communication with peripheral units.

(1) Transmit "$00_{16}$" from a peripheral unit 16 times. (The MCU with internal flash memory sets the bit rate generator so that "$00_{16}$" can be successfully received.)

(2) The MCU with internal flash memory outputs the "$B0_{16}$" check code and initial communications end successfully. Initial communications must be transmitted at a speed of 9,600 bps and a transfer interval of a minimum 15 ms. Also, the baud rate at the end of initial communications is 9,600 bps.




Note. If the peripheral unit cannot receive "$B0_{16}$" successfully, change the oscillation frequency of the main clock.

**Figure 1.178. Peripheral unit and initial communication**



Downloaded from Arrow.com.

### Frequency identification

When "00$_{16}$" data is received 16 times from a peripheral unit at a baud rate of 9,600 bps, the value of the bit rate generator is set to match the operating frequency (2 - 16 MHz). The highest speed is taken from the first 8 transmissions and the lowest from the last 8. These values are then used to calculate the bit rate generator value for a baud rate of 9,600 bps. Baud rate cannot be attained with some operating frequencies. Table 1.74 lists the operation frequency and the baud rate.

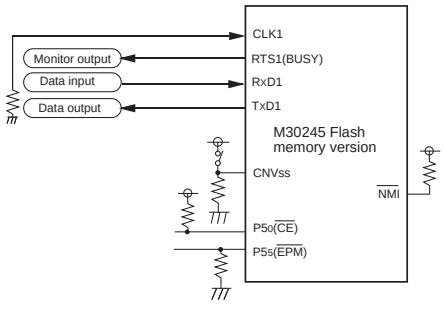
**Table 1.74. Operation frequency and baud rate**

| Operation Frequency | Baud rate 9,600 | Baud rate 19,200 | Baud rate 38,400 | Baud rate 57,600 |
|---|---|---|---|---|
| 16 MHz | + | + | + | + |
| 12 MHz | + | + | + | + |
| 11 MHz | + | + | + | + |
| 10 MHz | + | + | + | + |
| 8 MHz | + | + | + | + |
| 7.3728 MHz | + | + | + | + |
| 6 MHz | + | + | + | – |
| 5 MHz | + | + | + | – |
| 4.5 MHz | + | + | + | + |
| 4.194304 MHz | + | + | + | – |
| 4 MHz | + | + | – | – |
| 3.58 MHz | + | + | + | + |
| 3 MHz | + | + | + | – |
| 2 MHz | + | – | – | – |

+ : Communications possible
_ : Communications not possible

### Example Circuit Application

Figure 1.179 shows a circuit application for the standard serial I/O mode 2.




Note: In this example, the microprocessor mode and standard serial I/O mode are switched via a switch.

**Figure 1.179. Example circuit application for the standard serial I/O mode 2**



Downloaded from Arrow.com.

### Software Commands

In the standard serial I/O mode 2, erase, program, and read operations are controlled by transferring software commands using the RxD1 pin. Standard serial I/O mode 2 adds four transmission speed commands - 9,600, 19,200, 38,400, and 57,600 bps - to the software commands of standard serial I/O mode 1. Table 1.75 lists the software commands for serial I/O mode 2.

**Table 1.75. Software commands**

| | Control command | | 2nd byte | 3rd byte | 4th byte | 5th byte | 6th byte | | When ID is not verified |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Page read | $FF_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 2 | Page program | $41_{16}$ | Address (middle) | Address (high) | Data input | Data input | Data input | Data input to 259th byte | Not acceptable |
| 3 | Block erase | $20_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 4 | Erase all unlocked blocks | $A7_{16}$ | $D0_{16}$ | | | | | | Not acceptable |
| 5 | Read status register | $70_{16}$ | SRD output | SRD1 output | | | | | Acceptable |
| 6 | Clear status register | $50_{16}$ | | | | | | | Not acceptable |
| 7 | Read lock bit status | $71_{16}$ | Address (middle) | Address (high) | Lock bit data output | | | | Not acceptable |
| 8 | Lock bit program | $77_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 9 | Lock bit enable | $7A_{16}$ | | | | | | | Not acceptable |
| 10 | Lock bit disable | $75_{16}$ | | | | | | | Not acceptable |
| 11 | ID check function | $F5_{16}$ | Address (low) | Address (middle) | Address (high) | ID size | ID1 | To ID7 | Acceptable |
| 12 | Download function | $FA_{16}$ | Size (low) | Size (high) | Check sum | Data input | As required | | Not acceptable |
| 13 | Version data output function | $FB_{16}$ | Version data output | Version data output | Version data output | Version data output | Version data output | Version data output to 9th byte | Acceptable |
| 14 | Boot ROM area output function | $FC_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 15 | Read check data | $FD_{16}$ | Check data (low) | Check data (high) | | | | | Not acceptable |
| 16 | Baud rate 9600 | $B0_{16}$ | $B0_{16}$ | | | | | | Acceptable |
| 17 | Baud rate 19200 | $B1_{16}$ | $B1_{16}$ | | | | | | Acceptable |
| 18 | Baud rate 38400 | $B2_{16}$ | $B2_{16}$ | | | | | | Acceptable |
| 19 | Baud rate 57600 | $B3_{16}$ | $B3_{16}$ | | | | | | Acceptable |

Note 1: The shaded areas indicate a transfer from flash memory MCU to peripheral unit. All other data is transferred from the peripheral unit to the flash memory MCU.
Note 2: SRD refers to Status Register Data. SRD1 refers to Status Register Data 1.
Note 3: All commands are accepted if the flash memory is blank.



Downloaded from Arrow.com.

### 1. Page Read Command

The page read command reads the specified page (256 bytes) in the flash memory sequentially one byte at a time. Figure 1.180 shows the page read timing. To execute the page read command:
(1) Transfer the "$FF_{16}$" command code with the 1st byte.
(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.
(3) From the 4th byte on, data ($D_0$-$D_7$) for the page (256 bytes) specified by addresses $A_8$ to $A_{23}$, will be output sequentially from the smallest address first, in sync with the rise of the clock.

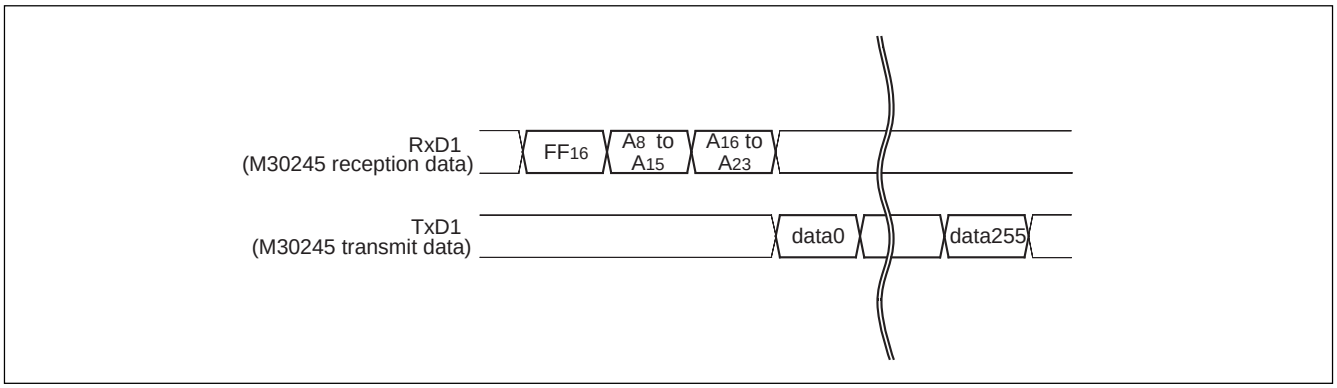



**Figure 1.180. Timing for page read**

### 2. Page Program Command

This command writes the specified page (256 bytes) in the flash memory sequentially one byte at a time. Figure 1.181 shows the page program timing. To execute the page program command:
(1) Transfer the "$41_{16}$" command code with the 1st byte.
(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.
(3) From the 4th byte on, write data (D0-D7) for the page (256 bytes) specified by addresses $A_8$ to $A_{23}$, is input sequentially from the smallest address first. The page is automatically written.
The page program results can be reviewed in the status register. Refer to the status register section for more details.
Each block can be write-protected with the lock bit. Refer to the data protection function for more details. Additional writing of previously programmed pages is not allowed.

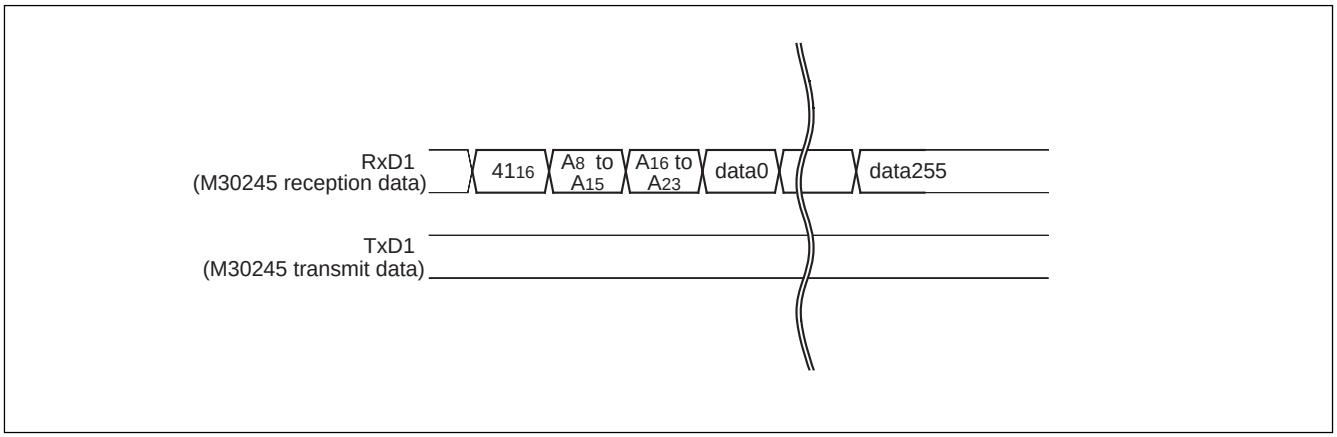



**Figure 1.181. Timing for the page program**



Downloaded from Arrow.com.

### 3. Block Erase Command

This command erases all the data in the specified block. Figure 1.182 shows the block erase timing. To execute the block erase command:
(1) Transfer the "$20_{16}$" command code with the 1st byte.
(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.
(3) Transfer the verify command code "$D0_{16}$" with the 4th byte. With the verify command code, the erase operation will start for the specified block in the flash memory. Write the highest address of the specified block for addresses $A_8$ to $A_{23}$. After block erase ends, the results of the block erase operation can be reviewed in the status register. Refer to the status register section for more details. Each block can be erase-protected with the lock bit. Refer to the data protection function section for more details.

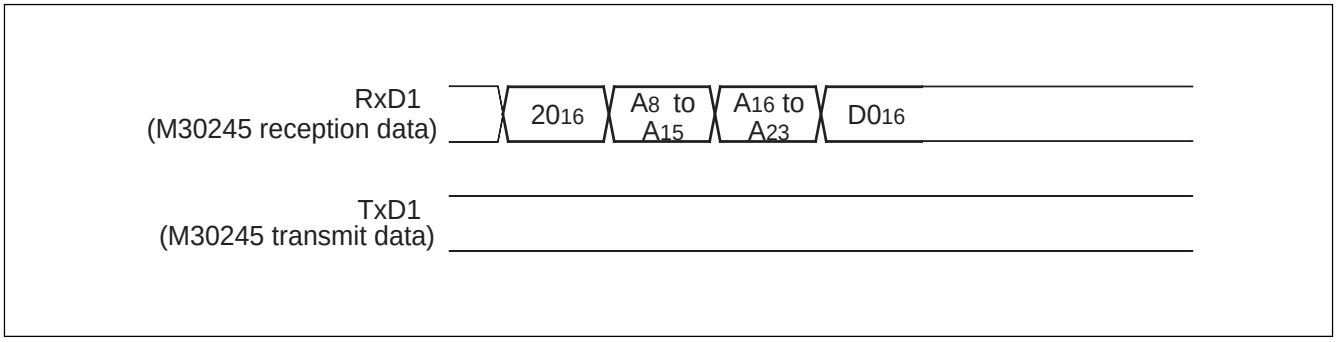



**Figure 1.182. Timing for block erasing**

### 4. Erase All Unlocked Blocks Command

This command erases the content of all blocks. Figure 1.183 shows the erase all unlocked blocks timing. To execute the erase all unlocked blocks command:
(1) Transfer the "$A7_{16}$" command code with the 1st byte.
(2) Transfer the verify command code "$D0_{16}$" with the 2nd byte. With the verify command code, the erase operation will start and continue for all blocks in the flash memory.
The results of the erase operation can be reviewed in the status register. Each block can be erase-protected with the lock bit. Refer to the data protection function and status register sections for more details.

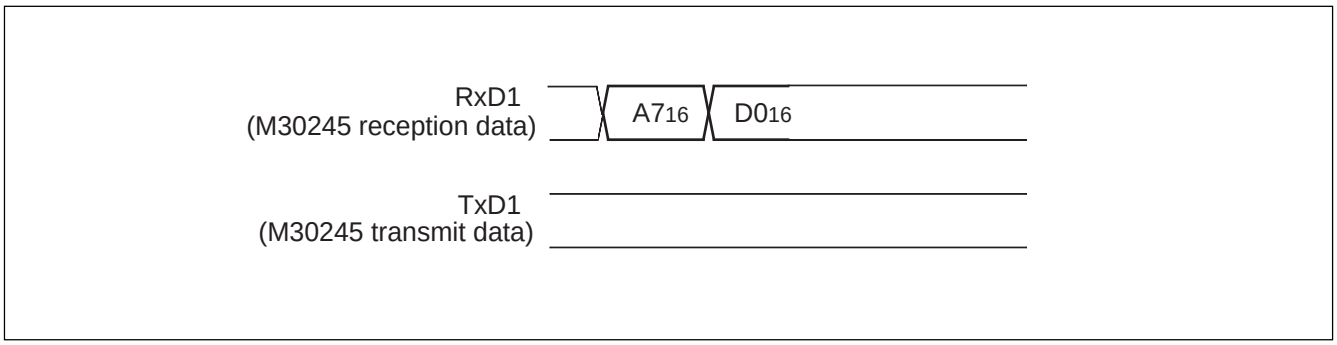



**Figure 1.183. Timing for erasing all unlocked blocks**



Downloaded from Arrow.com.

**5. Read Status Register Command**

This command reads the status information. When the "$70_{16}$" command code is sent with the 1st byte, the contents of the status register (SRD) are read with the 2nd byte and the contents of status register 1 (SRD1) are read with the 3rd byte. Figure 1.184 shows the read status register timing.

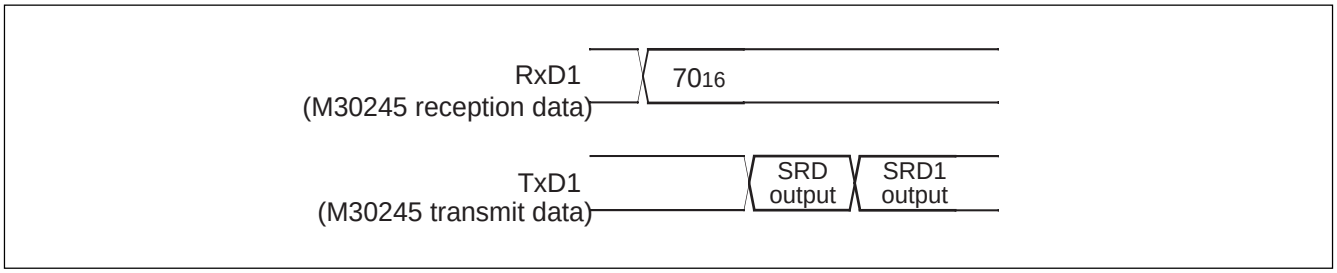



**Figure 1.184. Timing for reading the status register**

**6. Clear Status Register Command**

This command clears the bits (SR3–SR5) that are set when an erase, program or status operation ends in error. When the "$50_{16}$" command code is sent with the 1st byte, the SR3-SR5 bits are cleared. Figure 1.185 shows the clear status register timing.

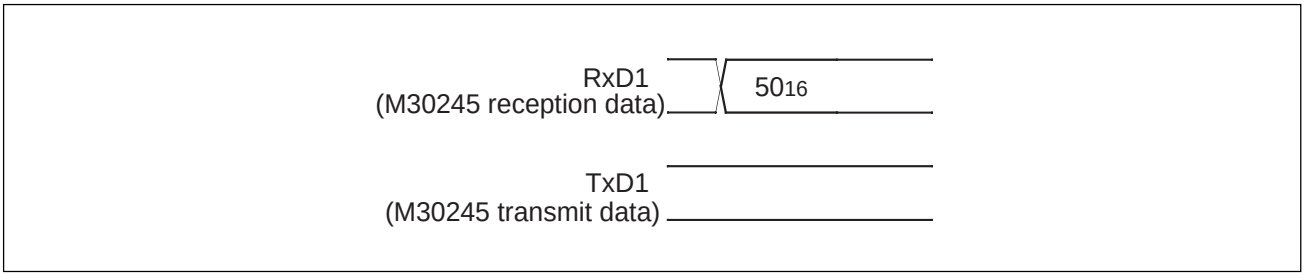



**Figure 1.185. Timing for clearing the status register**

**7. Read Lock Bit Status Command**

This command reads the lock bit status of the specified block. Figure 1.186 shows the read lock bit status timing. To execute the read lock bit status command:

(1) Transfer the "$71_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) The lock bit data of the specified block is output with the 4th byte. The 6th bit ($D_6$) of the output data is the lock bit data. Write the highest address of the specified block for addresses $A_8$ to $A_{23}$.

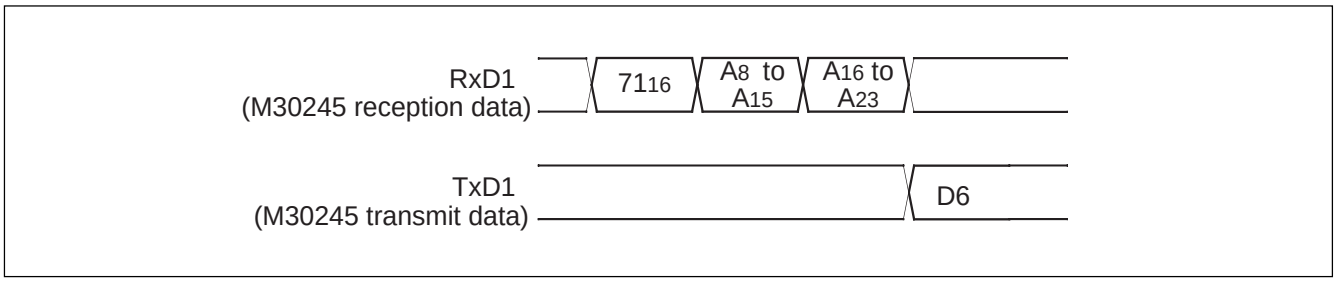



**Figure 1.186. Timing for reading lock bit status**



Downloaded from Arrow.com.

**8. Lock Bit Program Command**

This command writes "0" (lock) for the lock bit of the specified block. Figure 1.187 shows the lock bit program timing. To execute the lock bit program command:

(1) Transfer the "$77_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code "$D0_{16}$" with the 4th byte. With the verify command code, "0" is written for the lock bit of the specified block. Write the highest address of the specified block for addresses $A_8$ to $A_{23}$.

The lock bit status can be read with the read lock bit status command. Refer to the data protection function for more details about the lock bit function and reset procedure.

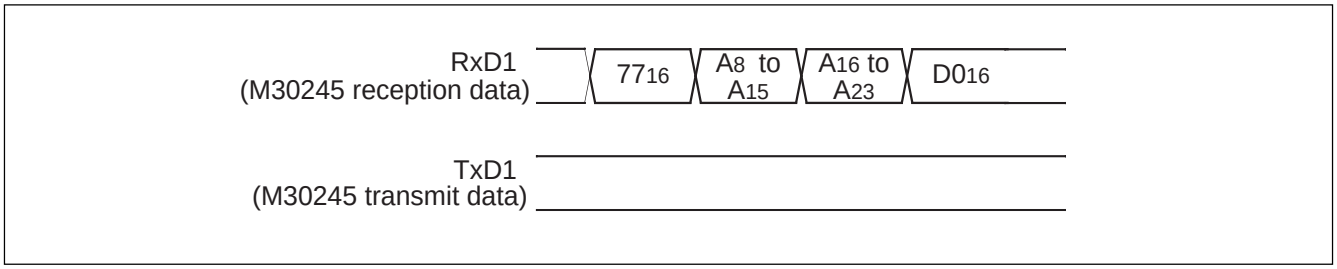



**Figure 1.187. Timing for the lock bit program**

**9. Lock Bit Enable Command**

This command enables the lock bits for all blocks. The command code "$7A_{16}$" is sent with the 1st byte of the serial transmission. This command only enables the lock bit function; it does not set the lock bit itself. Figure 1.188 shows the lock bit enable timing.

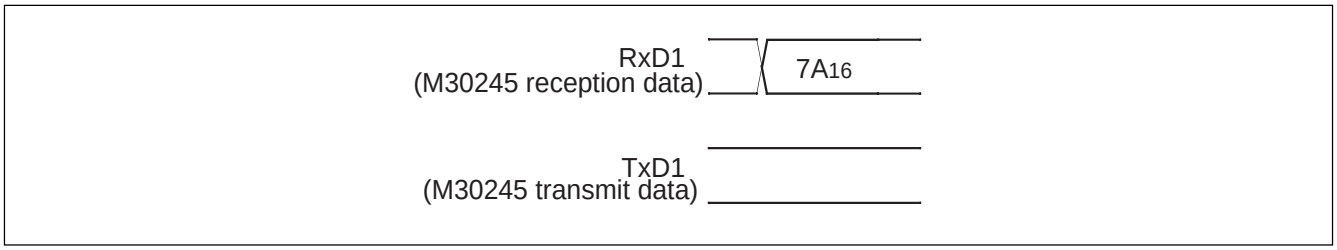



**Figure 1.188. Timing for enabling the lock bit**

**10. Lock Bit Disable Command**

This command disables the lock bit for all blocks. The command code "$75_{16}$" is sent with the 1st byte of the serial transmission. This command only disables the lock bit function; it does not set the lock bit itself. However, if an erase command is executed after executing the lock bit disable command, all "0" (locked) lock bit data is set to "1" (unlocked) after the erase operation ends. Figure 1.189 shows the lock bit disable timing. After reset the lock bit is always enabled.

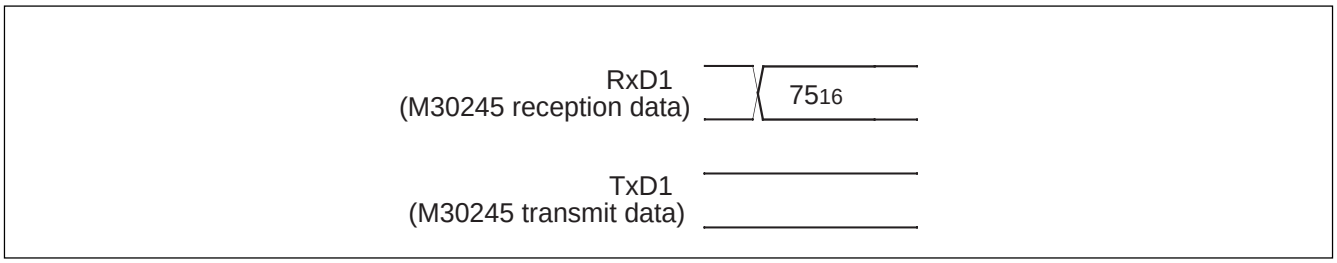



**Figure 1.189. Timing for disabling the lock bit**



Downloaded from Arrow.com.

### 11. ID Check

This command checks the ID code. Figure 1.190 shows the ID check command timing. To execute the boot ID check command:

(1) Transfer the "F5$_{16}$" command code with the 1st byte.

(2) Transfer addresses A$_0$ to A$_7$, A$_8$ to A$_{15}$ and A$_{16}$ to A$_{23}$ of the 1st byte of the ID code with the 2nd, 3rd and 4th bytes respectively.

(3) Transfer the number of data sets of the ID code with the 5th byte.

(4) The ID code is sent with the 6th byte on, starting with the 1st byte of the code.

See the ID code section for more information.

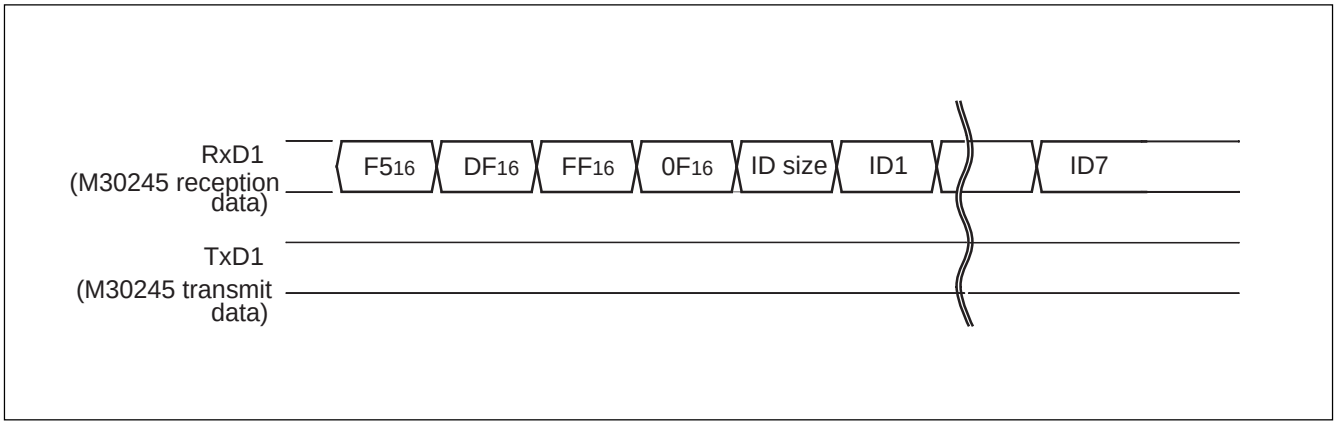



**Figure 1.190. Timing for the ID check**

### 12. Download Command

This command downloads a program to the RAM for execution. Figure 1.191 shows the download command timing. To execute the download command:

(1) Transfer the "FA$_{16}$" command code with the 1st byte.

(2) Transfer the program size with the 2nd and 3rd bytes.

(3) Transfer the check sum with the 4th byte. The check sum is added to all data sent from the 5th byte on.

(4) The program to execute is sent starting with the 5th byte.

When all data has been transmitted, if the check sum matches, the downloaded program is executed. The size of the program will vary according to the internal RAM.

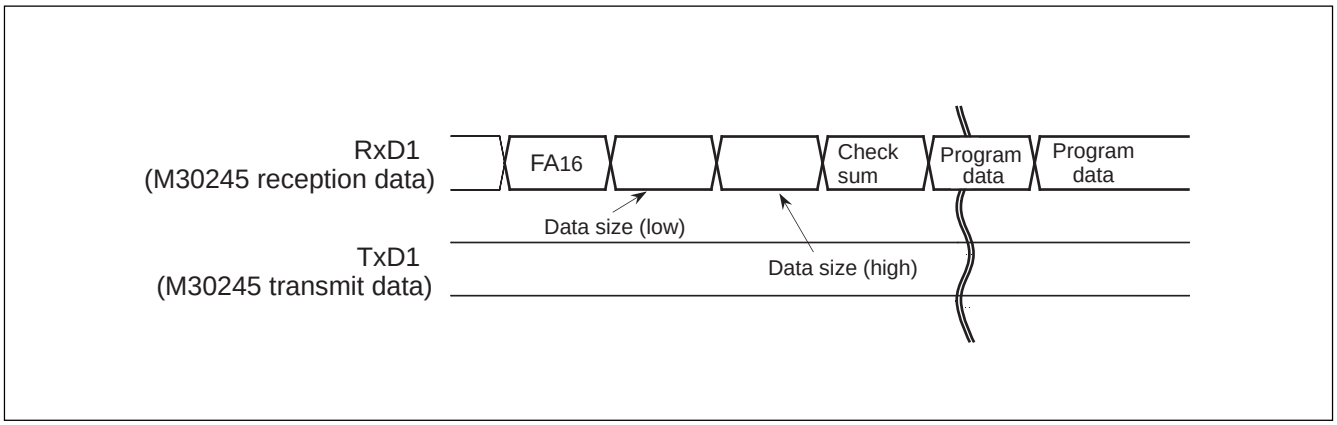



**Figure 1.191. Timing for download**



Downloaded from Arrow.com.

### 13. Version Information Output Command

This command outputs the version information of the control program stored in the boot area. Figure 1.192 shows the version information output timing. To execute the version information output command:
(1) Transfer the "FB16" command code with the 1st byte.
(2) The version information will be output from the 2nd byte on. This version data is composed of 8 ASCII code characters.

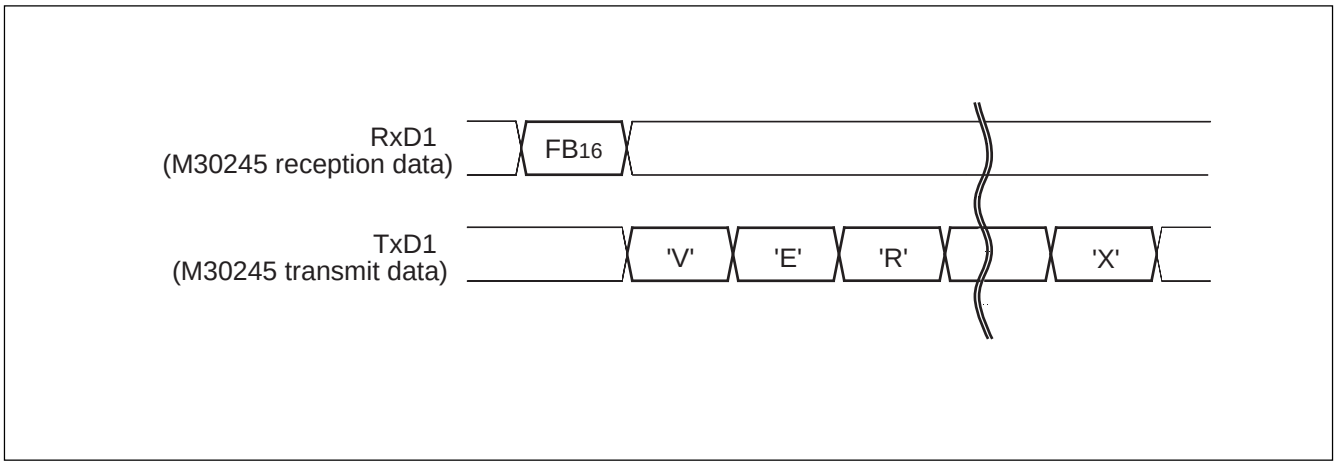



**Figure 1.192. Timing for version information output**

### 14. Boot ROM Area Output Command

This command outputs the control program stored in the boot ROM area in one page blocks (256 bytes). Figure 1.193 shows the boot ROM area output timing. To execute the boot ROM area output command:
(1) Transfer the "FC16" command code with the 1st byte.
(2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.
(3) Starting with the 4th byte, data (D0-D7) for the page (256 bytes) specified by addresses A8 to A23, will be output sequentially from the smallest address first.

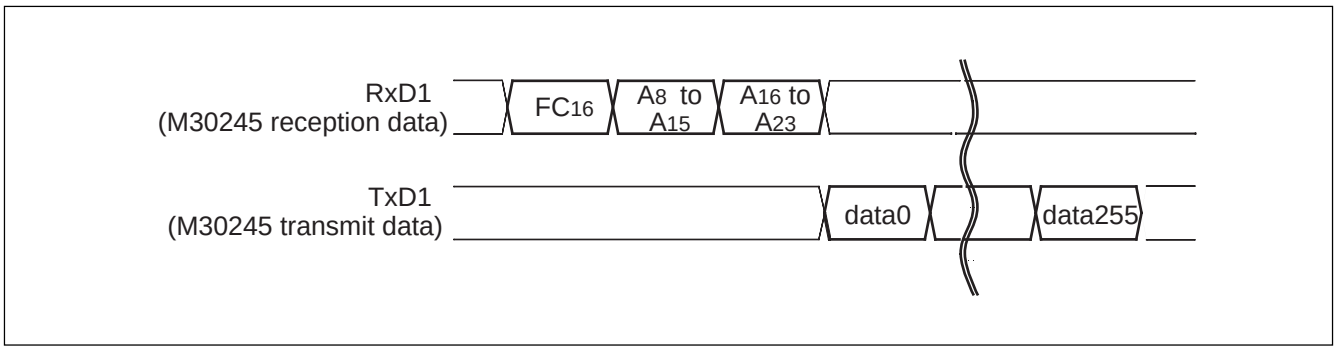



**Figure 1.193. Timing for boot ROM area output**



Downloaded from Arrow.com.

### 15. Read Check Data

This command reads the check data, that confirms that the write data, sent with the page program command, was successfully received. Figure 1.194 shows the read check data timing. To execute the read check data:
(1) Transfer the "FD16" command code with the 1st byte.
(2) The check data (low) is received with the 2nd byte and the check data (high) with the 3rd.
To use this read check data command, first execute the command and then initialize the check data. Next, execute the page program command the required number of times. Afterwards, when the read check command is executed again, the check data (for all of the read data sent with the page program command during this time) is read. The check data is the result of a CRC operation of write data.

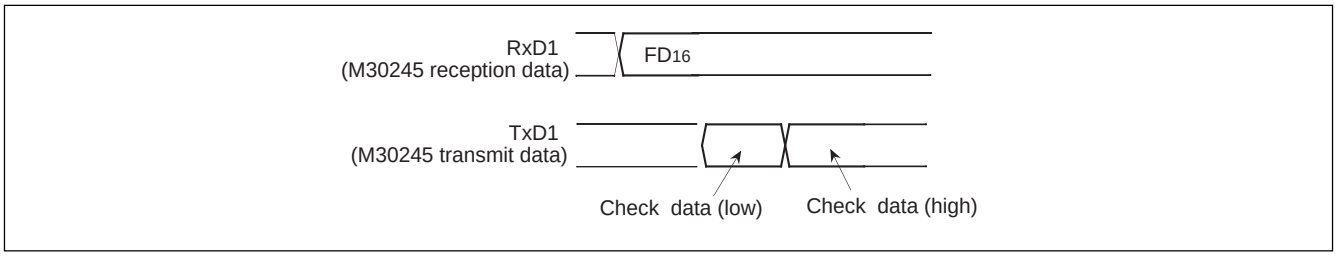



**Figure 1.194. Timing for the read check data**

### 16. Baud Rate 9600

This command changes the baud rate to 9,600 bps. Figure 1.195 shows the baud rate 9600 command timing. To execute the baud rate 9600 bps command:
(1) Transfer the "B016" command code with the 1st byte.
(2) After the "B016" check code is output with the 2nd byte, change the baud rate to 9,600 bps.

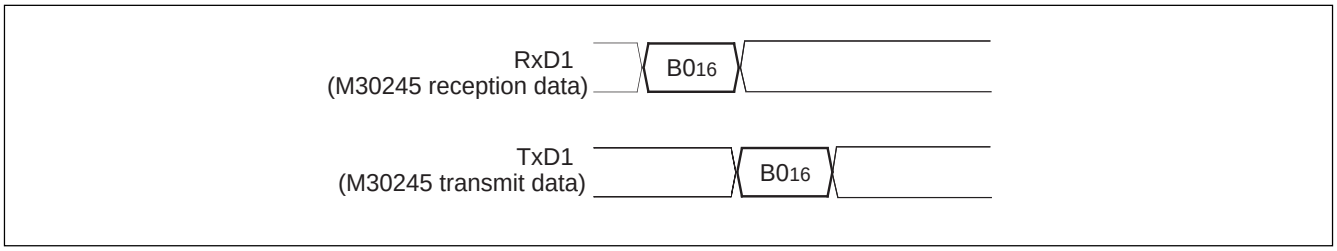



**Figure 1.195. Timing of baud rate 9600**

### 17. Baud Rate 19200

This command changes the baud rate to 19,200 bps. Figure 1.196 shows the baud rate 19200 command timing. To execute the baud rate 19200 bps command:
(1) Transfer the "B116" command code with the 1st byte.
(2) After the "B116" check code is output with the 2nd byte, change the baud rate to 19,200 bps.

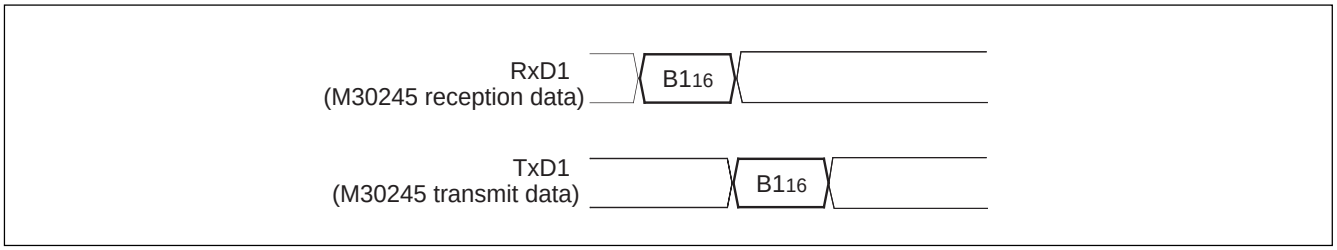



**Figure 1.196. Timing of baud rate 19200**



Downloaded from Arrow.com.

### 18. Baud Rate 38400

This command changes the baud rate to 38,400 bps. Figure 1.197 shows the baud rate 38400 command timing. To execute the baud rate 38400 bps command:

(1) Transfer the "B2$_{16}$" command code with the 1st byte.

(2) After the "B2$_{16}$" check code is output with the 2nd byte, change the baud rate to 38,400 bps.

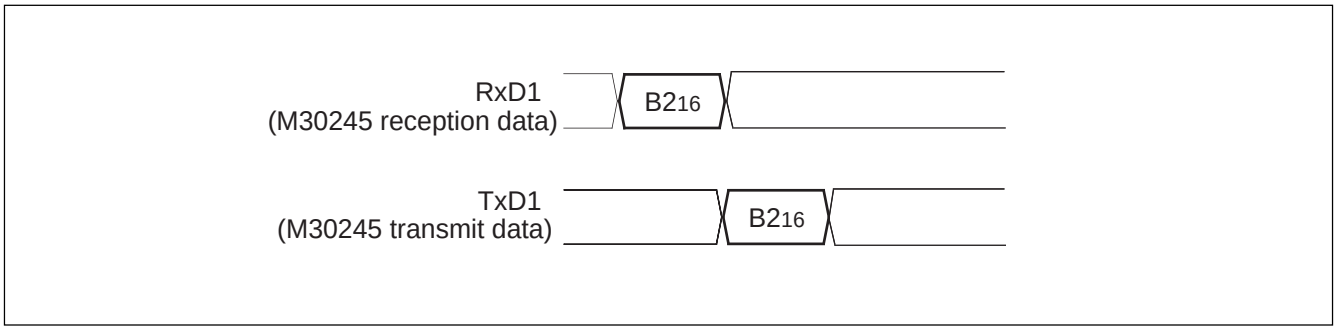


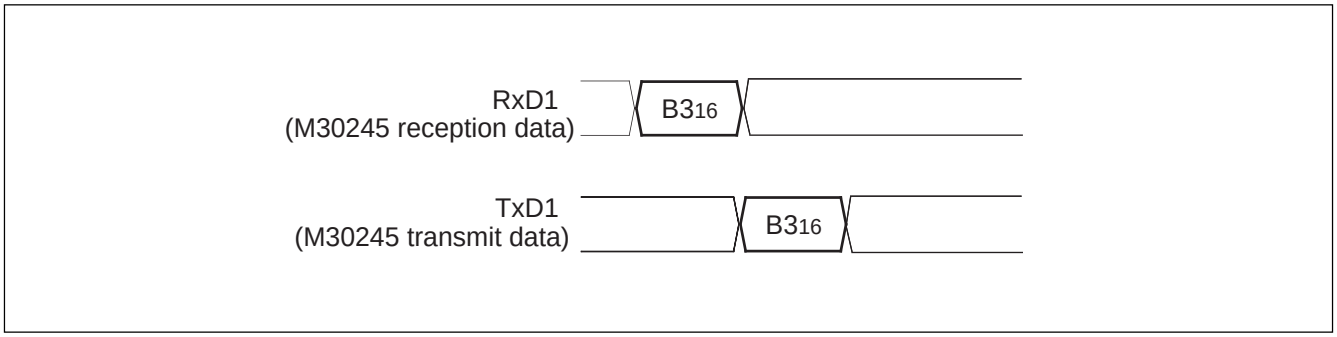

**Figure 1.197. Timing of baud rate 38400**

### 19. Baud Rate 57600

This command changes the baud rate to 57,600 bps. Figure 1.198 shows the baud rate 57600 command timing. To execute the baud rate 57600 bps command:

(1) Transfer the "B3$_{16}$" command code with the 1st byte.

(2) After the "B3$_{16}$"check code is output with the 2nd byte, change the baud rate to 57,600 bps.




**F.igure 1.198. Timing of baud rate 57600**



Downloaded from Arrow.com.

# Electrical Specifications

## Absolute Maximum Ratings

**Table 1.76. Absolute maximum ratings**

| Symbol | Parameter | | Condition | Rated value | Unit |
|---|---|---|---|---|---|
| Vcc | Supply voltage | | Vcc = AVcc = UVcc | -0.3 to 4.0 | V |
| AVcc | Analog supply voltage | | Vcc = AVcc = UVcc | -0.3 to 4.0 | V |
| UVcc | USB circuit supply voltage | | Vcc = AVcc = UVcc | -0.3 to 4.0 | V |
| $V_I$ | Input voltage | $\overline{RESET}$, CNVss, BYTE, $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_7$ $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $V_{REF}$, $X_{IN}$, D+, D- | | -0.3 to Vcc + 0.3 | V |
| | | $P7_0$ to $P7_1$ | | -0.3 to 4.0 | V |
| | | VbusDTCT | | -0.3 to 5.50 | V |
| $V_O$ | Output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $X_{OUT}$, D+, D- | | -0.3 to Vcc + 0.3 | V |
| | | $P7_0$ to $P7_1$ | | -0.3 to 4.0 | V |
| Pd | Power dissipation | | Topr = 25°C | 300 | mW |
| Topr | Operating ambient temperature | | | -20 to 85 | °C |
| Tstg | Storage temperature | | | -65 to 150 | °C |



Downloaded from Arrow.com.

## Recommended operating conditions

**Table 1.77. Recommended operating conditions (Note 1)**

| Symbol | Parameter | | Standard | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Vcc | Supply voltage | | 3.0 | 3.3 | 3.6 | V |
| AVcc | Analog supply voltage | | | Vcc | | V |
| UVcc | USB supply voltage | | 3.0 | 3.3 | 3.6 | V |
| Vss | Supply voltage | | | 0 | | V |
| AVss | Analog supply voltage | | | 0 | | V |
| $V_{IH}$ | HIGH input voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $X_{IN}$, $\overline{RESET}$, CNVss, BYTE | 0.8 Vcc | | Vcc | V |
| | | $P7_0$ to $P7_1$ | 0.8 Vcc | | 4.0 | V |
| | | D+, D- | 2.0 | | | V |
| | | VbusDTCT | 4.0 | | 5.25 | V |
| $V_{IL}$ | LOW input voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $X_{IN}$, $\overline{RESET}$, CNVss, BYTE | | | 0.2 Vcc | V |
| | | $P7_0$ to $P7_1$ | | | 0.2 Vcc | V |
| | | D+, D- | | | 0.8 | V |
| | | VbusDTCT | | | 1.0 | V |
| $I_{OH}$ (peak) | HIGH peak output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | | | -10.0 | mA |
| $I_{OH}$ (avg) | HIGH average output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | | | -5.0 | mA |
| $I_{OL}$ (peak) | LOW peak output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | | | 10.0 | mA |
| $I_{OL}$ (avg) | LOW average output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | | | 5.0 | mA |
| $f(X_{IN})$ | Main clock input oscillation frequency | | (Note 4) | | 16 | MHz |
| $f(X_{CIN})$ | Sub clock oscillation frequency | | | 32.768 | 50 | kHz |

Note 1: Vcc = 3.0V to 3.6V at Topr = -20 to 85°C unless otherwise stated.
Note 2: The mean output current is the mean values within 100ms.
Note 3: The total $I_{OL}$ (peak) and $I_{OH}$ (peak) for Ports P0, P1, P2, $P8_6$, $P8_7$, P9 and P10 must be 80mA max. The total $I_{OL}$ (peak) for Ports P3, P4, P5, P6, P7 and $P8_0$ to $P8_4$ must be 80mA max. The total $I_{OH}$ (peak) for ports P3, P4, P5, P6, $P7_2$ to $P7_7$ and $P8_0$ to $P8_4$ must be 80mA max.
Note 4: When using the USB function, set $f(X_{IN})$ to 4MHz or higher.



Downloaded from Arrow.com.

## Electrical characteristics

**Table 1.78. Electrical characteristics (Note 1)**

| Symbol | Parameter | | Measuring condition | Standard | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| $V_{OH}$ | HIGH output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_2$, $P6_4$ to $P6_6$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | $I_{OH}$ = -1mA | 2.5 | | | V |
| | | $P6_3$, $P6_7$ | $I_{OH}$ = -10mA | 2.0 | | | V |
| | | $X_{OUT}$ HIGHPOWER | $I_{OH}$ = -0.1mA | 2.5 | | | V |
| | | $X_{OUT}$ LOWPOWER | $I_{OH}$ = -50µA | 2.5 | | | |
| | | $X_{COUT}$ HIGHPOWER | $I_{OH}$ = -1µA | 2.5 | | | V |
| | | $X_{COUT}$ LOWPOWER | $I_{OH}$ = -0.5µA | 2.5 | | | |
| $V_{OL}$ | LOW output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_2$, $P6_4$ to $P6_6$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | $I_{OL}$ = 1mA | | | 0.5 | V |
| | | $P6_3$, $P6_7$, $P7_0$ to $P7_7$ (P7 high drive mode) | $I_{OL}$ = 10mA | | | 0.8 | V |
| | | $X_{OUT}$ HIGHPOWER | $I_{OL}$ = 0.1mA | | | 0.5 | V |
| | | $X_{OUT}$ LOWPOWER | $I_{OL}$ = 50µA | | | 0.5 | |
| | | $X_{COUT}$ HIGHPOWER | $I_{OL}$ = 1µA | | | 0.5 | V |
| | | $X_{COUT}$ LOWPOWER | $I_{OL}$ = 0.5µA | | | 0.5 | |
| $V_T+ - V_T-$ | Hysteresis | $\overline{HOLD}$, $\overline{RDY}$, $TA0_{IN}$ to $TA4_{IN}$, $\overline{INT0}$ to $\overline{INT2}$, $\overline{AD_{TRG}}$, $\overline{CTS0}$ to $\overline{CTS3}$, CLK0, CLK1, $TA2_{OUT}$ to $TA4_{OUT}$, $\overline{NMI}$, $\overline{KI_0}$ to $\overline{KI_7}$, RXD0 to RXD3, SCL, SDA | | 0.2 | | 1.0 | V |
| | | $\overline{RESET}$ | | 0.2 | | 1.8 | V |
| | | VbusDTCT | | 1.0 | | | V |
| $I_{IH}$ | HIGH input current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $X_{IN}$, $\overline{RESET}$, CNVss, BYTE | $V_I$ = Vcc | | | 4 | µA |
| | | VbusDTCT | | | | 50 | µA |
| $I_{IL}$ | LOW input current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$, $X_{IN}$ RESET, CNVss, BYTE, VbusDTCT | $V_I$ = 0V | | | -4 | µA |
| $R_{PULLUP}$ | Pull-up resistance | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$, $P9_3$, $P10_0$ to $P10_7$ | $V_I$ = 0V | 30.0 | 50.0 | 167.0 | kΩ |
| $R_{fXIN}$ | Feedback resistance | $X_{IN}$ | | | 1.0 | | MΩ |
| $R_{fXCIN}$ | | $X_{CIN}$ | | | 10 | | MΩ |
| $V_{RAM}$ | RAM retention voltage | | When clock is stopped | 2.0 | | | V |
| Icc | Power supply current | In single-chip mode, the output pins are open and other pins are Vss | $f(X_{IN})$ = 16MHz Square wave, no division, USB off | | 16 | | mA |
| | | | $f(X_{IN})$ = 16MHz Square wave, no division, USB on | | 25 | 43 | mA |
| | | | $f(X_{CIN})$ = 32kHz Square wave | | 30 | | µA |
| | | | $f(X_{CIN})$ = 32kHz When a WAIT instruction is executed. (Note 2) | | 12 | | µA |
| | | | Flash memory version: Topr = 25°C, when clock is stopped, USB suspend mode | | | 235 | µA |
| | | | Flash memory version: Topr = 45°C, when clock is stopped, USB suspend mode | | | 420 | µA |
| | | | Mask ROM version: Topr = 25°C, when clock is stopped, USB suspend mode | | | 95 | µA |
| | | | Mask ROM version: Topr = 45°C, when clock is stopped, USB suspend mode | | | 190 | µA |

Note 1 : Vcc = 3.0V to 3.6V, Vss = 0V at Topr = -20°C to 85°C, $f(X_{IN})$ = 16MHz unless otherwise stated.
Note 2 : With one timer operated using fc32.



Downloaded from Arrow.com.

**Table 1.79. USB electrical characteristics (Note 1)**

| Symbol | Parameter | Measuring Condition | | | Standard | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | | | | Min | Typ | Max | |
| $V_{OH}$ | D+, D- | $I_{OH}$ /$I_{OL}$ = +/- 18.3mA, UVCC = 3.00V, Rx = 33Ω | | | 2.2 | | 3.6 | V |
| $V_{OL}$ | D+, D- | $I_{OH}$ /$I_{OL}$ = +/- 18.3mA, UVCC = 3.00V, Rx = 33Ω | | | 0 | | 0.8 | V |
| $I_{susp}$ | Suspend current | USB suspend mode, internal clock stopped, with / without Vbus Detect | Flash memory version | Topr=25°C | | | 235 | µA |
| | | | | Topr=45°C | | | 420 | µA |
| | | | Mask ROM version | Topr=25°C | | | 95 | µA |
| | | | | Topr=45°C | | | 190 | µA |

Note1 : Vcc = 3.0V to 3.6V, Vss = 0V, Topr = -20°C to 85°C unless otherwise specified.

**Table 1.80. A/D conversion characteristics (Note 1)**

| Symbol | Parameter | | Measuring condition | Standard | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min | Typ | Max | |
| - | Resolution | | $V_{REF}$ = Vcc | | | 10 | Bits |
| - | Absolute accuracy | Sample and hold function not used (10 bit) | $V_{REF}$ = Vcc | | | ±4 | LSB |
| | | Sample and hold function used (10 bit) | $V_{REF}$ = Vcc | | | ±4 | LSB |
| | | Sample and hold function not used (8 bit) | $V_{REF}$ = Vcc | | | ±2 | LSB |
| | | Sample and hold function used (8 bit) | $V_{REF}$ = Vcc | | | ±2 | LSB |
| $R_{LADDER}$ | Ladder resistance | | $V_{REF}$ = Vcc | 10 | | 40 | kΩ |
| $t_{CONV}$ | Conversion time (10 bit) | | $V_{REF}$ = Vcc | 3.3 | | | µs |
| $t_{CONV}$ | Conversion time (8 bit) | | $V_{REF}$ = Vcc | 2.8 | | | µs |
| $t_{SAMP}$ | Sampling time | | | 0.3 | | | µs |
| $V_{REF}$ | Reference voltage | | | | Vcc | | V |
| $V_{IA}$ | Analog input voltage | | | 0 | | Vcc | V |

Note 1 : Vcc, AVcc, $V_{REF}$ = 3.3V, Vss, AVss = 0V, Topr = 25°C, f($X_{IN}$) = 16MHz.
Note 2 : Divide the frequency if f($X_{IN}$) exceeds 10MHz, and make ΦAD equal to or less than 10MHz.

**Table 1.81. Flash memory version electrical characteristics (Note 1)**

| Symbol | Parameter | Measuring condition | Standard | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | |
| - | Page Program Time | | | 6 | 120 | ms |
| - | Block erase time | | | 50 | 600 | ms |
| - | Erase all unlocked blocks time | | | 50xN (Note 2) | 60xN (Note 2) | ms |
| - | Lock bit program time | | | 6 | 120 | ms |

Note 1: Vcc = 3.0V to 3.6V, Vss = 0V, Topr = 0°C to 60°C
Note 2: N denotes the number of block erases.



Downloaded from Arrow.com.

## Timing requirements (Vcc = 3.3V, Vss=0V, Topr =-20 °C to 85 °C unless otherwise stated)

**Table 1.82. External clock input**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc | External clock input cycle time | 62.5 | | ns |
| tw(H) | External clock input HIGH pulse width | 29.5 | | ns |
| tw(L) | External clock input LOW pulse width | 29.5 | | ns |
| tr | External clock rise time | | 10 | ns |
| tf | External clock fall time | | 10 | ns |

**Table 1.83. Memory expansion and microprocessor modes**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tac1 (RD-DB) | Data input access time (no wait) | | (Note 1) | ns |
| tac2 (RD-DB) | Data input access time (with wait) | | (Note 2) | ns |
| tsu (DB-RD) | Data input setup time | 50 | | ns |
| tsu (RDY-BCLK) | $\overline{\text{RDY}}$ input setup time | | 40 | ns |
| tsu (HOLD-BCLK) | $\overline{\text{HOLD}}$ input setup time | | 105 | ns |
| th (RD-DB) | Data input hold time | 0 | | ns |
| th (BCLK-RDY) | $\overline{\text{RDY}}$ input hold time | 0 | | ns |
| th (BCLK-HOLD) | $\overline{\text{HOLD}}$ input hold time | 0 | | ns |

Note 1: tac1 = tcyc / 2 - 60nS
Note 2: tac2 = (m+0.5) x tcyc - 60nS
m = number of wait states (1 to 3)

**Table 1.84. Timer A input (counter input in event counter mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(TA) | AiIN input cycle time | 100 | | ns |
| tw(TAH) | TAiIN input HIGH pulse width | 50 | | ns |
| tw(TAL) | TAiIN input LOW pulse width | 40 | | ns |

**Table 1.85. Timer A input (gating input in timer mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(TA) | AiIN input cycle time | Note | | ns |
| tw(TAH) | TAiIN input HIGH pulse width | Note | | ns |
| tw(TAL) | TAiIN input LOW pulse width | Note | | ns |

Note: When using the external gating mode feature, the width of $TAi_{IN}$ needs to be greater than the period of the clock source selected.



Downloaded from Arrow.com.

### Timing requirements (Vcc = 3.3V, Vss=0V, Topr =-20 °C to 85 °C unless otherwise stated)

**Table 1.86. Timer A input (external trigger input in one-shot timer mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(TA) | AiIN input cycle time | 200 | | ns |
| tw(TAH) | TAiIN input HIGH pulse width | 100 | | ns |
| tw(TAL) | TAiIN input LOW pulse width | 100 | | ns |

**Table 1.87. Timer A input (external trigger input in pulse width modulation mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tw(TAH) | TAiIN input HIGH pulse width | 100 | | ns |
| tw(TAL) | TAiIN input LOW pulse width | 100 | | ns |

**Table 1.88. Timer input (up/down input in event counter mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(UP) | AiOUT input cycle time | 2000 | | ns |
| tw(UPH) | TAiOUT input HIGH pulse width | 1000 | | ns |
| tw(UPL) | TAiOUT input LOW pulse width | 1000 | | ns |
| tsu(UP-TIN) | TAiOUT input setup time | 400 | | ns |
| th(TIN-UP) | TAiOUT input hold time | 400 | | ns |

**Table 1.89. A/D trigger input**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(AD) | $\overline{AD_{TRG}}$ input cycle time(triggerable minimum) | 1000 | | ns |
| tw(ADL) | $\overline{AD_{TRG}}$ input LOW pulse width | 125 | | ns |

**Table 1.90. External interrupt INTi inputs**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tw(INH) | $\overline{INT_I}$ input HIGH pulse width | 250 | | ns |
| tw(INL) | $\overline{INT_I}$ input LOW pulse width | 250 | | ns |



Downloaded from Arrow.com.

### Timing requirements (Vcc = 3.3V, Vss=0V, Topr =-20°C to 85 ℃ unless otherwise stated)

**Table 1.91. Serial I/O timing**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tc(CK) | CLKi input cycle time | 160 | | ns |
| tw(CKH) | CLKi input HIGH pulse width | 60 | | ns |
| tw(CKL) | CLKi input LOW pulse width | 60 | | ns |
| tsu(D-C) | RxDi input setup time | 60 | | ns |
| th(C-D) | RxDi input hold time | 20 | | ns |

**Table 1.92. Vbus Detect interrupt**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| tw(INT) | VbusDTCT Interrupt pulse width | 50 | | μS |

**Table 1.93. Serial Sound Interface (SSI)**

| Symbol | Parameter | | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min | Max | |
| tc(SCK) | SCKi input cycle time | | 62.5 | | ns |
| tw(SCKH) | SCKi input HIGH pulse width | | 29.5 | | ns |
| tw(SCKL) | SCKi input LOW pulse width | | 29.5 | | ns |
| tsu(SRxD-SCK) | SRxDi input setup time | | 10 | | ns |
| th(SCK-SRxD) | SRxDi input hold time | | 10 | | ns |
| t1(SCK-WS) | SCKP=0 | SCKi rising edge to WS edge | 10 | | ns |
| | SCKP=1 | SCKi falling edge to WS edge | 10 | | ns |
| t2(WS-SCK) | SCKP=0 | WS edge to SCKi rising edge | 10 | | ns |
| | SCKP=1 | WS edge to SCKi falling edge | 10 | | |

**Table 1.94. AND Flash Control timing**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| th(OE-D) | AND_DATA (status data) input hold time | 0 | | ns |
| th2(SC-D) | AND_DATA input hold time | 0 | | ns |
| tsu(D-OE) | AND_DATA (status data) input setup time | 50 | | ns |
| tsu(D-SC) | AND_DATA input setup time | 43 | | ns |



Downloaded from Arrow.com.

## Switching characteristics (Vcc = 3.3V, Vss=0V, Topr =-20 ℃ to 85 ℃ unless otherwise stated)

**Table 1.95. Memory expansion mode and microprocessor mode**

| Symbol | Parameter | Measuring condition | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min. | Max. | |
| td (BCLK-AD) | Address output delay time | See Figure 1.199<br><br>VIL = 0.2Vcc,<br>VIH = 0.8Vcc,<br>VOL = 0.5Vcc,<br>VOH = 0.5Vcc | | 30 | ns |
| th (BCLK-AD) | Address output hold time | | 0 | | ns |
| th (RD-AD) | Address output hold time | | 0 | | ns |
| th (WR-AD) | Address output hold time | | 0 | | ns |
| td (BCLK-CS) | Chip select output delay time | | | 30 | ns |
| th (BCLK-CS) | Chip select output hold time | | 0 | | ns |
| td (BCLK-ALE) | ALE signal output delay time | | | 30 | ns |
| th (BCLK-ALE) | ALE signal output hold time | | 0 | | ns |
| td (BCLK-RD) | RD signal output delay time | | | 30 | ns |
| th (BCLK-RD) | RD signal output hold time | | 0 | | ns |
| td (BCLK-WR) | WR signal output delay time | | | 30 | ns |
| th (BCLK-WR) | WR signal output hold time | | 0 | | ns |
| td (BCLK-DB) | Data output delay time | | | 40 | ns |
| th (BCLK-DB) | Data output tristate time | | 0 | | ns |
| td (DB-WR) | Data output delay time | | (Note 1) | | ns |
| th (WR-DB) | Data output hold time | | (Note 2) | | ns |
| th (WR-CS) | Write high to Chip select high time | | (Note 3) | | ns |
| td (BCLK-HLDA) | $\overline{\text{HLDA}}$ output delay time | | | 40 | ns |

Note 1:Calculated according toothe BCLK frequency as shown below:

(When PM16 = 0) td (DB-WR) = (m-0.5) X tcyc - 40nS

(When PM16 = 1) td (DB-WR) = m X tcyc - 40nS

} 0 Wait selected: m = 1
1 Wait selected: m = 1
2 Wait selected: m = 2
3 Wait selected: m = 3

Note 2: Calculated as follows: th (WR-DB) = tcyc / 2

Note 3: Calculated as follows: th (WR-CS) = tcyc / 2

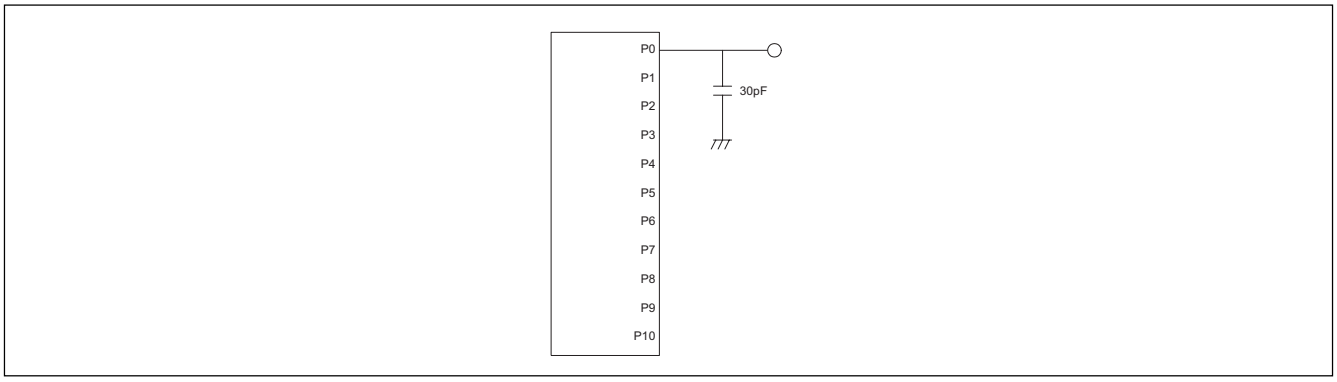



**Figure 1.199. Port P0 to P10 measurement circuit**



Downloaded from Arrow.com.

### Switching characteristics (Vcc = 3.3V, Vss=0V, Topr =-20 ℃ to 85 ℃ unless otherwise stated)

**Table 1.96. Serial I/O switching**

| Symbol | Parameter | | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min | Max | |
| td(C-Q) | TxDi output delay time | External clock is selected as transfer clock | | 80 | ns |
| | | Internal clock is selected as transfer clock | | 30 | ns |
| th(C-Q) | TxDi hold time | | 0 | | ns |
| tr(CK) | CLKi output rise time | Internal clock is selected as transfer clock | | 7 | ns |
| tf(CK) | CLKi output fall time | Internal clock is selected as transfer clock | | 7 | ns |

**Table 1.97. Serial Sound Interface (SSI)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| td(WS-XMT) | XMTi output delay time (WS based) | | 20 | ns |
| td(SCK-XMT) | XMTi output delay time (SCK based) | | 20 | ns |

**Table 1.98. AND Flash Control switching**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| td(D-SC) | AND_DATA (program data) output delay time | | (Note 1) | ns |
| td(D-WE) | AND_DATA (command/address data) output delay time | (Note 2) | | ns |
| th1(SC-D) | AND_DATA (program data) output hold time | (Note 3) | | ns |
| th(WE-D) | AND_DATA (command/address data) output hold time | (Note 4) | | ns |
| tw(OEL) | $\overline{\text{AND_OE}}$ LOW pulse width | (Note 5) | | ns |
| tw1(SCH) | AND_SC write HIGH pulse width | (Note 6) | | ns |
| tw2(SCH) | AND_SC read HIGH pulse width | (Note 7) | | ns |
| tw(WEL) | $\overline{\text{AND_WE}}$ LOW pulse width | (Note 8) | | ns |

Note 1: td(D-SC) = 0.5tcyc - 15nS
Note 2: td(D-WE) = 1.5tcyc - 43nS
Note 3: th1(SC-D) = 1.5tcyc - 30nS
Note 4: th(WE-D) = 0.5tcyc nS
Note 5: tw(OEL) = 1.5tcyc - 10nS
Note 6: tw1(SCH) = tcyc - 15nS
Note 7: tw2(SCH) = 1.5tcyc - 10nS
Note 8: tw(WEL) = tcyc - 15nS



Downloaded from Arrow.com.

Timing Diagrams

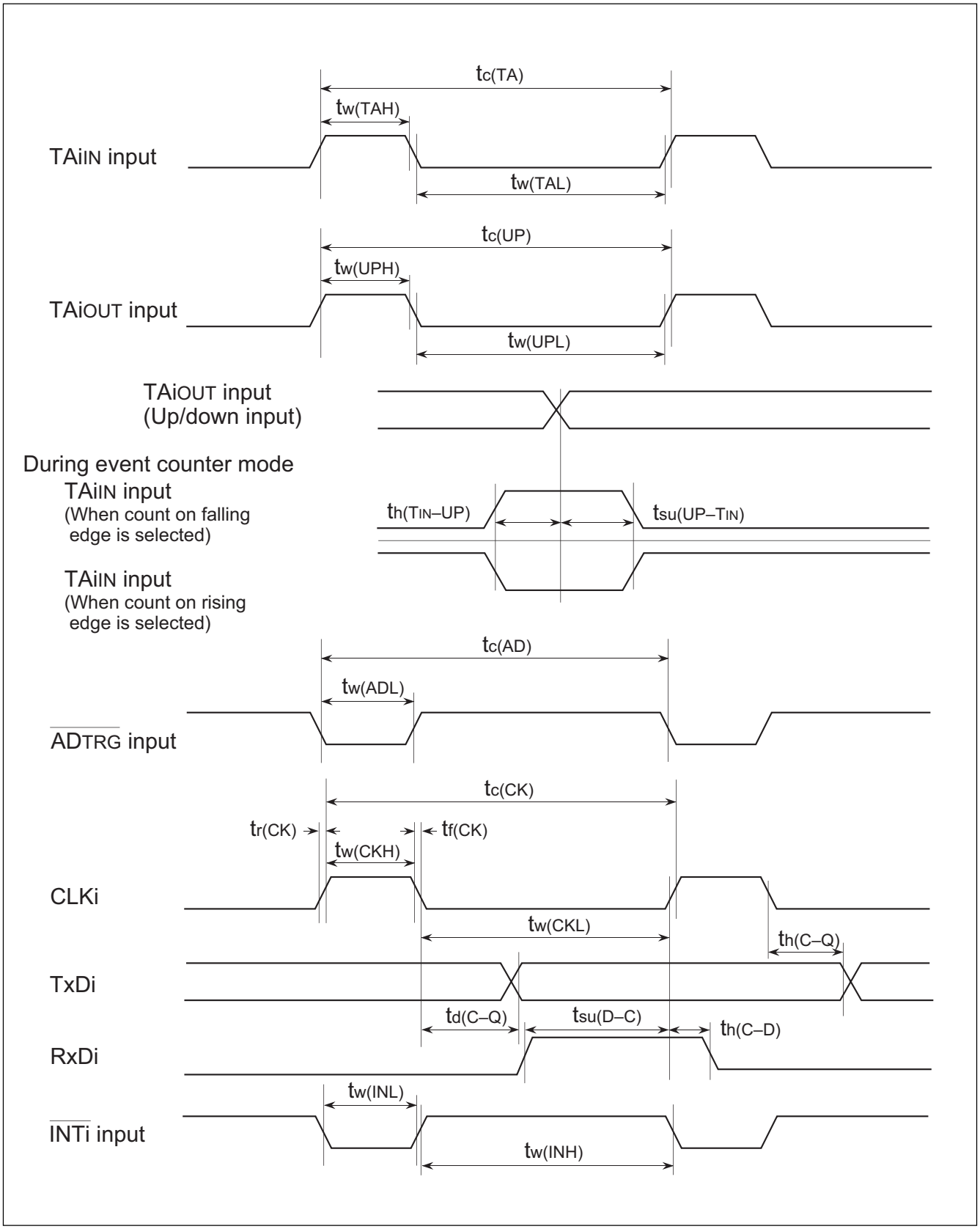


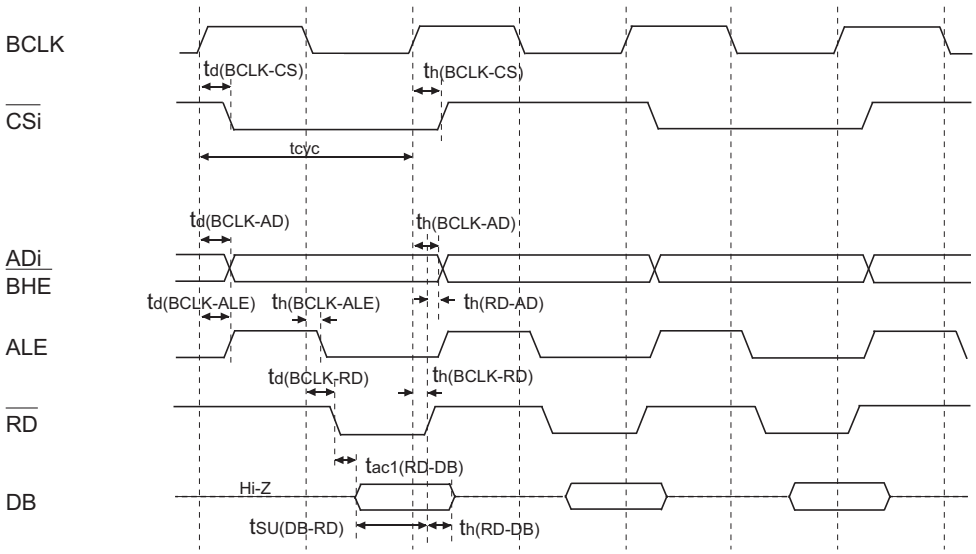

**Figure 1.200. Timing diagram 1**



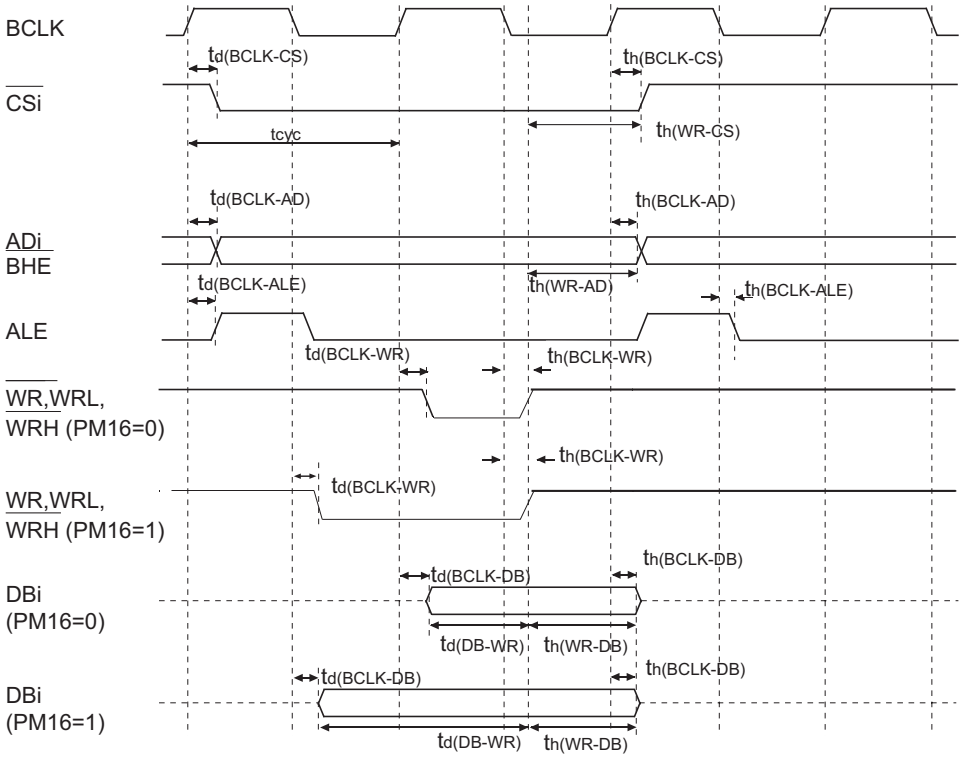
Downloaded from Arrow.com.

**Memory Expansion Mode and Microprocessor Mode**
**(With no wait)**

**Read timing**




**Write timing**




**Figu!re 1.201. Timing diagram 2**



Downloaded from Arrow.com.

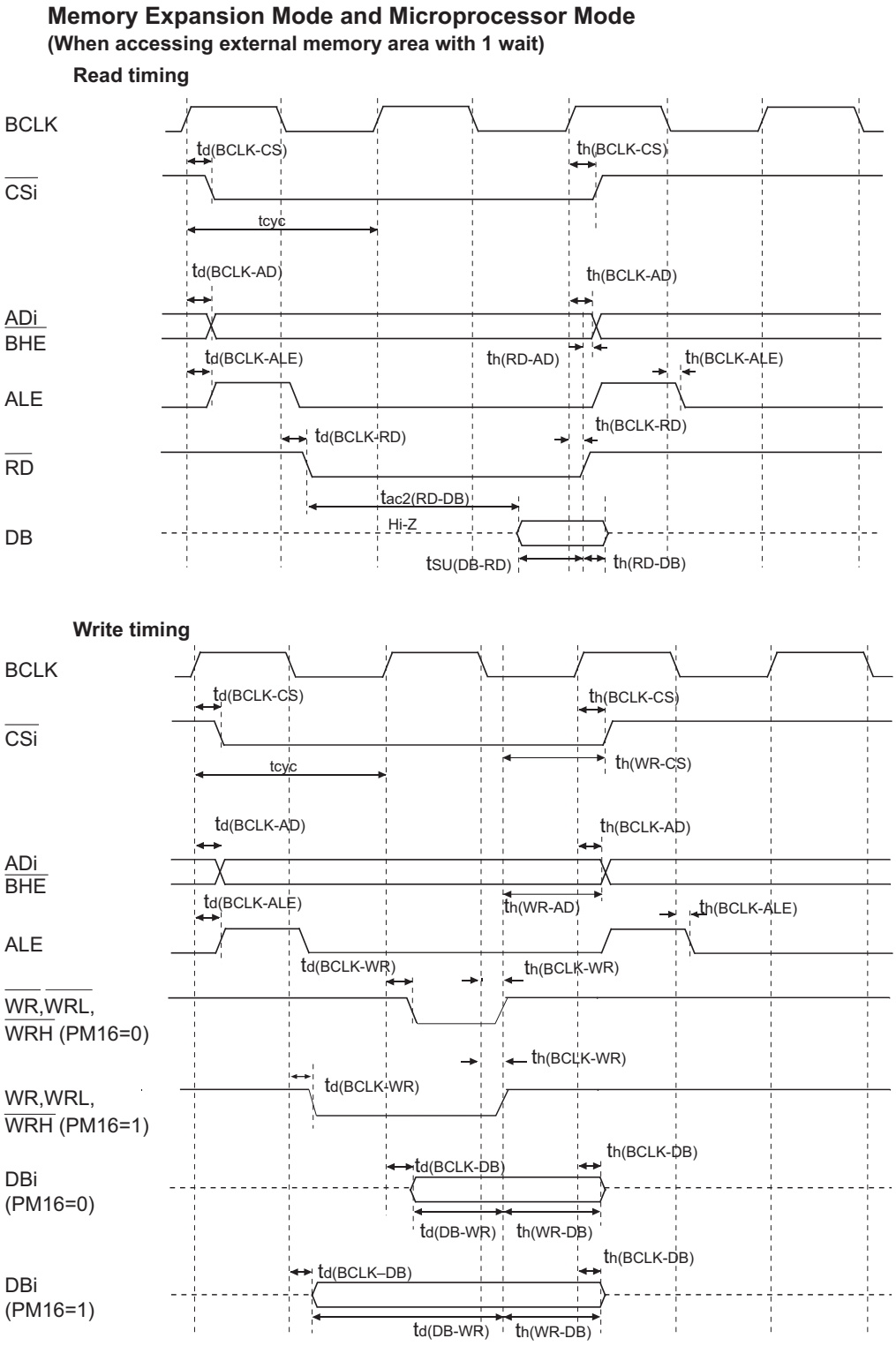


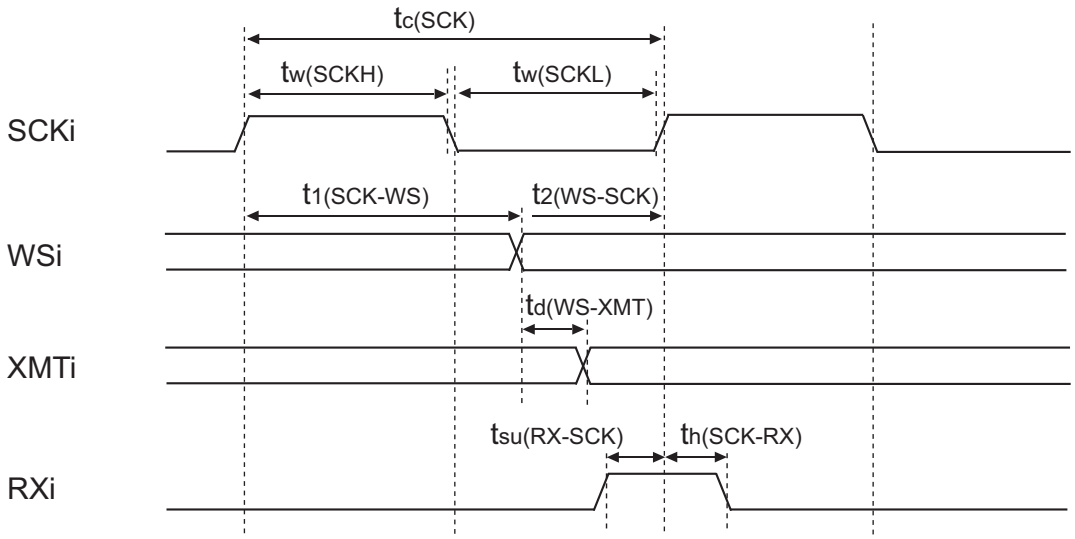

**Figure 1.202. Timing diagram 3**



Downloaded from Arrow.com.

## Serial Sound Interface Timing

**(SCKP=0, SCK falling edge is before WS edge)**




**(SCKP=0, SCK falling edge is after WS edge)**

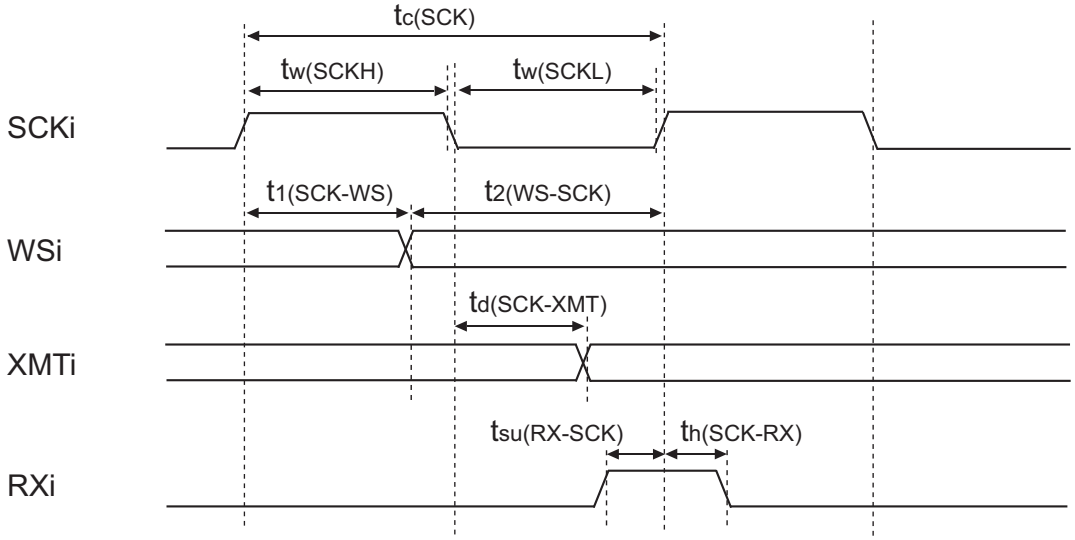


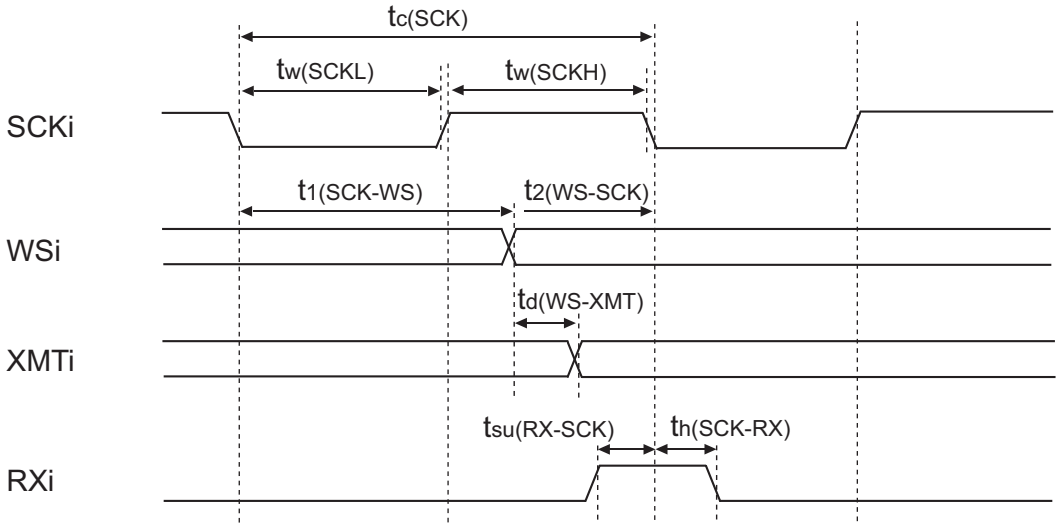

**Figure 1.203. Serial Sound Interface timing diagram 1**



Downloaded from Arrow.com.

## Serial Sound Interface Timing

**(SCKP=1, SCK rising edge is before WS edge)**




**(SCKP=1, SCK rising edge is after WS edge)**

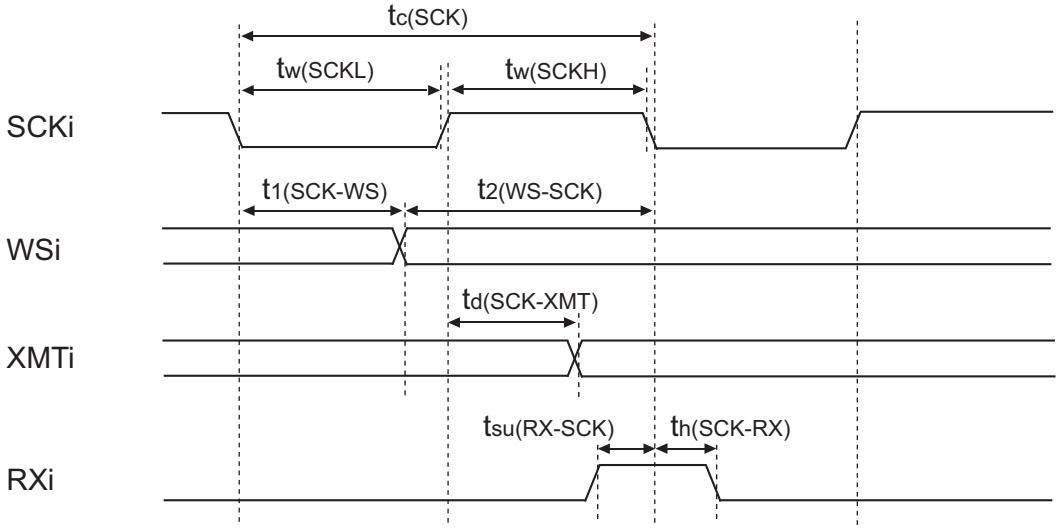



**Figure 1.204. Serial Sound Interface timing diagram 2**



Downloaded from Arrow.com.

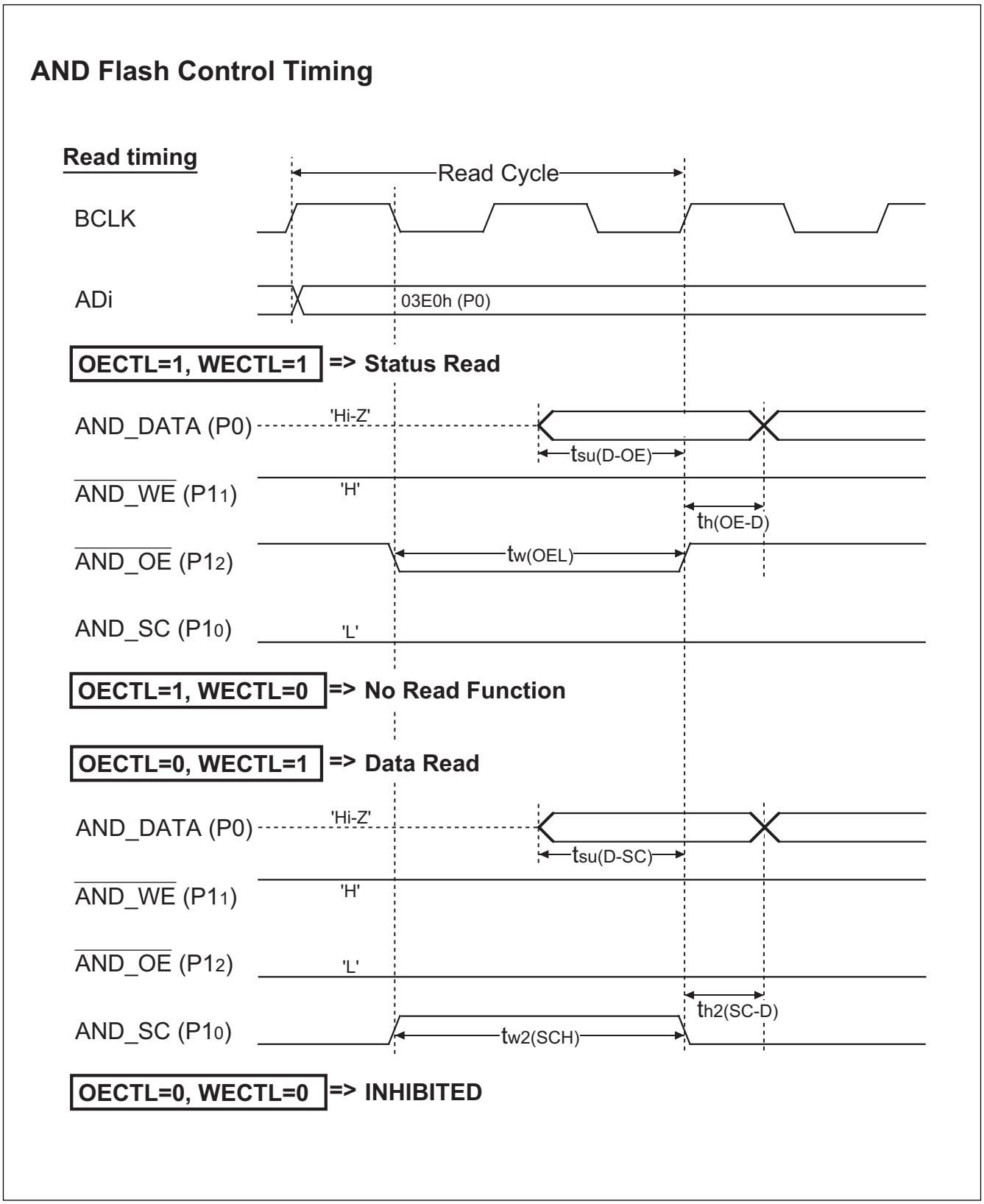



**Figure 1.205. AND Flash Control read timing diagram**



Downloaded from Arrow.com.

## AND Flash Control Timing

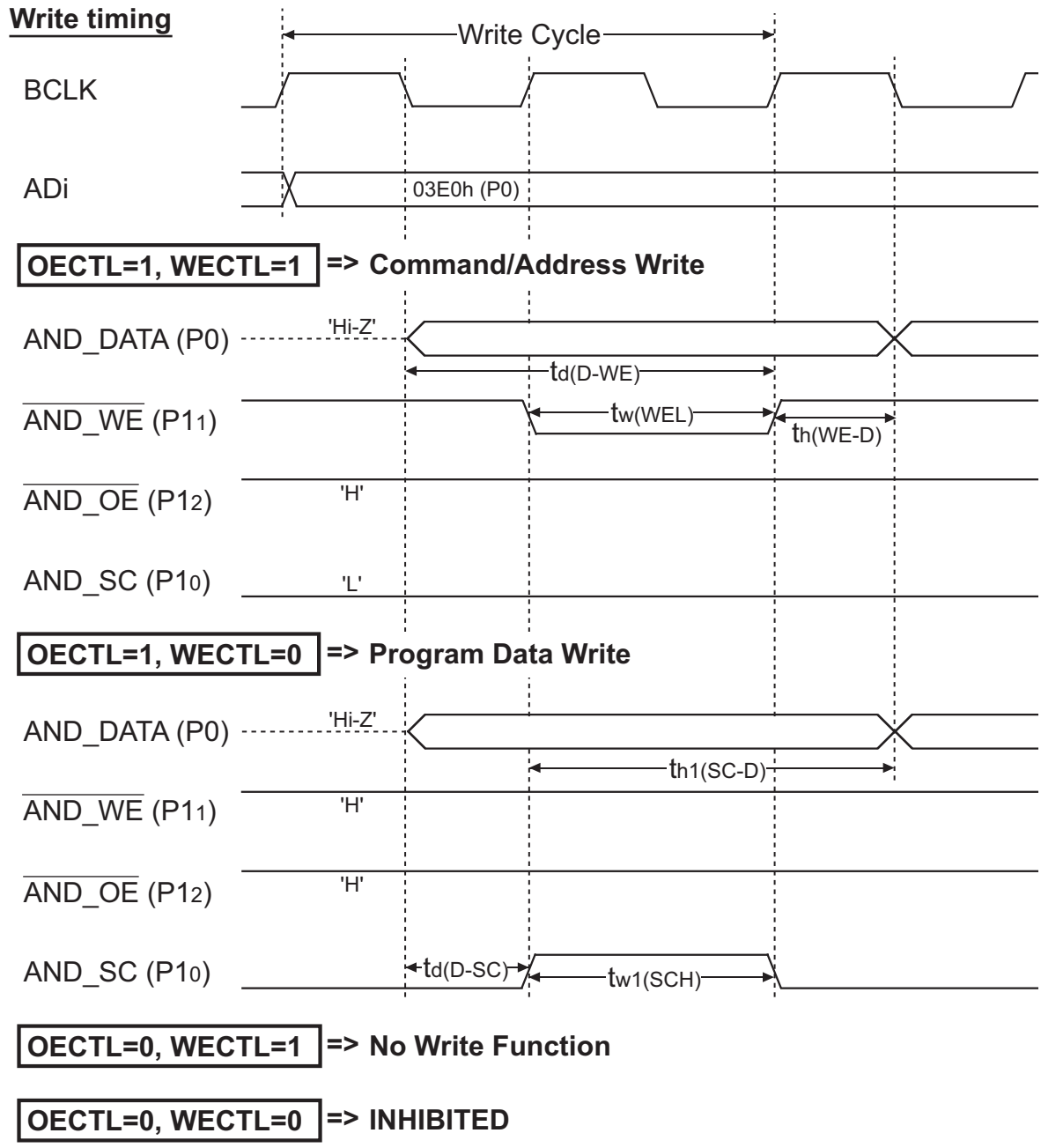



**Figure 1.206. AND Flash Control write timing diagram**



Downloaded from Arrow.com.

# Usage Notes

## Timer A

### Timer mode

The value of the counter can be read, with arbitrary timing, by reading the Timer Ai register while a count is in progress.
Reading the Timer Ai register with the reload timing gets “FFFF16”.
After setting a value in the Timer Ai register, a proper value can be read with the counter stopped before it starts counting.

### Event counter mode

The value of the counter can be read, with arbitrary timing, by reading the Timer Ai register while a count is in progress.
Reading the Timer Ai register with the reload timing gets “FFFF16” by underflow or “000016” by overflow.
After setting a value in the Timer Ai register, a proper value can be read with the counter stopped before it starts counting .
Reset the timer when counting has stopped in free run type.
If using “Free-Run type”, the timer register contents may be unknown when counting begins. Set the timer value immediately after counting has started.
Example if the up/down count is not switched:
• Enable the “Reload” function and write to the timer register before counting begins.
• Rewrite the value to the timer register immediately after counting has started.
• If counting up, rewrite “000016” to the timer register.
• If counting down, rewrite “FFFF1” to the timer register. This will cause the same operation as “Free-Run type”.
Example if the up/down count is switched:
• Use the “Reload type” operation until the first count pulse is input.
• Switch to “Free-Run type”.

### One-shot timer mode

Setting the count start flag to “0” while a count is in progress causes as the following:
• The counter stops counting and a content of reload register is reloaded.
• The TAiOUT pin outputs “L” level.
• The interrupt request generated and the Timer Ai interrupt request bit goes to “1”.
The output from the one-shot timer synchronizes with the count source generated internally. Therefore, when an external trigger has been selected, a delay of one cycle of count source (maximum) occurs between the trigger input to the TAiIN pin and the one-shot timer output.
The Timer Ai interrupt request bit goes to “1” if the timer's operation mode is set using any of the following procedures:
• Selecting one-shot timer mode after reset.
• Changing operation mode from timer mode to one-shot timer mode.
• Changing operation mode from event counter mode to one-shot timer mode.
Therefore, to use Timer Ai interrupt (interrupt request bit), set Timer Ai interrupt request bit to “0” after the above listed changes have been made.
If a trigger occurs while a count is in progress, after the counter performs one down count following the reoccurrence of a trigger, the reload register contents are reloaded, and the count continues.
To generate a trigger while a count is in progress, generate the second trigger after a period longer than one cycle of the timer's count source after the previous trigger occurred.



Downloaded from Arrow.com.

**Pulse modulation mode**

The Timer Ai interrupt request bit becomes “1” if setting operation mode of the timer in compliance with any of the following procedures:

• Selecting PWM mode after reset.

• Changing operation mode from timer mode to PWM mode.

• Changing operation mode from event counter mode to PWM mode.

Therefore, to use Timer Ai interrupt (interrupt request bit), set Timer Ai interrupt request bit to “0” after the above listed changes have been made.

Setting the count start flag to “0” while PWM pulses are being output causes the counter to stop counting. If the TAiOUT pin is outputting an “H” level in this instance, the output level goes to “L”, and the Timer Ai interrupt request bit goes to “1”. If the TAiOUT pin is outputting an “L” level in this instance, the level does not change, and the Timer Ai interrupt request bit does not becomes “1”.

## A/D converter

• Write to each bit (except bit 6) of AD control register 0, AD control register 1, and to bit 0 of AD control register 2 when A/D conversion is stopped (before a trigger occurs). When the VREF connection bit is changed from "0" to “1”, wait 1 µs or longer before starting A/D conversion.

• When changing A/D operation mode, select the analog input pin again.

• Using one-shot mode or single sweep mode:
Read the corresponding AD register after confirming A/D conversion is finished. (Check the A/D conversion interrupt request bit.)

• Using repeat mode, repeat sweep mode 0 or repeat sweep mode 1:
Use the undivided main clock as the internal CPU clock.
When f(Xin) is faster than 10MHz, make the A/D frequency 10MHz or less by dividing.

## Serial I/O (UART Mode)

Description When the CLKi and CTSi pin level goes to “H” (Note 1), if the UiMR register is set to either of the following settings, the UiERE bit of the UiC1 register is set to “1” (parity error signal output enabled). When the PRYE bit of the UiMR register is set to “1” while the UiERE bit is “1” (parity error signal output enabled), if a parity error occurs at receiving data, the TXDi pin outputs the “L” level. To prevent this, set the UiERE bit after setting the UiMR register.

- Set bits SMD2 through SMD0 to “$000_2$” (serial I/O disabled) through “$101_2$” (UART mode transfer data 8 bits long)
- Set bits SMD2 through SMD0 to “$001_2$” (clock synchronous serial I/O mode) through “$100_2$” (UART mode transfer data 7 bits long)
- Set bits SMD2 through SMD0 to “$001_2$” (clock synchronous serial I/O mode) through “$101_2$” (UART mode transfer data 8 bits long)
- Set bits SMD2 through SMD0 to “$001_2$” (clock synchronous serial I/O mode) transfer data 9 bits long)
- Set bits SMD2 through SMD0 to “$010_2$” (I2C mode) through “$101_2$” (UART mode transfer data 8 bits long)

Note 1: If the pins are not used as CLKi or CTSi, these conditions apply when the pin level goes to “H”.



Downloaded from Arrow.com.

**DMA**

(1) Additional description of the DMA enable bit

Bit 3 of the DMA0 and DMA1 control registers is assigned as the DMA enable bit. Setting the DMA enable bit to “1” makes DMA active. If data transfer starts immediately after the DMA becomes active, the DMAC performs the following operations.

(a) The value of either the source pointer or the destination pointer, whichever is set to the forward direction, is reloaded to the forward direction address pointer.

(b) The value of the transfer counter register is reloaded to the transfer counter.

Thus, writing “1” to the DMA enable bit when DMA is active causes the above operations to be carried out, and the DMAC operates again from the initial state at that point.

(2) Additional description of the DMA request bit

Bit 2 of the DMA0 and DMA1 control registers is assigned as the DMA request bit. The DMA request bit is set to “1” if a DMA transfer request signal occurs even if DMA is not active. Also, changing the DMA transfer request cause select bits may set the DMA request bit to “1”. Make sure to set the DMA request bit to “0” after changing the DMA request cause select bits.

The DMA request bit is set to “1” if a DMA transfer request signal occurs and is set to “0” immediately after data transfer starts. If DMA is active, data transfer starts immediately, so the value of the DMA request bit, if read by software, will be “0” in most cases. To determine whether DMA is active, read the DMA enable bit. Figure 1.207 shows the setting routine for the DMA-related registers.

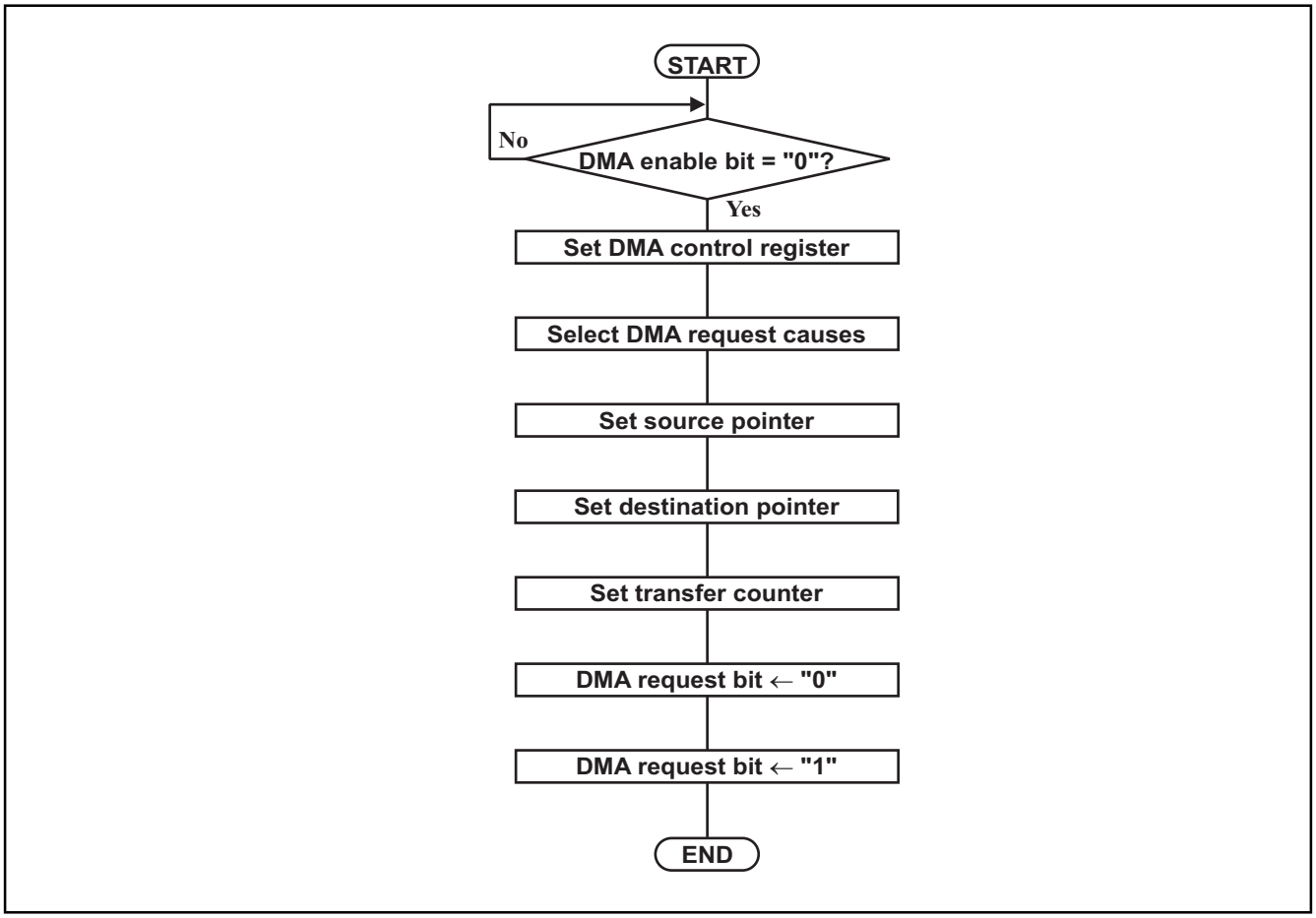



**Figure 1.207. Setting routine of DMA control registers**



Downloaded from Arrow.com.

(3) Writing to the DMAE bit in DMiCON register

If the following conditions are met:

The DMAE bit is set to “1” again while it is already set to “1” (DMAi is in active state).

A DMA request may occur simultaneously when the DMAE bit is being written.

Follow the steps below:

Step 1: Write “1” to the DMAE bit and DMAS bit in DMiCON register simultaneously (Note 1).

Step 2: Make sure that the DMAi is in an initial state (Note 2) in a program.

If the DMAi is not in an initial state, the above steps should be repeated.

Note 1: The DMAS bit remains unchanged even if “1” is written. However, if “0” is written to this bit, it is set to “0” (DMA not requested). In order to prevent the DMAS bit from being modified to “0”, “1” should be written to the DMAS bit when “1” is written to the DMAE bit. In this way the state of the DMAS bit immediately before being written can be maintained. Similarly, when writing to the DMAE bit with a read-modify-write instruction, “1” should be written to the DMAS bit in order to maintain a DMA request which is generated during execution.

Note 2: Read the TCRi register to verify whether the DMAi is in an initial state. If the read value is equal to a value which was written to the TCRi register before DMA transfer start, the DMAi is in an initial state. (If a DMA request occurs after writing to the DMAE bit, the value written to the TCRi register is “1”.) If the read value is a value in the middle of a transfer, the DMAi is not in an initial state.



Downloaded from Arrow.com.

### Stop Mode and Wait Mode

(1) When returning from stop mode by hardware reset, $\overline{\text{RESET}}$ pin must be set to “L” level until main clock oscillation is stabilized.
(2) When switching to either wait mode or stop mode, instructions occupying four bytes either from the WAIT instruction or from the instruction that sets the all clock stop control bit to “1” within the instruction queue are prefetched and then the program stops. So put at least four NOPs in succession either to the WAIT instruction or to the instruction that sets the all clock stop control bit to “1”.
(3) When using low-speed mode and low power dissipation mode, set the WAIT peripheral function clock stop bit (CM02) to “1” and do not shift to wait mode.
(4) When using $f_{SYN}$ as the internal system clock, change to $f(X_{IN})$ before entering to stop mode (set bit 0 of the frequency synthesizer control register to “0”).

### Interrupts

#### Reading address $00000_{16}$

When maskable interrupt occurs, the CPU reads the interrupt information (the interrupt number and interrupt request evel) in the interrupt sequence. The interrupt request bit of the interrupt written in address $00000_{16}$ will then be set to “0”.
Do not read address $00000_{16}$ by software. Reading address $00000_{16}$ by software sets enabled highest priority interrupt source request bit to “0”. Though the interrupt is generated, the interrupt routine may not be executed.

#### Setting the stack pointer

The value of the stack pointer immediately after reset is initialized to $0000_{16}$. Accepting an interrupt before setting a value in the stack pointer may cause program runaway. Be sure to set a value in the stack pointer before accepting an interrupt.
When using the $\overline{\text{NMI}}$ interrupt, initialize the stack pointer at the beginning of a program. Generating any interrupts including the $\overline{\text{NMI}}$ interrupt is prohibited for the first instruction immediately after reset.

### The $\overline{\text{NMI}}$ interrupt

The $\overline{\text{NMI}}$ interrupt can not be disabled. Be sure to connect $\overline{\text{NMI}}$ pin to Vcc with a pull-up resistor if unused. Do not go into stop mode when the $\overline{\text{NMI}}$ pin set to “L”.
The $\overline{\text{NMI}}$ pin also serves as $P8_5$, which is exclusively an input. Reading the contents of the P8 register allows the pin value to be read. Reading this pin is only to be used for establishing the pin level when the $\overline{\text{NMI}}$ interrupt is input.
Do not reset the CPU with the input to the $\overline{\text{NMI}}$ pin in the “L” state.
Do not attempt to go into stop mode when the input to the $\overline{\text{NMI}}$ pin is in “L” state. When the input to the $\overline{\text{NMI}}$ is in “L” state, CM10 is fixed to “0” thereby refusing to go into stop mode.
Do not attempt to go into wait mode when the input to the $\overline{\text{NMI}}$ pin is in “L” state. When the input to the $\overline{\text{NMI}}$ pin is in “L” state, the CPU stops but the oscillation does not. This action does not save power. When this occurs, the CPU is returned to the normal state by a later interrupt.
Signals input to the $\overline{\text{NMI}}$ pin require an “L” level of (2 clocks + 300nS) or more from the operation clock of the CPU.

### External interrupt

Either an “H” or “L” level of at least 250 ns width is necessary for the signal input to pins $\overline{\text{INT0}}$ to $\overline{\text{INT2}}$ regardless of the CPU operation clock.
When the polarity of the $\overline{\text{INT0}}$ to $\overline{\text{INT2}}$ pins is changed, the interrupt request bit is sometimes set to “1”. After changing the polarity, reset the interrupt request bit to “0”. Figure 1.208 shows the procedure for changing the INT interrupt generate factor.



Downloaded from Arrow.com.

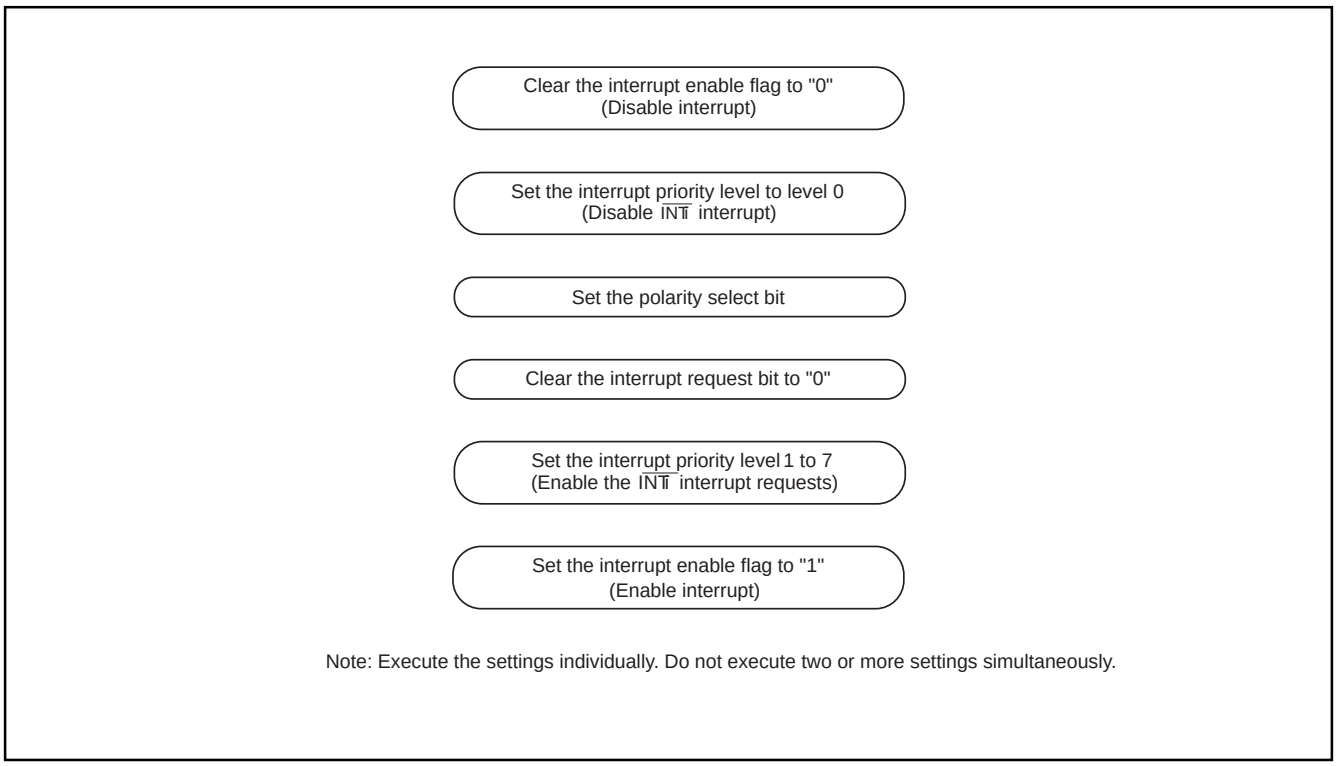



**Figure 1.208. Switching condition of INT interrupt request**



Downloaded from Arrow.com.

### Clearing the Interrupt request bit

Even when the IR bit (bit 3 of the interrupt control register) is cleared to "0" (interrupt not requested), it may not actually get cleared to "0" depending on the instruction used to clear it. Therefore, use the MOV instruction to clear the IR bit.

Rewriting the interrupt control register

Rewrite the interrupt control register so that it does not generate an interrupt request for that register. If an interrupt request occurs, rewrite the interrupt control register after the interrupt is disabled. Some program examples are described below.

When an instruction to rewrite the interrupt control register is executed but the interrupt is disabled, the interrupt request bit is not always set even if the interrupt request for that register has been generated. This will depend on the instruction. If this creates problems, use the instructions below to change the register.

Instructions: AND, OR, BCLR, BSET

Examples 1 through 3 show how to prevent the I flag from being set to "1" (interrupts enabled) before the interrupt control register is rewritting, due to the effects of the internal bus and the instruction queue buffer.

**Example 1:**

```
INT_SWITCH1:
        FCLR    I           :Disable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.
        NOP                 ;Four NOP instructions are required when using the HOLD function.
        NOP
        FSET    I           ;Enable interrupts.
```

**Example 2:**

```
INT_SWITCH2:
        FCLR    I           :Disable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.
        MOV.W   MEM, R0     ;Dummy read.
        FSET    I           ;Enable interrupts.
```

**Example 3:**

```
INT_SWITCH3:
        PUSHC   FLG         ;Push Flag register onto stack
        FCLR    I           ;Diable interrupts.
        AND.B   #00h, 0054h ;Clear TA0IC int. priority level and int. request bit.‘
        POPC    FLG         ;Enable interrupts.
```

**The reason why two NOP instructions (four using the HOLD function) or a dummy read is inserted before "FSET I" in Examples 1 and 2, is to prevent the interrupt enable flag I from being set before the interrupt control register is rewritten due to the effects of the instruction queue.**

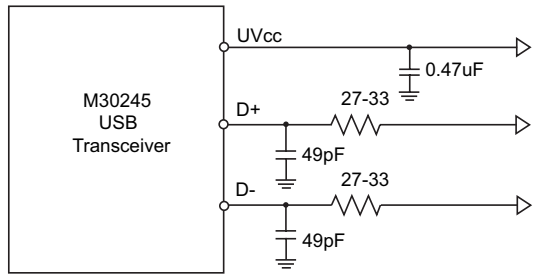


Downloaded from Arrow.com.

# Applications

## USB Transceiver

In order to meet the impedance matching requirements of the USB Specification, a 27Ω-33Ω resistor must be added to USB D+ (pin 4) and to USB D- (pin 5). In addition, capacitors connected between USB D+ and USB D- and Vss may need to be added for rise/fall time matching and edge control. These capacitors, if necessary, can be placed between the mcu and the 27-33Ω resistors. A coupling capactior may also be placed between D+ and D-. Their configuration and values will depend on the PCBs layout. Perform the approriate USB testing to determine the correct component placement. An example placement of external components is shown in Figure 1.209.




Note 1: Capacitor and resistor values and their configuration depend on PCB layout.
Note 2: Connecting any type of choke coil to D+ or D- is not recomended.

**Figure 1.209. Example configuration of External USB components**

## Attach/Detach Function

The Attach/Detach Function can be used to attach or detach a USB device from the host without physically disconnecting the USB cable. When attaching a USB device, the attach/detach register should be set to $03_{16}$ at the same time or before the USB Enable bit is set. Similarly, when detaching the device from the host , the attach/detach register should be set to $01_{16}$ when disabling the USB block.

If you do not set the Attach/Detach bit (bit 1 at address $001F_{16}$) to HIGH, the system will default to its normal mode.

D+ is connected to $P9_0$/ATTACH through a 1.5 K resistor in compliance with the USB specification.

Note: If the D+ pin is connected to UVcc, this mode will not work.

Hardware connections are shown in Figure 1.210.

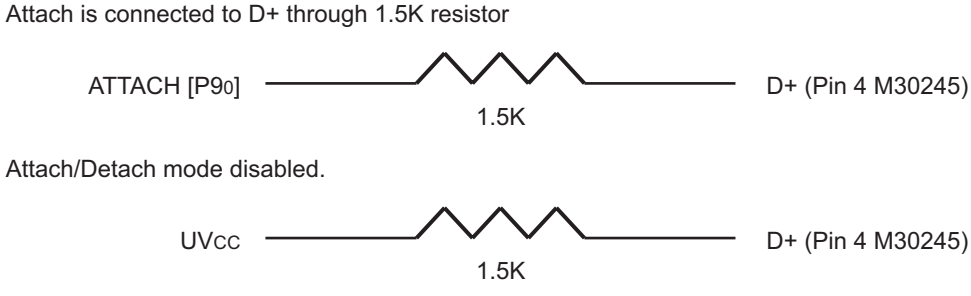



**Figure 1.210. Attach/Detach function connections**



Downloaded from Arrow.com.

## Programming Notes

### USB

The following Programming Notes should be incorporated into user code, to ensure strict adherence to the USB protocol for Control Transfers.

(1) In applications requiring high-reliability, we recommend providing the system with protective measures, such as USB function initialization by software or USB reset by the host, to prevent USB communication from being terminated unexpectedly, for example due to external causes such as noise.

(2) USB2.0 specification stipulates a driver impedance 28 to 44Ω (see 7.1.1.1 Full-speed (12Mb/s) Driver Characteristics). Connect a serial resistor (recommended value: 27 to 33Ω) to the USB D+ pin and the USB D- pin to satisfy this specification. Also connect, if required, a capacitor between the USB D+ pin or USB D-pin and the Vss pin. These capacitors are to control ringing or adjust the rise and fall times and the crossover point of D+/D-. The numerical values and configuration of the peripheral components need to be adjusted according to differences in the characteristic impedance and layout of the printed circuit board. on which they are mounted. Therefore, perform careful evaluation of the system in use and observe the waveforms before deciding on connection or disconnection and adjusting the values of the resistors and capacitors.

(3) Do not connect the D+ pin or the D- pin to a choke coil.

(4) If the USB Attach/Detach function will not be used, connect the UVcc pin and the USB D+ pin via a 1.5 kΩ resistor. (The D+ line pull-up timing depends on the UVcc pin.) If the USB Attach/Detach function is used, connect the P9₀/ATTACH pin and the USB D+ pin via a 1.5 kΩ resistor. Regardless of whether or not the USB Attach/Detach function is used, connect the UVcc pin to the power supply. In addition, the time required for the host PC to recognize the USB Attach/Detach state will vary depending state of the system as a whole, including board resistance and capacitance components, USB cable capacitance, and the board characteristics and processing speed of the host. Perform careful evaluation of the system in use.

(5) The interrupt service routine (ISR) associated with those USB Function interrupts that are caused by errors must have execution priority over the ISR for EP0 interrupts. Upon receipt of a USB Function interrupt, the following actions should be taken:

*Step #1:* From the USB Interrupt Status (USBIS) and the USB Endpoint 0 Control & Status (EP0CS) registers, determine if the 'Error Interrupt Status Flag' & the SETUP_END flag (i.e., INTST8 & EP0CSR5, respectively) are both set.
[YES] => Set CLR_SETUP_END (EP0CSR11). Go to Step #2.
[NO] => No special S/W action required. Go to Step #1 after the next USB Function interrupt.

*Step #2:* Is EP0 IN FIFO loading in progress - i.e., data has been written to EP0 IN FIFO, but SET_IN_BUF_RDY (EP0CSR7) is not yet set?
[YES] => Set SET_IN_BUF_RDY (EP0CSR7). Go to Step #3 after the next EP0 interrupt.
[NO] => No special S/W action required. Go to Step #1 after the next USB Function interrupt.

*Step #3:* Are OUT_BUF_RDY & SETUP (i.e., EP0CSR0 & EP0CSR2, respectively) set?
[YES] => Go to Step #4.
[NO] => Go to Step #3 after the next EP0 interrupt.

*Step #4:* Does the current Control Transfer Setup stage DATA0 packet identify a Control Read Transfer?
[YES] => Complete loading EP0 IN FIFO. Set CLR_OUT_BUF_RDY, SET_IN_BUF_RDY, & CLR_SETUP (i.e., EP0CSR6, EP0CSR7, & EP0CSR8, respectively). Go to Step #1 after the next USB Function interrupt.
[NO] => Go to Step #3 after the next EP0 interrupt.

Refer to the flowchart in Figure 1.211 for more information on this programming note.



Downloaded from Arrow.com.

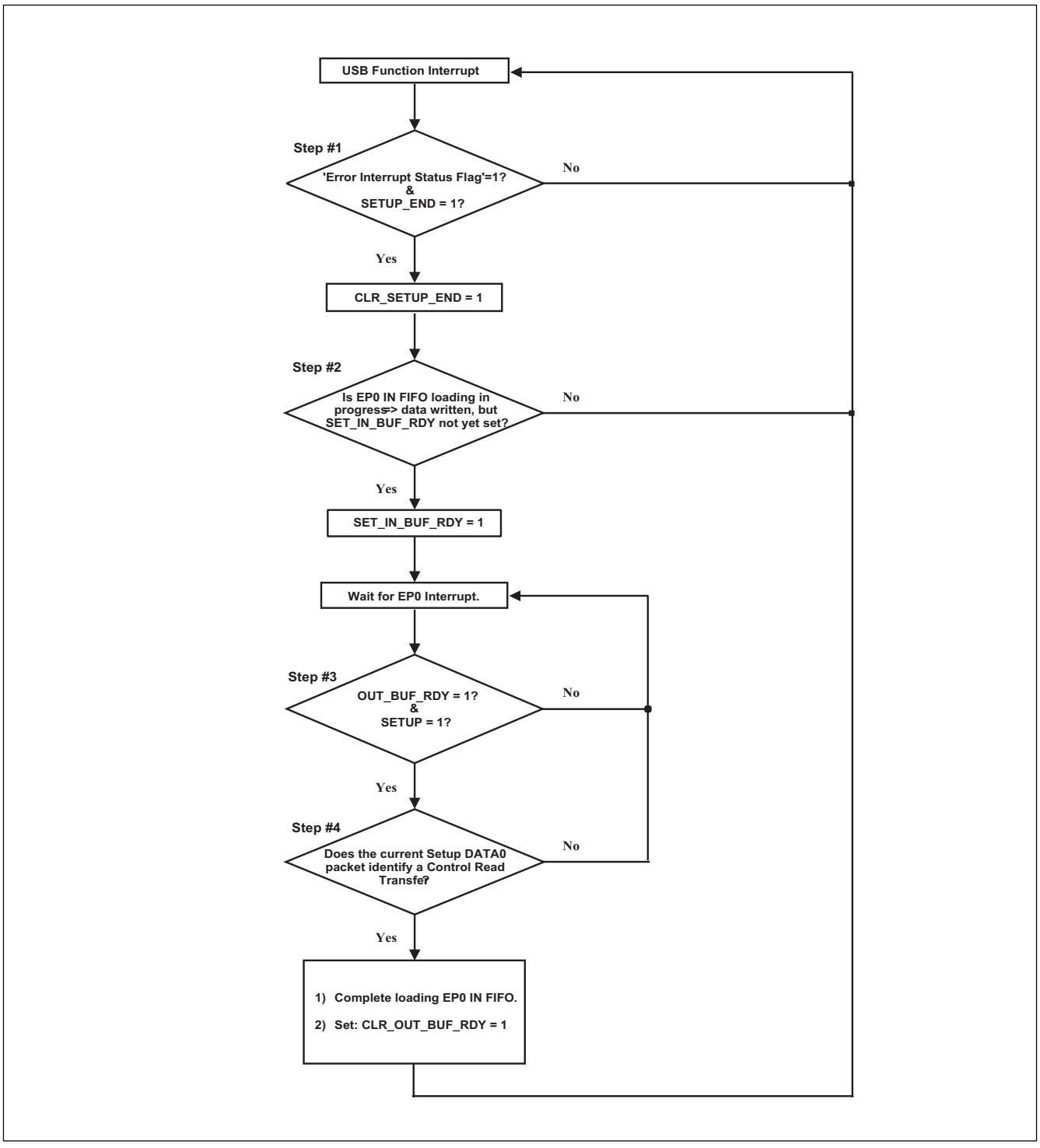



**Figure 1.211. USB Programming Note 1 Flowchart**



Downloaded from Arrow.com.

(6) Additional actions to take upon receipt of an EP0 interrupt are as follows (Refer to the flowchart in Figure 1.212):

*Step #1:* Is OUT_BUF_RDY (EP0CSR0) set?
[YES] => Go to Step #2.
[NO] => No special S/W action required. Go to Step #1 after the next EP0 interrupt.
*Step #2:* Is SETUP_END (EP0CSR5) set?
[YES] => Set CLR_OUT_BUF_RDY, CLR_SETUP_END, & SEND_STALL (i.e., EP0CSR6, EP0CSR11, & EP0CSR12, respectively). [Also set CLR_SETUP, if SETUP flag == '1'.] Go to Step #1 after the next EP0 interrupt.
[NO] => Go to Step #3.
*Step #3*: Read number of data bytes equal to the EP0 'Receive Byte Count', stored in EP0WC7-0, from EP0 OUT FIFO. Is this the final DATA packet of a Control Write Transfer?
[YES] => Go to Step #4_0.
[NO] => Go to Step #5_0.
*Step #4_0:* Is SETUP_END (EP0CSR5) set?
[YES] => Set CLR_OUT_BUF_RDY, CLR_SETUP_END, & SEND_STALL (i.e., EP0CSR6, EP0CSR11, & EP0CSR12, respectively). [Also set CLR_SETUP, if SETUP flag == '1'.] Go to Step #1 after the next EP0 interrupt.
[NO] => Set CLR_OUT_BUF_RDY & SET_DATA_END (i.e., EP0CSR6 & EP0CSR9, respectively). [Also set CLR_SETUP, if SETUP flag == '1'.] Go to Step #4_1.
*Step #4_1:* Is SETUP_END (EP0CSR5) set?
[YES] => Set SEND_STALL (EP0CSR12). Go to Step #6_0.
[NO] => Go to Step #1 after the next EP0 interrupt.
*Step #5_0:* Is SETUP_END (EP0CSR5) set?
[YES] => Set CLR_OUT_BUF_RDY, CLR_SETUP_END, & SEND_STALL (i.e., EP0CSR6, EP0CSR11, & EP0CSR12, respectively). [Also set CLR_SETUP, if SETUP flag == '1'.] Go to Step #1 after the next EP0 interrupt.
[NO] => Set CLR_OUT_BUF_RDY (i.e., EP0CSR6). [Also set CLR_SETUP, if SETUP flag == '1'.] Go to Step #5_1.
Step #5_1: Is SETUP_END (EP0CSR5) set?
[YES] => Set SEND_STALL (EP0CSR12). Go to Step #6_0.
[NO] => Go to Step #1 after the next EP0 interrupt.
*Step #6_0:* Are OUT_BUF_RDY & SETUP (EP0CSR0 & EP0CSR2) set?
[YES] => Go to Step #6_1.
[NO] => Go to Step #6_0 after the next EP0 interrupt.
Step #6_1: Is SETUP_END (EP0CSR5) set?
[YES] => Set CLR_OUT_BUF_RDY, CLR_SETUP, & CLR_SETUP_END (EP0CSR6, EP0CSR8 & EP0CSR11). Go to Step #6_0 after the next EP0 interrupt.
[NO] => Set CLR_OUT_BUF_RDY & CLR_SETUP (EP0CSR6 & EP0CSR8), and clear SEND_STALL (EP0CSR12). Go to Step #1 after the next EP0 interrupt.

(7) Writing to the USB Function Interrupt Clear Register (USBIC).

Writing to the USB Function Interrupt Clear Register (USBIC) to clear USB Function Interrupt Status bits requires special consideration. Before performing this operation, the USB Function Interrupt Enable Register (USBIE) should be cleared (i.e., all bits disabled). Upon completion of the write to USBIC, the value of USBIE just prior to its clearing should be restored.



Downloaded from Arrow.com.

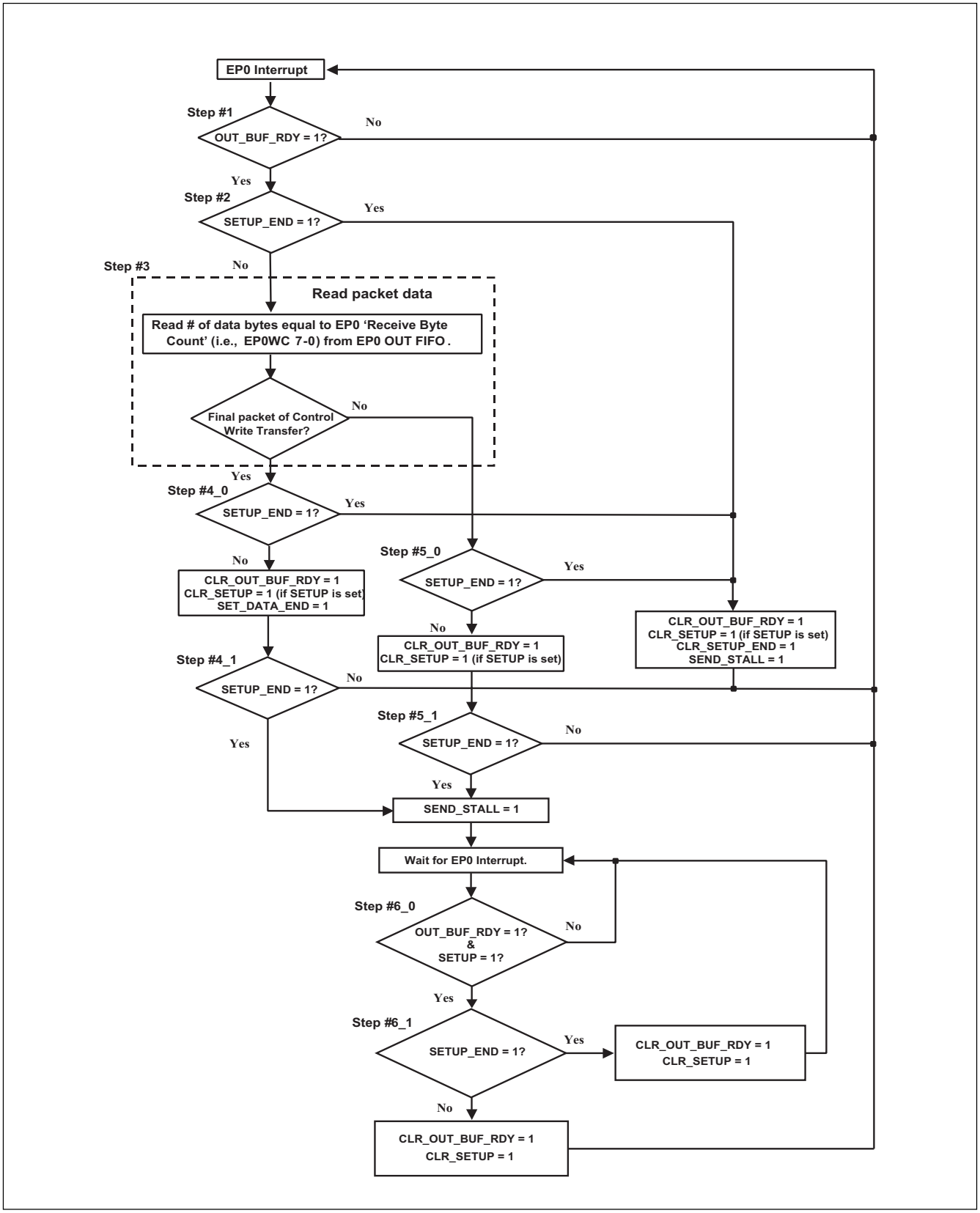



**Figure 1.212. USB Programming Note 2 Flowchart**



Downloaded from Arrow.com.

### Using $\overline{\text{HOLD}}$ Signal

When $\overline{\text{HOLD}}$ input is used, set P4$_0$ to P4$_7$ and P5$_0$ to P5$_2$ as input before the CPU shifts from single-chip mode to microprocessor mode or memory expansion mode.

### Decreasing Power Consumption

When A/D conversion is not carried out, select not to connect $V_{REF}$ using the $V_{REF}$ connect bit in AD control register 1. To carry out A/D conversion, start the conversion 1µs or longer after connecting $V_{REF}$.

### Microprocessor Mode and Shifting from Microprocessor Mode to Memory Expansion Mode or Single-chip Mode

In microprocessor mode, the SFR, internal RAM and external memory space can be accessed. Therefore, the internal ROM area cannot be accessed.

If microprocessor mode is set (“H” is applied to the $CNV_{SS}$ pin) when coming out of a reset, the internal ROM cannot be accessed even if the CPU shifts to memory expansion mode or single-chip mode.

### Resetting when "H" is applied to CNVss pin

If the microprocessor is reset when “H” is applied to the CNVss pin, the internal ROM cannot be read.

### Noise

Connect a bypass capacitor (at least 0.1 µF) using the shortest and thickest wire possible.

### Input-only Pins

If different power supplies are provided to the system, as shown in Figure 1.213 Circuit example, and the voltage of an unused input-only pin is higher than Vcc, do not directly connect the dedicated input pin to the power supply. As in the circuit example indicated by the arrow, connect the input-only pin to the power supply via a resistor rated at approximately 1 kΩ. The above applies even if the power rise time is different at power-on.

If the voltage of the input pin voltage is higher than Vcc, latch up could occur.

*: A resistor is not required when using a Vcc voltage equal to or higher than the voltage of the dedicated input pin.

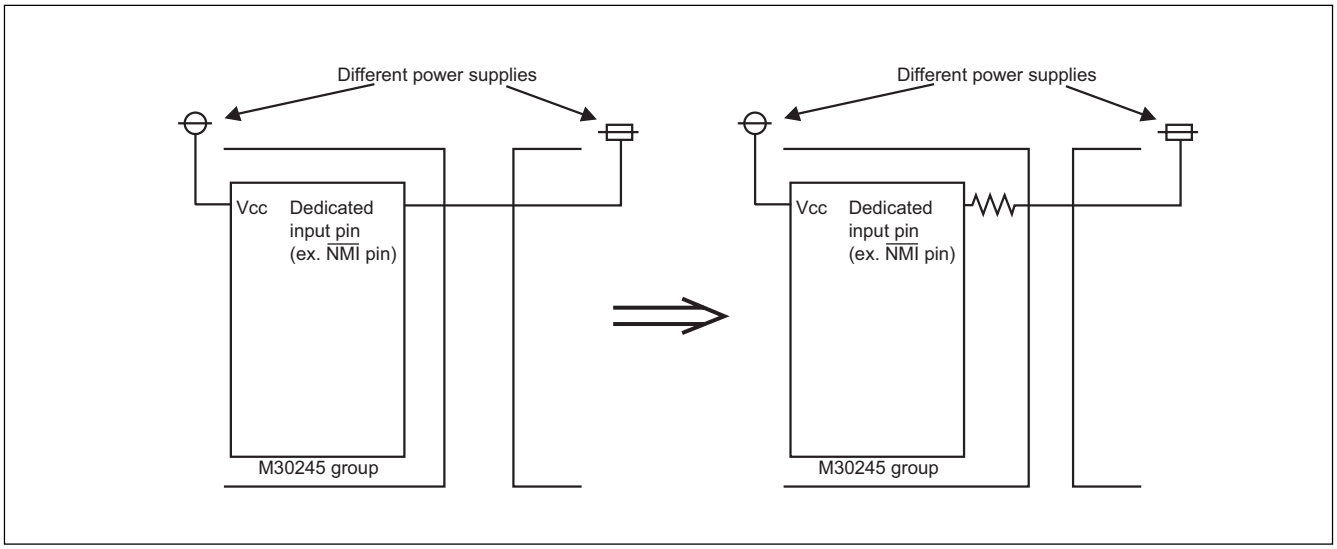



**Figure 1.213. Circuit example**



Downloaded from Arrow.com.

### Electric Characteristic Differences Between Mask ROM and Flash Memory Version MCUs

There are differences in electric characteristics, operation margin, noise immunity, and noise radiation between Mask ROM and Flash Memory version MCUs due to the difference in the manufac-turing processes.
When manufacturing an application system with the Flash Memory version and then switching to use of the Mask ROM ver-sion, please perform sufficient evaluations for the commercial samples of the Mask ROM version.

### Mask ROM Version

Do not write to the internal ROM area in the Mask ROM version.

### ROM ORDERING METHOD

1.Mask ROM Order Confirmation Form
2.Mark Specification Form
3.Data to be written to ROM, in one floppy disk.

* For the mask ROM confirmation and the mark specifications, refer to the “Renesas Technology Corp.” Homepage (http://www.renesas.com).

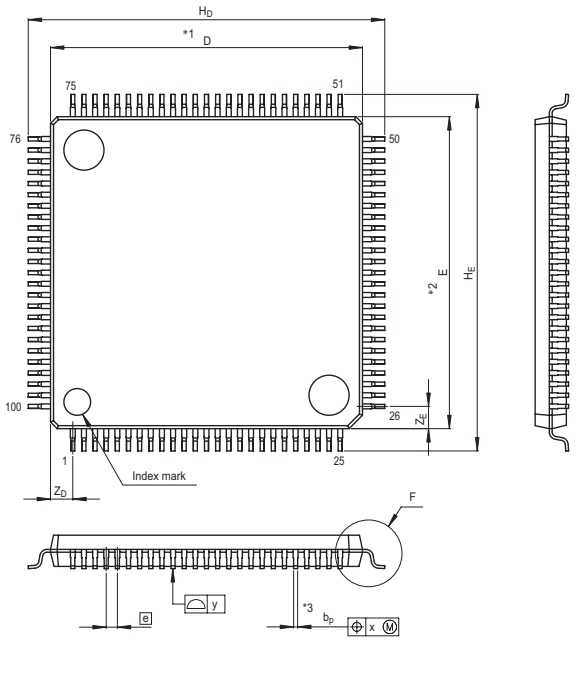


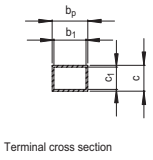
Downloaded from Arrow.com.

## Package Outline

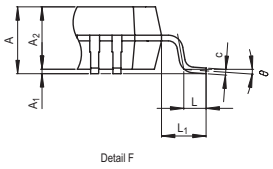
# PLQP0100KB-A

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LQFP100-14x14-0.50 | PLQP0100KB-A | 100P6Q-A / FP-100U / FP-100UV | 0.6g |




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 | 13.9 | 14.0 | 14.1 |
| E | 13.9 | 14.0 | 14.1 |
| $A_2$ | — | 1.4 | — |
| $H_D$ | 15.8 | 16.0 | 16.2 |
| $H_E$ | 15.8 | 16.0 | 16.2 |
| A | — | — | 1.7 |
| $A_1$ | 0.05 | 0.1 | 0.15 |
| $b_p$ | 0.15 | 0.20 | 0.25 |
| $b_1$ | — | 0.18 | — |
| c | 0.09 | 0.145 | 0.20 |
| $c_1$ | — | 0.125 | — |
| θ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.08 |
| y | — | — | 0.08 |
| $Z_D$ | — | 1.0 | — |
| $Z_E$ | — | 1.0 | — |
| L | 0.35 | 0.5 | 0.65 |
| $L_1$ | — | 1.0 | — |



Downloaded from Arrow.com.

# REVISION HISTORY — M30245 Group Datasheet

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jul 20, 2004 | – | First Edition issued |
| 2.00 | Oct 16, 2006 | All pages | Package names “PLQP0100KB-A” → “100P6Q-A” revised |
| | | 16 | Figure 1.8 revised |
| | | 55 | Modifying the interrupt control registers revised |
| | | 145 | I2C Bus interface mode “To use the I2C bus, .... the SDAi to output.” → “In $I^2C$ master mode, .... of the direction register” |
| | | 146 | Figure 1.106 Note revised |
| | | 147 | UARTi Special Mode Register (UiSMR) “Port (SCLi) is .... of the port direction register.” → “In $I^2C$ master mode, .... of the port direction register.” |
| | | 165 | Precautions “• For flash memory version .... by a receiver.” added |
| | | 250-256 | Usage Notes added |
| | | 258 | Programming Notes; USB (1) to (4) added |
| | | 262 | Using HOLD Signal, Decreasing Power Consumption, Microprocessor Mode and Shifting from Microprocessor Mode to Memory Expansion Mode or Singlechip Mode, Resetting when “H” is applied to CNVss pin, Noise, Input-only Pins added |
| | | 263 | Electric Characteristic Differences Between Mask ROM and Flash Memory Version MCUs, Mask ROM Version, ROM ORDERING METHOD added |
| | | 264 | Package Outline added |
| | | | |



Downloaded from Arrow.com.

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

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

## RENESAS SALES OFFICES

http://www.renesas.com

Refer to "**http://www.renesas.com/en/network**" for the latest and detailed information.

**Renesas Technology America, Inc.**
450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

**Renesas Technology Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

**Renesas Technology (Shanghai) Co., Ltd.**
Unit 204, 205, AZIACenter, No.1233 Lujiazui Ring Rd, Pudong District, Shanghai, China 200120
Tel: <86> (21) 5877-1818, Fax: <86> (21) 6887-7898

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

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

**Renesas Technology Singapore Pte. Ltd.**
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

**Renesas Technology Korea Co., Ltd.**
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> (2) 796-3115, Fax: <82> (2) 796-2145

**Renesas Technology Malaysia Sdn. Bhd**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jalan Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

© 2006. Renesas Technology Corp., All rights reserved. Printed in Japan.

Colophon .7.0

Downloaded from Arrow.com.