

Features

- Utilizes the ARM7TDMI™ ARM® Thumb® Processor Core
 - High-performance 32-bit RISC Architecture
 - High-density 16-bit Instruction Set
 - Leader in MIPS/Watt
 - Embedded ICE (In-circuit Emulation)
- 8K Bytes Internal SRAM
- Fully Programmable External Bus Interface (EBI)
 - Maximum External Address Space of 64M Bytes
 - Up to 8 Chip Selects
 - Software Programmable 8/16-bit External Data Bus
- 8-channel Peripheral Data Controller
- 8-level Priority, Individually Maskable, Vectored Interrupt Controller
 - 5 External Interrupts, Including a High-priority, Low-latency Interrupt Request
- 54 Programmable I/O Lines
- 6-channel 16-bit Timer/Counter
 - 6 External Clock Inputs, 2 Multi-purpose I/O Pins per Channel
- 2 USARTs
 - 2 Dedicated Peripheral Data Controller (PDC) Channels per USART
 - Support for up to 9-bit Data Transfers
- 2 Master/Slave SPI Interfaces
 - 2 Dedicated Peripheral Data Controller (PDC) Channels per SPI
 - 8- to 16-bit Programmable Data Length
 - 4 External Slave Chip Selects per SPI
- 3 System Timers
 - Period Interval Timer (PIT); Real-time Timer (RTT); Watchdog Timer (WDT)
- Power Management Controller (PMC)
 - CPU and Peripherals Can be Deactivated Individually
- Clock Generator with 32.768 kHz Low-power Oscillator and PLL
 - Support for 38.4 kHz Crystals
 - Software Programmable System Clock (up to 33 MHz)
- IEEE 1149.1 JTAG Boundary Scan on All Active Pins
- Fully Static Operation: 0 Hz to 33 MHz, Internal Frequency Range at $V_{DDCORE} = 3.0V$, 85°C
- 2.7V to 3.6V Core and PLL Operating Voltage Range; 2.7V to 5.5V I/O Operating Voltage Range
- -40°C to +85°C Temperature Range
- Available in a 144-lead TQFP Package and in 144-ball BGA Package

Description

The AT91M42800A is a member of the Atmel AT91 16/32-bit microcontroller family, which is based on the ARM7TDMI processor core. This processor has a high-performance 32-bit RISC architecture with a high-density 16-bit instruction set and very low power consumption. In addition, a large number of internally banked registers result in very fast exception handling, making the device ideal for real-time control applications. The AT91 ARM-based MCU family also features Atmel's high-density, in-system programmable, nonvolatile memory technology. The AT91M42800A has a direct connection to off-chip memory, including Flash, through the External Bus Interface.

The Power Management Controller allows the user to adjust device activity according to system requirements, and, with the 32.768 kHz low-power oscillator, enables the AT91M42800A to reduce power requirements to an absolute minimum. The AT91M42800A is manufactured using Atmel's high-density CMOS technology. By combining the ARM7TDMI processor core with on-chip SRAM and a wide range of peripheral functions including timers, serial communication controllers and a versatile clock generator on a monolithic chip, the AT91M42800A provides a highly flexible and cost-effective solution to many compute-intensive applications.



AT91 ARM® Thumb® Microcontrollers

AT91M42800A



Pin Configuration

Figure 1. Pin Configuration in TQFP144 Package (Top View)

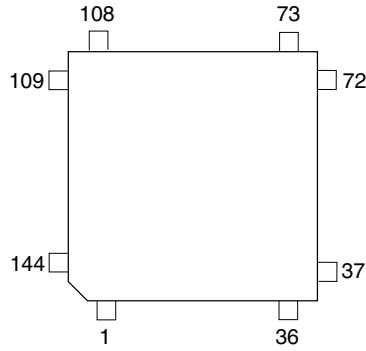


Figure 2. Pin Configuration in BGA144 Package (Top View)

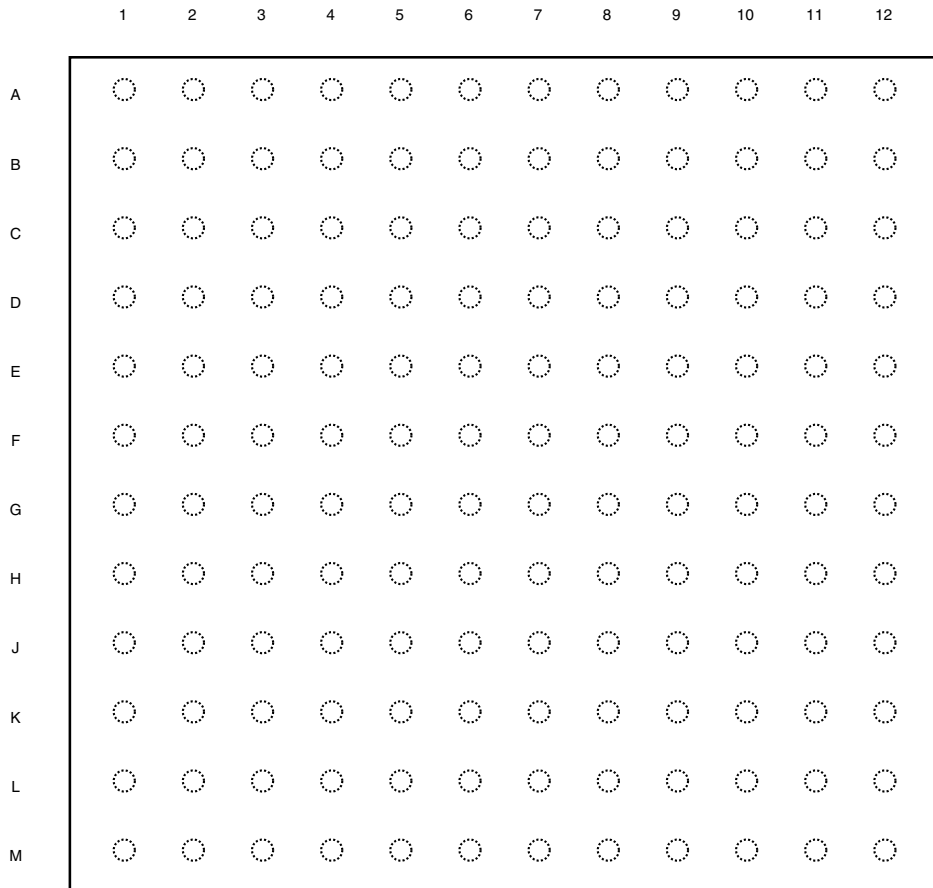


Table 1. AT91M42800A Pinout in TQFP 144 Package

| Pin# | Name | Pin# | Name | Pin# | Name | Pin# | Name |
|------|-------------|------|------------|------|------------------|------|----------|
| 1 | GND | 37 | GND | 73 | GND | 109 | GND |
| 2 | GND | 38 | GND | 74 | GND | 110 | GND |
| 3 | NLB/A0 | 39 | D4 | 75 | PB22/TIOA5 | 111 | PA26 |
| 4 | A1 | 40 | D5 | 76 | PB23/TIOB5 | 112 | MODE0 |
| 5 | A2 | 41 | D6 | 77 | PA0/IRQ0 | 113 | XIN |
| 6 | A3 | 42 | D7 | 78 | PA1/IRQ1 | 114 | XOUT |
| 7 | A4 | 43 | D8 | 79 | PA2/IRQ2 | 115 | GND |
| 8 | A5 | 44 | D9 | 80 | PA3/IRQ3 | 116 | PLLRC A |
| 9 | A6 | 45 | D10 | 81 | PA4/FIQ | 117 | VDDPLL |
| 10 | A7 | 46 | D11 | 82 | PA5/SCK0 | 118 | PLLRCB |
| 11 | A8 | 47 | D12 | 83 | PA6/TXD0 | 119 | VDDPLL |
| 12 | VDDIO | 48 | VDDIO | 84 | VDDIO | 120 | VDDIO |
| 13 | GND | 49 | GND | 85 | GND | 121 | GND |
| 14 | A9 | 50 | D13 | 86 | PA7/RXD0 | 122 | NWDOVF |
| 15 | A10 | 51 | D14 | 87 | PA8/SCK1 | 123 | PA27/BMS |
| 16 | A11 | 52 | D15 | 88 | PA9/TXD1/NTRI | 124 | MODE1 |
| 17 | A12 | 53 | PB6/TCLK0 | 89 | PA10/RXD1 | 125 | TMS |
| 18 | A13 | 54 | PB7/TIOA0 | 90 | PA11/SPCKA | 126 | TDI |
| 19 | A14 | 55 | PB8/TIOB0 | 91 | PA12/MISOA | 127 | TDO |
| 20 | A15 | 56 | PB9/TCLK1 | 92 | PA13/MOSIA | 128 | TCK |
| 21 | A16 | 57 | PB10/TIOA1 | 93 | PA14/NPCSA0/NSSA | 129 | NTRST |
| 22 | A17 | 58 | PB11/TIOB1 | 94 | PA15/NPCSA1 | 130 | NRST |
| 23 | A18 | 59 | PB12/TCLK2 | 95 | PA16/NPCSA2 | 131 | PA28 |
| 24 | VDDIO | 60 | VDDIO | 96 | VDDIO | 132 | VDDIO |
| 25 | GND | 61 | GND | 97 | GND | 133 | GND |
| 26 | A19 | 62 | PB13/TIOA2 | 98 | PA17/NPCSA3 | 134 | PA29/PME |
| 27 | PB2/A20/CS7 | 63 | PB14/TIOB2 | 99 | PA18/SPCKB | 135 | NWAIT |
| 28 | PB3/A21/CS6 | 64 | PB15/TCLK3 | 100 | PA19/MISOB | 136 | NOE/NRD |
| 29 | PB4/A22/CS5 | 65 | PB16/TIOA3 | 101 | PA20/MOSIB | 137 | NWE/NWR0 |
| 30 | PB5/A23/CS4 | 66 | PB17/TIOB3 | 102 | PA21/NPCSB0/NSSB | 138 | NUB/NWR1 |
| 31 | D0 | 67 | PB18/TCLK4 | 103 | PA22/NPCSB1 | 139 | NCS0 |
| 32 | D1 | 68 | PB19/TIOA4 | 104 | PA23/NPCSB2 | 140 | NCS1 |
| 33 | D2 | 69 | PB20/TIOB4 | 105 | PA24/NPCSB3 | 141 | PB0/NCS2 |
| 34 | D3 | 70 | PB21/TCLK5 | 106 | PA25/MCKO | 142 | PB1/NCS3 |
| 35 | VDDCORE | 71 | VDDCORE | 107 | VDDCORE | 143 | VDDCORE |
| 36 | VDDIO | 72 | VDDIO | 108 | VDDIO | 144 | VDDIO |

Table 2. AT91M42800A Pinout in BGA 144 Package

| Pin# | Name |
|------|-------------|
| A1 | PB1/NCS3 |
| A2 | NCS0 |
| A3 | NCS1 |
| A4 | GND |
| A5 | PLLRCB |
| A6 | GND |
| A7 | PLLRCA |
| A8 | GND |
| A9 | XOUT |
| A10 | XIN |
| A11 | MODE0 |
| A12 | PA22/NPCSB1 |
| B1 | NUB/NWR1 |
| B2 | PB0/NCS2 |
| B3 | VDDCORE |
| B4 | NWE/NWR0 |
| B5 | VDDPLL |
| B6 | TDO |
| B7 | VDDPLL |
| B8 | NWDOVF |
| B9 | PA26 |
| B10 | PA19/MISOB |
| B11 | PA24/NPCSB3 |
| B12 | PA23/NPCSB2 |
| C1 | NLB/A0 |
| C2 | A1 |
| C3 | VDDIO |
| C4 | NOE/NRD |
| C5 | VDDIO |
| C6 | NRST |
| C7 | TDI |
| C8 | VDDIO |
| C9 | PA27/BMS |
| C10 | VDDIO |
| C11 | VDDCORE |
| C12 | PA20/MOSIB |

| Pin# | Name |
|------|-------------|
| D1 | A2 |
| D2 | A3 |
| D3 | A4 |
| D4 | NWAIT |
| D5 | PA29/PME |
| D6 | PA28 |
| D7 | TCK |
| D8 | TMS |
| D9 | MODE1 |
| D10 | PA25/MCKO |
| D11 | PA21/NPCSB0 |
| D12 | PA18/SPCKB |
| E1 | A7 |
| E2 | VDDIO |
| E3 | A6 |
| E4 | A5 |
| E5 | GND |
| E6 | GND |
| E7 | GND |
| E8 | NTRST |
| E9 | PA13/MOSIA |
| E10 | PA16/NPCSA2 |
| E11 | VDDIO |
| E12 | PA17/NPCSA3 |
| F1 | A8 |
| F2 | A12 |
| F3 | A9 |
| F4 | A10 |
| F5 | GND |
| F6 | GND |
| F7 | GND |
| F8 | GND |
| F9 | PA12/MISOA |
| F10 | PA15/NPCSA1 |
| F11 | PA11/SPCKA |
| F12 | PA14/NPCSA0 |

| Pin# | Name |
|------|---------------|
| G1 | A17 |
| G2 | A16 |
| G3 | A11 |
| G4 | A13 |
| G5 | GND |
| G6 | GND |
| G7 | GND |
| G8 | GND |
| G9 | PA9/TXD1/NTRI |
| G10 | PA10/RXD1 |
| G11 | PA8/SCK1 |
| G12 | PA7/RXD0 |
| H1 | A18 |
| H2 | VDDIO |
| H3 | A15 |
| H4 | A14 |
| H5 | A19 |
| H6 | GND |
| H7 | GND |
| H8 | GND |
| H9 | PA6/TXD0 |
| H10 | PA4/FIQ |
| H11 | VDDIO |
| H12 | PA5/SCK0 |
| J1 | PB5/A23/CS4 |
| J2 | D0 |
| J3 | PB4/A22/CS5 |
| J4 | PB3/A21/CS6 |
| J5 | PB2/A20/CS7 |
| J6 | D15 |
| J7 | PB6/TCLK0 |
| J8 | PB10/TIOA1 |
| J9 | PA3/IRQ3 |
| J10 | PA2/IRQ2 |
| J11 | PA0/IRQ0 |
| J12 | PA1/IRQ1 |

| Pin# | Name |
|------|------------|
| K1 | D1 |
| K2 | VDDCORE |
| K3 | VDDIO |
| K4 | D9 |
| K5 | D10 |
| K6 | D14 |
| K7 | PB9/TCLK1 |
| K8 | PB13/TIOA2 |
| K9 | PB11/TIOB1 |
| K10 | VDDIO |
| K11 | PB16/TIOA3 |
| K12 | PB23/TIOB5 |
| L1 | D3 |
| L2 | D2 |
| L3 | D5 |
| L4 | D8 |
| L5 | VDDIO |
| L6 | D13 |
| L7 | PB8/TIOB0 |
| L8 | VDDIO |
| L9 | PB17/TIOB3 |
| L10 | VDDCORE |
| L11 | PB20/TIOB4 |
| L12 | PB22/TIOA5 |
| M1 | D4 |
| M2 | D6 |
| M3 | D7 |
| M4 | D11 |
| M5 | D12 |
| M6 | PB7/TIOA0 |
| M7 | PB12/TCLK2 |
| M8 | PB15/TCLK3 |
| M9 | PB14/TIOB2 |
| M10 | PB18/TCLK4 |
| M11 | PB19/TIOA4 |
| M12 | PB21/TCLK5 |

Pin Description

Table 3. AT91M42800A Pin Description

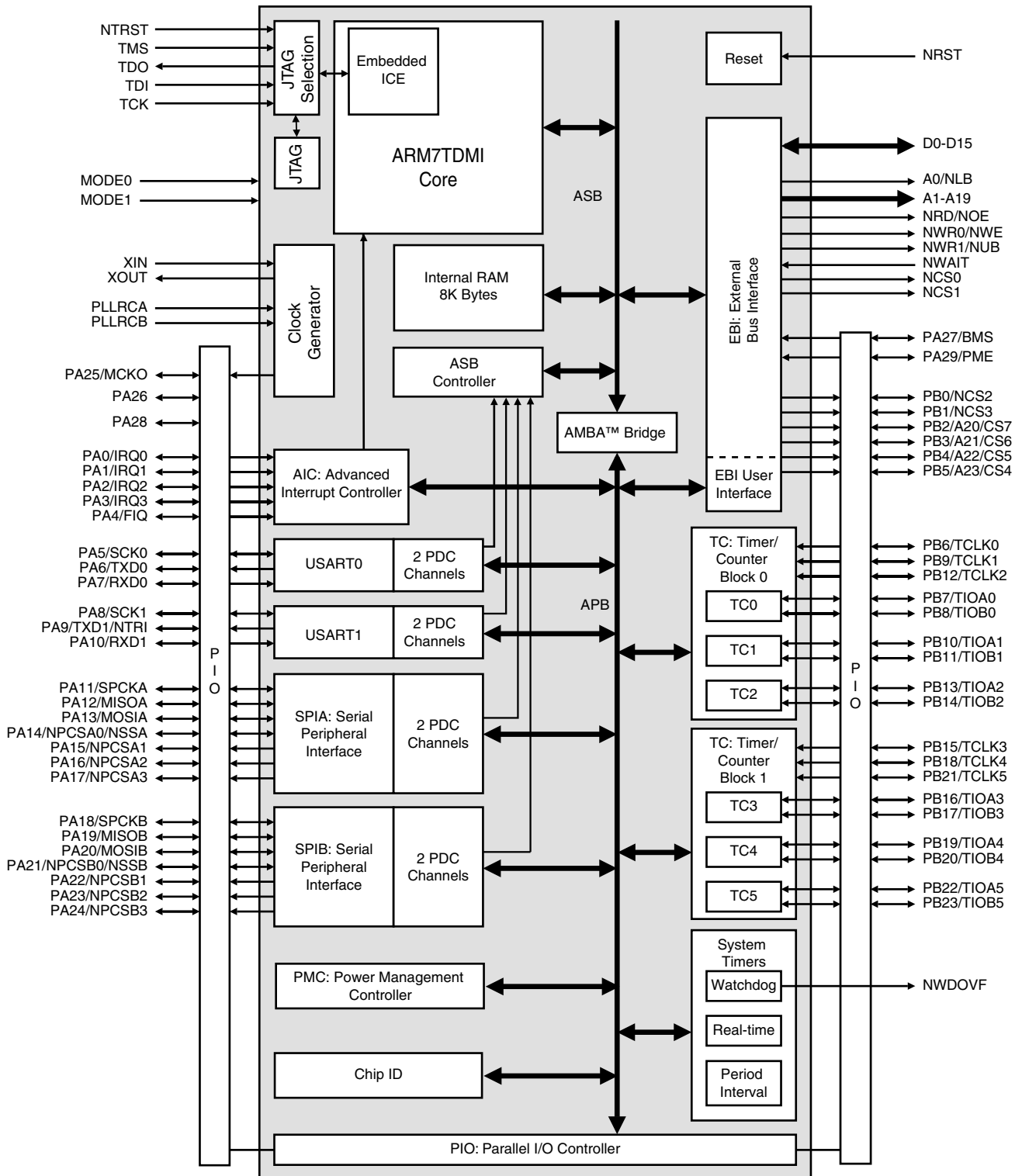
| Module | Name | Function | Type | Active Level | Comments |
|----------------|------------------------------------|------------------------------------|--------|--------------|----------------------------|
| EBI | A0 - A23 | Address Bus | Output | – | All valid after reset |
| | D0 - D15 | Data Bus | I/O | – | |
| | CS4 - CS7 | Chip Select | Output | High | A23 - A20 after reset |
| | NCS0 - NCS3 | Chip Select | Output | Low | |
| | NWR0 | Lower Byte 0 Write Signal | Output | Low | Used in Byte Write option |
| | NWR1 | Lower Byte 1 Write Signal | Output | Low | Used in Byte Write option |
| | NRD | Read Signal | Output | Low | Used in Byte Write option |
| | NWE | Write Enable | Output | Low | Used in Byte Select option |
| | NOE | Output Enable | Output | Low | Used in Byte Select option |
| | NUB | Upper Byte Select (16-bit SRAM) | Output | Low | Used in Byte Select option |
| | NLB | Lower Byte Select (16-bit SRAM) | Output | Low | Used in Byte Select option |
| | NWAIT | Wait Input | Input | Low | |
| | BMS | Boot Mode Select | Input | – | Sampled during reset |
| | PME | Protect Mode Enable | Input | High | PIO-controlled after reset |
| AIC | IRQ0 - IRQ3 | External Interrupt Request | Input | – | PIO-controlled after reset |
| | FIQ | Fast External Interrupt Request | Input | – | PIO-controlled after reset |
| TC | TCLK0 - TCLK5 | Timer External Clock | Input | – | PIO-controlled after reset |
| | TIOA0 - TIOA5 | Multi-purpose Timer I/O Pin A | I/O | – | PIO-controlled after reset |
| | TIOB0 - TIOB5 | Multi-purpose Timer I/O Pin B | I/O | – | PIO-controlled after reset |
| USART | SCK0 - SCK1 | External Serial Clock | I/O | – | PIO-controlled after reset |
| | TXD0 - TXD1 | Transmit Data Output | Output | – | PIO-controlled after reset |
| | RXD0 - RXD1 | Receive Data Input | Input | – | PIO-controlled after reset |
| SPIA SPIB | SPCKA/SPCKB | Clock | I/O | – | PIO-controlled after reset |
| | MISOA/MISOB | Master In Slave Out | I/O | – | PIO-controlled after reset |
| | MOSIA/MOSIB | Master Out Slave In | I/O | – | PIO-controlled after reset |
| | NSSA/NSSB | Slave Select | Input | Low | PIO-controlled after reset |
| | NPCSA0 - NPCSA3 NPCSB0 - NPCSB3 | Peripheral Chip Selects | Output | Low | PIO-controlled after reset |
| PIO | PA0 - PA29 | Programmable I/O Port A | I/O | – | Input after reset |
| | PB0 - PB23 | Programmable I/O Port B | I/O | – | Input after reset |
| ST | NWDOVF | Watchdog Timer Overflow | Output | Low | Open drain |
| CLOCK | XIN | Oscillator Input or External Clock | Input | – | |
| | XOUT | Oscillator Output | Output | – | |
| | PLLRCA | RC Filter for PLL A | Input | – | |
| | PLLRCB | RC Filter for PLL B | Input | – | |
| | MCKO | Clock Output | Output | – | |
| Test and Reset | NRST | Hardware Reset Input | Input | Low | Schmitt trigger |
| | MODE0 - MODE1 | Mode Selection | Input | | |

Table 3. AT91M42800A Pin Description (Continued)

| Module | Name | Function | Type | Active Level | Comments |
|-----------|---------|-----------------------|--------|--------------|-----------------------------------|
| JTAG/ICE | TMS | Test Mode Select | Input | – | Schmitt trigger, internal pull-up |
| | TDI | Test Data In | Input | – | Schmitt trigger, internal pull-up |
| | TDO | Test Data Out | Output | – | |
| | TCK | Test Clock | Input | – | Schmitt trigger, internal pull-up |
| | NTRST | Test Reset Input | Input | Low | Schmitt trigger, internal pull-up |
| Emulation | NTRI | Tri-state Mode Enable | Input | Low | Sampled during reset |
| Power | VDDIO | I/O Power | Power | – | 3V or 5V nominal supply |
| | VDDCORE | Core Power | Power | – | 3V nominal supply |
| | VDDPLL | PLL Power | Power | – | 3V nominal supply |
| | GND | Ground | Ground | – | |

Block Diagram

Figure 3. AT91M42800A



Architectural Overview

The AT91M42800A microcontroller integrates an ARM7TDMI with its embedded ICE interface, memories and peripherals. Its architecture consists of two main buses, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). Designed for maximum performance and controlled by the memory controller, the ASB interfaces the ARM7TDMI processor with the on-chip 32-bit memories, the External Bus Interface (EBI) and the AMBA™ Bridge. The AMBA Bridge drives the APB, which is designed for accesses to on-chip peripherals and optimized for low power consumption.

The AT91M42800A microcontroller implements the ICE port of the ARM7TDMI processor on dedicated pins, offering a complete, low-cost and easy-to-use debug solution for target debugging.

Memories

The AT91M42800A microcontroller embeds up to 8K bytes of internal SRAM. The internal memory is directly connected to the 32-bit data bus and is single-cycle accessible. This provides maximum performance of 30 MIPS at 33 MHz by using the ARM instruction set of the processor. The on-chip memory significantly reduces the system power consumption and improves its performance over external memory solutions.

The AT91M42800A microcontroller features an External Bus Interface (EBI), which enables connection of external memories and application-specific peripherals. The EBI supports 8- or 16-bit devices and can use two 8-bit devices to emulate a single 16-bit device. The EBI implements the early read protocol, enabling single clock cycle memory accesses two times faster than standard memory interfaces.

Peripherals

The AT91M42800A microcontroller integrates several peripherals, which are classified as system or user peripherals. All on-chip peripherals are 32-bit accessible by the AMBA Bridge, and can be programmed with a minimum number of instructions. The peripheral register set is composed of control, mode, data, status and enable/disable/status registers.

An on-chip Peripheral Data Controller (PDC) transfers data between the on-chip USARTs/SPIs and the on- and off-chip memories without processor intervention. Most importantly, the PDC removes the processor interrupt handling overhead and significantly reduces the number of clock cycles required for a data transfer. It can transfer up to 64K continuous bytes without reprogramming the start address. As a result, the performance of the microcontroller is increased and the power consumption reduced.

System Peripherals

The External Bus Interface (EBI) controls the external memory and peripheral devices via an 8- or 16-bit data bus and is programmed through the APB. Each chip select line has its own programming register.

The Power Management Controller (PMC) optimizes power consumption of the product by controlling the clocking elements such as the oscillator and the PLLs, system and user peripheral clocks.

The Advanced Interrupt Controller (AIC) controls the internal sources from the internal peripherals and the five external interrupt lines (including the FIQ) to provide an interrupt and/or fast interrupt request to the ARM7TDMI. It integrates an 8-level priority controller, and, using the Auto-vectoring feature, reduces the interrupt latency time.

The Parallel Input/Output Controllers (PIOA, PIOB) controls up to 54 I/O lines. It enables the user to select specific pins for on-chip peripheral input/output functions, and general-purpose input/output signal pins. The PIO controllers can be programmed to detect an interrupt on a signal change from each line.

There are three embedded system timers. The Real-time Timer (RTT) counts elapsed seconds and can generate periodic or programmed interrupts. The Period Interval Timer (PIT) can be used as a user-programmable time-base, and can generate periodic ticks. The Watchdog (WD) can be used to prevent system lock-up if the software becomes trapped in a deadlock.

The Special Function (SF) module integrates the Chip ID and the Reset Status registers.

User Peripherals

Two USARTs, independently configurable, enable communication at a high baud rate in synchronous or asynchronous mode. The format includes start, stop and parity bits and up to 9 data bits. Each USART also features a Time-out and a Time-guard register, facilitating the use of the two dedicated Peripheral Data Controller (PDC) channels.

The two 3-channel, 16-bit Timer/Counters (TC) are highly-programmable and support capture or waveform modes. Each TC channel can be programmed to measure or generate different kinds of waves, and can detect and control two input/output signals. Each TC also has three external clock signals.

Two independently configurable SPIs provide communication with external devices in master or slave mode. Each has four external chip selects which can be connected to up to 15 devices. The data length is programmable, from 8- to 16-bit.

Associated Documentation

Table 4. Associated Documentation

| Information | Document Title |
|--|--|
| Internal architecture of processor ARM/Thumb instruction sets Embedded in-circuit-emulator | ARM7TDMI (Thumb) Datasheet |
| External memory interface mapping Peripheral operations Peripheral user interfaces | AT91M42800A Datasheet (this document) |
| DC characteristics Power consumption Thermal and reliability chonsiderations AC characteristics | AT91M42800A Electrical Characteristics Datasheet |
| Product overview Ordering information Packaging information Soldering profile | AT91M42800A Summary Datasheet |

Product Overview

Power Supply

The AT91M42800A has three kinds of power supply pins:

- VDDCORE pins that power the chip core
- VDDIO pins that power the I/O lines
- VDDPLL pins that power the oscillator and PLL cells

VDDCORE and VDDIO pins allow core power consumption to be reduced by supplying it with a lower voltage than the I/O lines. The VDDCORE pins must never be powered at a voltage greater than the supply voltage applied to the VDDIO.

The VDDPLL pin is used to supply the oscillator and both PLLs. The voltage applied on these pins is typically 3.3V, and it must not be lower than VDDCORE.

Typical supported voltage combinations are shown in the following table:

| Pins | Nominal Supply Voltages | |
|--------|-------------------------|--------------|
| | VDDCORE | 3.3V |
| VDDIO | 5.0V | 3.0V or 3.3V |
| VDDPLL | 3.3V | 3.0V or 3.3V |

Input/Output Considerations

After the reset, the peripheral I/Os are initialized as inputs to provide the user with maximum flexibility. It is recommended that in any application phase, the inputs to the AT91M42800A microcontroller be held at valid logic levels to minimize the power consumption.

Operating Modes

The AT91M42800A has two pins dedicated to defining MODE0 and MODE1 operating modes. These pins allow the user to enter the device in Boundary Scan mode. They also allow the user to run the processor from the on-chip oscillator output and from an external clock by bypassing the on-chip oscillator. The last mode is reserved for test purposes. A chip reset must be performed (NRST and NTRST) after MODE0 and/or MODE1 have been changed.

| MODE0 | MODE1 | Operating Mode |
|-------|-------|---|
| 0 | 0 | Normal operating mode by using the on-chip oscillator |
| 0 | 1 | Boundary Scan Mode |
| 1 | 0 | Normal operating mode by using an external clock on XIN |
| 1 | 1 | Reserved for test |

Warning: The user must take the external oscillator frequency into account so that it is consistent with the minimum access time requested by the memory device used at the boot. Both the default EBI setting (zero wait state) on Chip Select 0 (See “Boot on NCS0” on page 28) and the minimum access time of the boot memory are two parameters that determine this maximum frequency of the external oscillator.

Clock Generator

The AT91M42800A microcontroller embeds a 32.768 kHz oscillator that generates the Slow Clock (SLCK). This on-chip oscillator can be bypassed by setting the correct logical level on the MODE0 and MODE1 pins, as shown above. In this case, SLCK equals XIN.



The AT91M42800A microcontroller has a fully static design and works either on the Master Clock (MCK), generated from the Slow Clock by means of the two integrated PLLs, or on the Slow Clock (SLCK).

These clocks are also provided as an output of the device on the pin MCKO, which is multiplexed with a general-purpose I/O line. While NRST is active, and after the reset, the MCKO is valid and outputs an image of the SLCK signal. The PIO Controller must be programmed to use this pin as standard I/O line.

Reset

Reset initializes the user interface registers to their default states as defined in the peripheral sections of this datasheet and forces the ARM7TDMI to perform the next instruction fetch from address zero. Except for the program counter, the ARM core registers do not have defined reset states. When reset is active, the inputs of the AT91M42800A must be held at valid logic levels. The EBI address lines drive low during reset. All the peripheral clocks are disabled during reset to save power.

NRST Pin

NRST is the active low reset input. It is asserted asynchronously, but exit from reset is synchronized internally to the slow clock (SLCK). At power-up, NRST must be active until the on-chip oscillator is stable. During normal operation, NRST must be active for a minimum of 10 SLCK clock cycles to ensure correct initialization.

The pins BMS and NTRI are sampled during the 10 SLCK clock cycles just prior to the rising edge of NRST.

The NRST pin has no effect on the on-chip Embedded ICE logic.

Watchdog Reset

The internally generated watchdog reset has the same effect as the NRST pin, except that the pins BMS and NTRI are not sampled. Boot mode and Tri-state mode are not updated. The NRST pin has priority if both types of reset coincide.

Emulation Functions

Tri-state Mode

The AT91M42800A provides a Tri-state mode, which is used for debug purposes in order to connect an emulator probe to an application board. In Tri-state mode the AT91M42800A continues to function, but all the output pin drivers are tri-stated.

To enter Tri-state mode, the pin NTRI must be held low during the last 10 SLCK clock cycles before the rising edge of NRST. For normal operation, the pin NTRI must be held high during reset, by a resistor of up to 400 k Ω . NTRI must be driven to a valid logic value during reset.

NTRI is multiplexed with Parallel I/O PA9 and USART 1 serial data transmit line TXD1.

Standard RS232 drivers generally contain internal 400 k Ω pull-up resistors. If TXD1 is connected to one of these drivers, this pull-up will ensure normal operation, without the need for an additional external resistor.

Embedded ICE

ARM standard embedded in-circuit emulation is supported via the JTAG/ICE port. It is connected to a host computer via an embedded ICE Interface.

Embedded ICE mode is selected when MODE1 is low.

It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed (NRST and NTRST) after MODE0 and/or MODE1 have/has been changed. The reset input to the embedded ICE (NTRST) is provided separately to facilitate debug of boot programs.

IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan is enabled when MODE0 is low and MODE1 is high. The functions SAMPLE, EXTEST and BYPASS are implemented. In ICE Debug mode, the ARM core responds with a non-JTAG chip ID that identifies the core to the ICE system. This is not IEEE 1149.1 JTAG compliant. It is not possible to switch directly between JTAG and ICE operations. A chip reset must be performed (NRST and NTRST) after MODE0 and/or MODE1 have/has been changed.

Memory Controller

The ARM7TDMI processor address space is 4G bytes. The memory controller decodes the internal 32-bit address bus and defines three address spaces:

- Internal Memories in the four lowest megabytes
- Middle Space reserved for the external devices (memory or peripherals) controlled by the EBI
- Internal Peripherals in the four highest megabytes

In any of these address spaces, the ARM7TDMI operates in little-endian mode only.

Protection Mode

The embedded peripherals can be protected against unwanted access. The PME (Protect Mode Enable) pin must be tied high and validated in its peripheral operation (PIO Disable) to enable the protection mode. When enabled, any peripheral access must be done while the ARM7TDMI is running in Privileged mode (i.e., the accesses in user mode result in an abort). Only the valid peripheral address space is protected and requests to the undefined addresses will lead to a normal operation without abort.

Internal Memories

The AT91M42800A microcontroller integrates an 8-Kbyte primary internal SRAM. All internal memories are 32 bits wide and single-clock cycle accessible. Byte (8-bit), half-word (16-bit) or word (32-bit) accesses are supported and are executed within one cycle. Fetching Thumb or ARM instructions is supported and internal memory can store twice as many Thumb instructions as ARM ones.

The SRAM bank is mapped at address 0x0 (after the remap command), and ARM7TDMI exception vectors between 0x0 and 0x20 that can be modified by the software. The rest of the bank can be used for stack allocation (to speed up context saving and restoring), or as data and program storage for critical algorithms.

Boot Mode Select

The ARM reset vector is at address 0x0. After the NRST line is released, the ARM7TDMI executes the instruction stored at this address. This means that this address must be mapped in non-volatile memory after the reset.

The input level on the BMS pin during the last 10 SLCK clock cycles before the rising edge of the NRST selects the type of boot memory. The Boot mode depends on BMS (see Table 5).

The pin BMS is multiplexed with the I/O line PA27 that can be programmed after reset like any standard PIO line.

Table 5. Boot Mode Select

| BMS | Boot Memory |
|-----|--------------------------------|
| 1 | External 8-bit memory NCS0 |
| 0 | External 16-bit memory on NCS0 |

Remap Command

The ARM vectors (Reset, Abort, Data Abort, Prefetch Abort, Undefined Instruction, Interrupt, Fast Interrupt) are mapped from address 0x0 to address 0x20. In order to allow these vectors to be redefined dynamically by the software, the AT91M42800A microcontroller uses a remap command that enables switching between the boot mem-

ory and the internal SRAM bank addresses. The remap command is accessible through the EBI User Interface, by writing one in RCB of EBI_RCR (Remap Control Register). Performing a remap command is mandatory if access to the other external devices (connected to chip selects 1 to 7) is required. The remap operation can only be changed back by an internal reset or an NRST assertion.

Notes: 1. NIRQ de-assertion and automatic interrupt clearing if the source is programmed as level sensitive.

Abort Control

The abort signal providing a Data Abort or a Prefetch Abort exception to the ARM7TDMI is asserted in the following cases:

- When accessing an undefined address in the EBI address space
- When the ARM7TDMI performs a misaligned access

No abort is generated when reading the internal memory or by accessing the internal peripherals, whether the address is defined or not.

When the processor performs a forbidden write access in a mode-protected peripheral register, the write is cancelled but no abort is generated.

The processor can perform word or half-word data access with a misaligned address when a register relative load/store instruction is executed and the register contains a misaligned address. In this case, whether the access is in write or in read, an abort is generated but the access is not cancelled.

The Abort Status Register traces the source that caused the last abort. The address and the type of abort are stored in registers of the External Bus Interface.

External Bus Interface

The External Bus Interface handles the accesses between addresses 0x0040 0000 and 0xFFC0 0000. It generates the signals that control access to the external devices, and can be configured from eight 1-Mbyte banks up to four 16-Mbyte banks. In all cases it supports byte, half-word and word aligned accesses.

For each of these banks, the user can program:

- Number of wait states
- Number of data float times (wait time after the access is finished to prevent any bus contention in case the device takes too long in releasing the bus)
- Data bus width (8-bit or 16-bit)
- With a 16-bit wide data bus, the user can program the EBI to control one 16-bit device (Byte Access Select mode) or two 8-bit devices in parallel that emulate a 16-bit memory (Byte Write Access mode).

The External Bus Interface features also the Early Read Protocol, configurable for all the devices, that significantly reduces access time requirements on an external device.

Peripherals

The AT91M42800A peripherals are connected to the 32-bit wide Advanced Peripheral Bus. Peripheral registers are only word accessible. Byte and half-word accesses are not supported. If a byte or a half-word access is attempted, the memory controller automatically masks the lowest address bits and generates a word access.

Each peripheral has a 16-Kbyte address space allocated (the AIC only has a 4-Kbyte address space).

Peripheral Registers

The following registers are common to all peripherals:

- Control Register – Write-only register that triggers a command when a one is written to the corresponding position at the appropriate address. Writing a zero has no effect.
- Mode Register – read/write register that defines the configuration of the peripheral. Usually has a value of 0x0 after a reset.
- Data Registers – read and/or write register that enables the exchange of data between the processor and the peripheral.
- Status Register – Read-only register that returns the status of the peripheral.
- Enable/Disable/Status Registers are shadow command registers. Writing a one in the Enable Register sets the corresponding bit in the Status Register. Writing a one in the Disable Register resets the corresponding bit and the result can be read in the Status Register. Writing a bit to zero has no effect. This register access method maximizes the efficiency of bit manipulation, and enables modification of a register with a single non-interruptible instruction, replacing the costly read-modify-write operation.

Unused bits in the peripheral registers are shown as “–” and must be written at 0 for upward compatibility. These bits read 0.

Peripheral Interrupt Control

The Interrupt Control of each peripheral is controlled from the status register using the interrupt mask. The status register bits are ANDed to their corresponding interrupt mask bits and the result is then ORed to generate the Interrupt Source signal to the Advanced Interrupt Controller.

The interrupt mask is read in the Interrupt Mask Register and is modified with the Interrupt Enable Register and the Interrupt Disable Register. The enable/disable/status (or mask) makes it possible to enable or disable peripheral interrupt sources with a non-interruptible single instruction. This eliminates the need for interrupt masking at the AIC or Core level in real-time and multi-tasking systems.

Peripheral Data Controller

The AT91M42800A has an 8-channel PDC dedicated to the two on-chip USARTs and to the two on-chip SPIs. One PDC channel is connected to the receiving channel and one to the transmitting channel of each peripheral.

The user interface of a PDC channel is integrated in the memory space of each USART channel and in the memory space of each SPI. It contains a 32-bit address pointer register and a 16-bit count register. When the programmed data is transferred, an end-of-transfer interrupt is generated by the corresponding peripheral. See the USART section and the SPI section for more details on PDC operation and programming.

System Peripherals

PMC: Power Management Controller

The AT91M42800A's Power Management Controller optimizes the power consumption of the device. The PMC controls the clocking elements such as the oscillator and the PLLs, and the System and the Peripheral Clocks. It also controls the MCKO pin and permits to the user to select four different signals to be driven on this pin.

The AT91M42800A has the following clock elements:

- The oscillator providing a clock that depends on the crystal fundamental frequency connected between the XIN and XOUT pins
- PLL A providing a low-to-middle frequency clock range
- PLL B providing a middle-to-high frequency range
- The Clock prescaler
- The ARM Processor Clock controller
- The Peripheral Clock controller
- The Master Clock Output controller

The on-chip low-power oscillator together with the PLL-based frequency multiplier and the prescaler results in a programmable clock between 500 Hz and 66 MHz. It is the responsibility of the user to make sure that the PMC programming does not result in a clock over the acceptable limits.

ST: System Timer

The System Timer module integrates three different free-running timers:

- A Period Interval Timer setting the base time for an Operating System.
- A Watchdog Timer that is built around a 16-bit counter, and is used to prevent system lock-up if the software becomes trapped in a deadlock. It can generate an internal reset or interrupt, or assert an active level on the dedicated pin NWDOVF.
- A Real-time Timer counting elapsed seconds.

These timers count using the Slow Clock. Typically, this clock has a frequency of 32768 Hz.

AIC: Advanced Interrupt Controller

The AT91M42800A has an 8-level priority, individually maskable, vectored interrupt controller. This feature substantially reduces the software and real-time overhead in handling internal and external interrupts.

The interrupt controller is connected to the NFIQ (fast interrupt request) and the NIRQ (standard interrupt request) inputs of the ARM7TDMI processor. The processor's NFIQ line can only be asserted by the external fast interrupt request input: FIQ. The NIRQ line can be asserted by the interrupts generated by the on-chip peripherals and the external interrupt request lines: IRQ0 to IRQ3.

The 8-level priority encoder allows the customer to define the priority between the different NIRQ interrupt sources.

Internal sources are programmed to be level sensitive or edge triggered. External sources can be programmed to be positive or negative edge triggered or high- or low-level sensitive.

PIO: Parallel I/O Controller

The AT91M42800A has 54 programmable I/O lines. I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins. These lines are controlled by two separate and identical PIO Controllers called PIOA and PIOB. Each PIO controller also provides an internal interrupt signal to the Advanced Interrupt Controller and insertion of a simple input glitch filter on any of the PIO pins.

SF: Special Function

The AT91M42800A provides registers that implement the following special functions.

- Chip Identification
- RESET Status

User Peripherals**USART: Universal Synchronous/Asynchronous Receiver Transmitter**

The AT91M42800A provides two identical, full-duplex, universal synchronous/asynchronous receiver/transmitters that interface to the APB and are connected to the Peripheral Data Controller.

The main features are:

- Programmable Baud Rate Generator with External or Internal Clock, as well as Slow Clock
- Parity, Framing and Overrun Error Detection
- Line Break Generation and Detection
- Automatic Echo, Local Loopback and Remote Loopback channel modes
- Multi-drop mode: Address Detection and Generation
- Interrupt Generation
- Two Dedicated Peripheral Data Controller channels
- 5-, 6-, 7-, 8- and 9-bit character length

TC: Timer/Counter

The AT91M42800A features two Timer/Counter blocks, each containing three identical 16-bit Timer/Counter channels. Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse-width modulation.

Each Timer/Counter (TC) channel has 3 external clock inputs, 5 internal clock inputs, and 2 multi-purpose input/output signals that can be configured by the user. Each channel drives an internal interrupt signal that can be programmed to generate processor interrupts via the AIC (Advanced Interrupt Controller).

The Timer/Counter block has two global registers that act upon all three TC channels. The Block Control Register allows the three channels to be started simultaneously with the same instruction. The Block Mode Register defines the external clock inputs for each Timer/Counter channel, allowing them to be chained.

Each Timer/Counter block operates independently and has a complete set of block and channel registers.

SPI: Serial Peripheral Interface

The AT91M42800A includes two SPIs that provide communication with external devices in Master or Slave mode. They are independent, and are referred to by the letters A and B. Each SPI has four external chip selects that can be connected to up to 15 devices. The data length is programmable from 8- to 16-bit.

Memory Map

Figure 4. AT91M42800A Memory Map before Remap Command

| Address | Function | Size | Protection ⁽¹⁾ | Abort Control |
|--------------------------|---|----------|---------------------------|---------------|
| 0xFFFFFFF | On-chip Peripherals | 4M Bytes | Privileged | Yes |
| 0xFFC00000 0xFFBFFFFF | Reserved | | | |
| 0x00400000 0x003FFFFF | On-chip SRAM | 1M Byte | No | No |
| 0x00300000 0x002FFFFF | Reserved On-chip Device | 1M Byte | No | No |
| 0x00200000 0x001FFFFF | Reserved On-chip Device | 1M Byte | No | No |
| 0x00100000 0x000FFFFF | External Devices Selected by NCS0 | 1M Byte | No | No |
| 0x00000000 | | | | |

Note: 1. The ARM core modes are defined in the ARM7TDMI Datasheet. Privileged is a non-user mode. The protection is active only if Protect mode is enabled.

Figure 5. AT91M42800A Memory Map after Remap Command

| Address | Function | Size | Protection ⁽¹⁾ | Abort Control |
|-------------|----------------------------------|--|---------------------------|---------------|
| 0xFFFFFFFF | On-chip Peripherals | 4M Bytes | Privileged | Yes |
| 0xFFC00000 | | | | |
| 0xFFBFFFFFF | External Devices (up to 8) | Up to 8 Devices Programmable Page Size 1, 4, 16, 64M Bytes | No | Yes |
| 0x00400000 | | | | |
| 0x003FFFFFF | Reserved | 1M Byte | No | No |
| 0x00300000 | | | | |
| 0x002FFFFFF | Reserved On-chip Device | 1M Byte | No | No |
| 0x00200000 | | | | |
| 0x001FFFFFF | Reserved On-chip Device | 1M Byte | No | No |
| 0x00100000 | | | | |
| 0x000FFFFFF | On-chip RAM | 1M Byte | No | No |
| 0x00000000 | | | | |

Note: 1. The ARM core modes are defined in the ARM7TDMI Datasheet. Privileged is a non-user mode. The protection is active only if Protect mode is enabled.

Peripheral Memory Map

Figure 6. AT91M42800A Peripheral Memory Map

| Address | Peripheral | Peripheral Name | Size | Protection |
|--|------------|--|-----------|------------|
| 0xFFFFFFF | AIC | Advanced Interrupt Controller | 4K Bytes | Privileged |
| 0xFFFFF000 | | Reserved | | |
| 0xFFFFEFFF 0xFFFFC000 0xFFFFBFFF | ST | System Timer | 16K Bytes | Privileged |
| 0xFFFF8000 | | Reserved | | |
| 0xFFFF7FFF | PMC | Power Management Controller | 16K Bytes | Privileged |
| 0xFFFF4000 | | Reserved | | |
| 0xFFFF3FFF | PIOB | Parallel I/O Controller B | 16K Bytes | Privileged |
| 0xFFFF0000 | | Reserved | | |
| 0xFFFEFFF | PIOA | Parallel I/O Controller A | 16K Bytes | Privileged |
| 0xFFFE000 | | Reserved | | |
| 0xFFFBFFF 0xFFFD8000 | | Reserved | | |
| 0xFFFD7FFF | TC1 | Timer Counter 1 Channels 3, 4 and 5 | 16K Bytes | Privileged |
| 0xFFFD4000 | | Reserved | | |
| 0xFFFD3FFF | TC0 | Timer Counter 0 Channels 0,1 and 2 | 16K Bytes | Privileged |
| 0xFFFD000 | | Reserved | | |
| 0xFFFCFFF | SPIB | Serial Peripheral Interface B | 16K Bytes | Privileged |
| 0xFFFC000 | | Reserved | | |
| 0xFFFBFFF | SPIA | Serial Peripheral Interface A | 16K Bytes | Privileged |
| 0xFFFC8000 | | Reserved | | |
| 0xFFFC7FFF | USART1 | Universal Synchronous/ Asynchronous Receiver/Transmitter 1 | 16K Bytes | Privileged |
| 0xFFFC4000 | | Reserved | | |
| 0xFFFC3FFF | USART0 | Universal Synchronous/ Asynchronous Receiver/Transmitter 0 | 16K Bytes | Privileged |
| 0xFFFC000 | | Reserved | | |
| 0xFFFBFFF 0xFF04000 | | Reserved | | |
| 0xFF03FFF | SF | Special Function | 16K Bytes | Privileged |
| 0xFF00000 | | Reserved | | |
| 0xFFEFFF 0xFF04000 | | Reserved | | |
| 0xFFE03FFF | EBI | External Bus Interface | 16K Bytes | Privileged |
| 0xFFE0000 | | Reserved | | |
| 0xFFDFFF 0xFFD0000 | | Reserved | | |

Note: 1. The ARM core modes are defined in the ARM7TDMI Datasheet. Privileged is a non-user mode. The protection is active only if Protect mode is enabled.

EBI: External Bus Interface

The EBI handles the access requests performed by the ARM core or the PDC. It generates the signals that control the access to the external memory or peripheral devices. The EBI is fully programmable and can address up to 64M bytes. It has eight chip selects and a 24-bit address bus, the upper four bits of which are multiplexed with a chip select.

The 16-bit data bus can be configured to interface with 8- or 16-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing.

The EBI supports different access protocols allowing single clock cycle memory accesses.

The main features are:

- External memory mapping
- Up to 8 chip select lines
- 8- or 16-bit data bus
- Byte write or byte select lines
- Remap of boot memory
- Two different read protocols
- Programmable wait state generation
- External wait request
- Programmable data float time

The EBI User Interface is described on page 48.

External Memory Mapping

The memory map associates the internal 32-bit address space with the external 24-bit address bus.

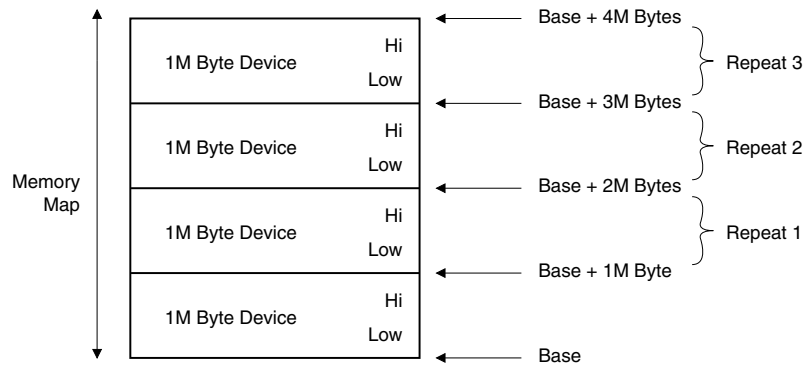
The memory map is defined by programming the base address and page size of the external memories (see EBI User Interface registers EBI_CSR0 to EBI_CSR7). Note that A0 - A23 is only significant for 8-bit memory; A1 - A23 is used for 16-bit memory.

If the physical memory device is smaller than the programmed page size, it wraps around and appears to be repeated within the page. The EBI correctly handles any valid access to the memory device within the page (see Figure 7).

In the event of an access request to an address outside any programmed page, an abort signal is generated. Two types of abort are possible: instruction prefetch abort and data abort. The corresponding exception vector addresses are 0x0000000C and 0x00000010, respectively. It is up to the system programmer to program the error handling routine to use in case of an abort (see the ARM7TDMI datasheet for further information).

The chip selects can be defined to the same base address and an access to the overlapping address space asserts both NCS lines. The Chip Select Register, having the smaller number, defines the characteristics of the external access and the behaviour of the control signals.

Figure 7. External Memory Smaller than Page Size



Abort Status

When an abort is generated, the EBI_AASR (Abort Address Status Register) and the EBI_ASR (Abort Status Register) provide the details of the source causing the abort. Only the last abort is saved and registers are left in the last abort status. After the reset, the registers are initialized to 0.

The following are saved:

In EBI_AASR:

- The address at which the abort is generated

In EBI_ASR:

- Whether or not the processor has accessed an undefined address in the EBI address space
- Whether or not the processor required an access at a misaligned address
- The size of the access (byte, word or half-word)
- The type of the access (read, write or code fetch)

EBI Behavior During Internal Accesses

When the ARM core performs accesses in the internal memories or the embedded peripherals, the EBI signals behave as follows:

- The address lines remain at the level of the last external access.
- The data bus is tri-stated.
- The control signals remain in an inactive state.

Pin Description

Table 6. External Bus Interface Pin Description

| Name | Description | Type |
|-------------|------------------------------|-------------|
| A0 - A23 | Address bus | Output |
| D0 - D15 | Data bus | I/O |
| NCS0 - NCS3 | Active low chip selects | Output |
| CS4 - CS7 | Active high chip selects | Output |
| NRD | Read Enable | Output |
| NWR0 - NWR1 | Lower and upper write enable | Output |
| NOE | Output enable | Output |
| NWE | Write enable | Output |
| NUB, NLB | Upper and lower byte select | Output |
| NWAIT | Wait request | Input |
| PME | Protection Mode Enabled | Input |

Table 7. EBI Multiplexed Signals

| Multiplexed Signals | | Functions |
|----------------------------|-----------|---|
| A23 - A20 | CS4 - CS7 | Allows from 4 to 8 chip select lines to be used |
| A0 | NLB | 8- or 16-bit data bus |
| NRD | NOE | Byte-write or byte select access |
| NWR0 | NWE | Byte-write or byte select access |
| NWR1 | NUB | Byte-write or byte select access |

Chip Select Lines

The EBI provides up to eight chip select lines:

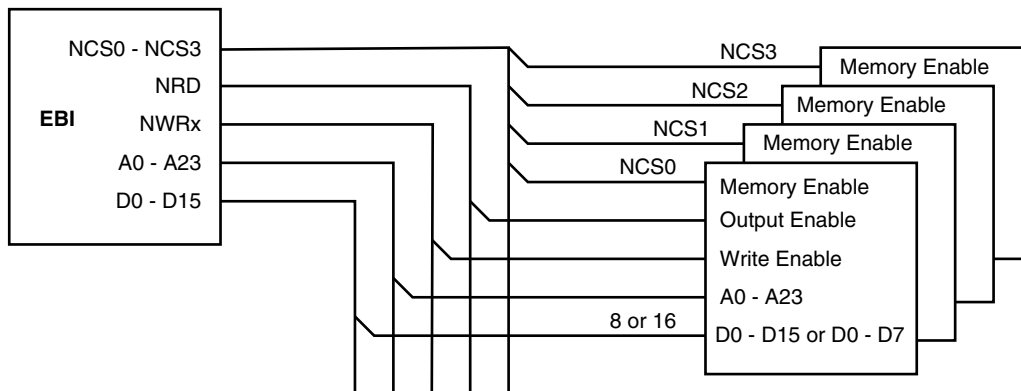
- Chip select lines NCS0 - NCS3 are dedicated to the EBI (not multiplexed).
- Chip select lines CS4 - CS7 are multiplexed with the top four address lines A23 - A20.

By exchanging address lines for chip select lines, the user can optimize the EBI to suit the external memory requirements: more external devices or larger address range for each device.

The selection is controlled by the ALE field in EBI_MCR (Memory Control Register). The following combinations are possible:

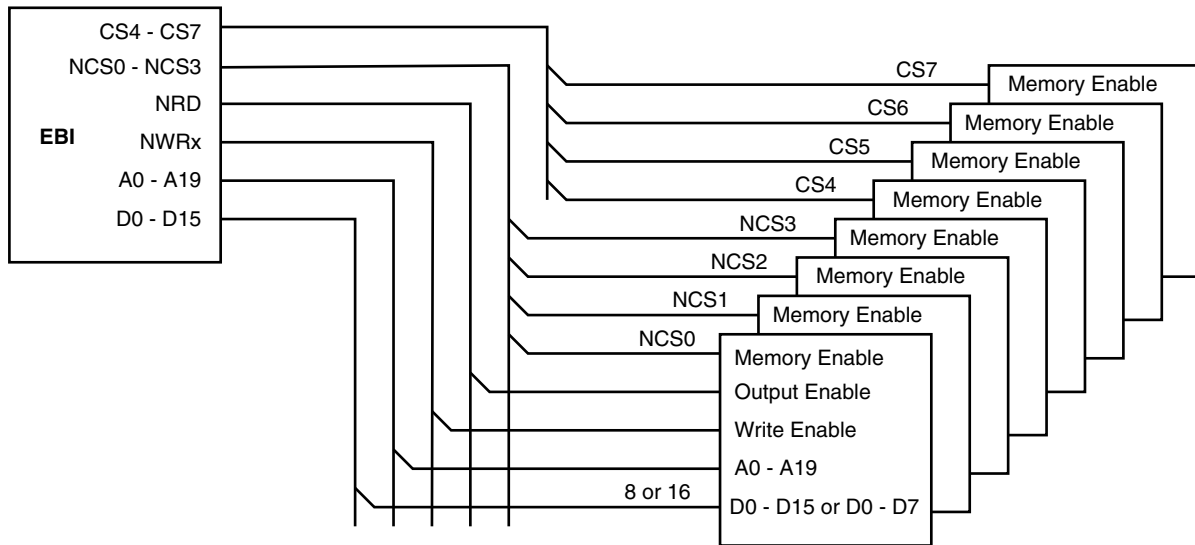
- A20, A21, A22, A23 (configuration by default)
- A20, A21, A22, CS4
- A20, A21, CS5, CS4
- A20, CS6, CS5, CS4
- CS7, CS6, CS5, CS4

Figure 8. Memory Connections for Four External Devices⁽¹⁾



Notes: 1. For four external devices, the maximum address space per device is 16M bytes.

Figure 9. Memory Connections for Eight External Devices⁽¹⁾



Notes: 1. For eight external devices, the maximum address space per device is 1M byte.

Data Bus Width

A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the DBW field in the EBI_CSR (Chip Select Register) for the corresponding chip select.

Figure 10 shows how to connect a 512K x 8-bit memory on NCS2.

Figure 10. Memory Connection for an 8-bit Data Bus

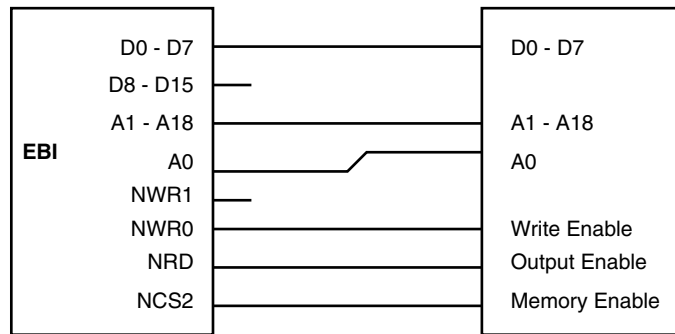
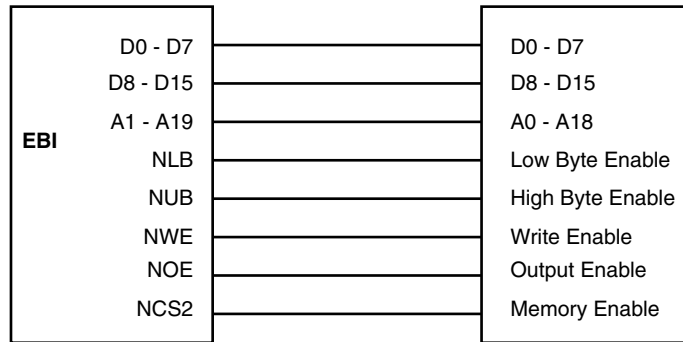


Figure 11 shows how to connect a 512K x 16-bit memory on NCS2.

Figure 11. Memory Connection for a 16-bit Data Bus



Byte Write or Byte Select Access

Each chip select with a 16-bit data bus can operate with one of two different types of write access:

- Byte Write Access supports two byte write and a single read signal.
- Byte Select Access selects upper and/or lower byte with two byte select lines, and separate read and write signals.

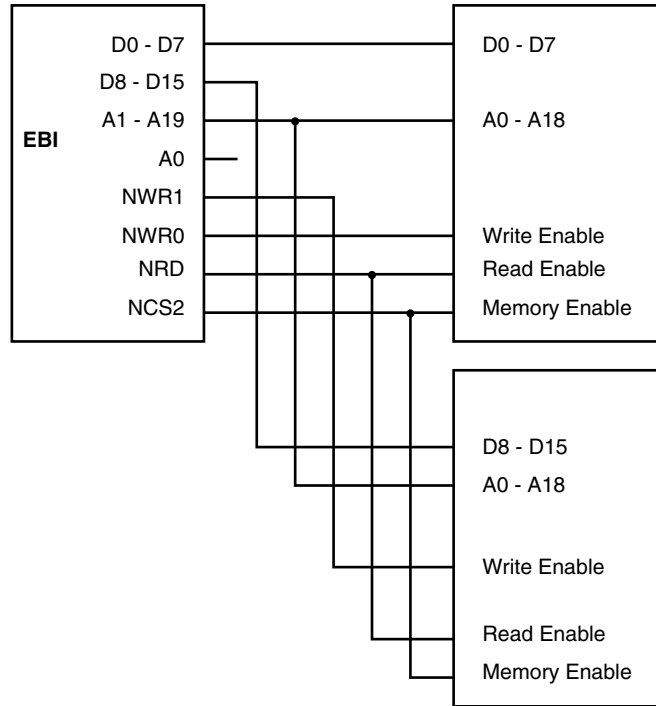
This option is controlled by the BAT field in the EBI_CSR (Chip Select Register) for the corresponding chip select.

Byte Write Access is used to connect 2 x 8-bit devices as a 16-bit memory page.

- The signal A0/NLB is not used.
- The signal NWR1/NUB is used as NWR1 and enables upper byte writes.
- The signal NWR0/NWE is used as NWR0 and enables lower byte writes.
- The signal NRD/NOE is used as NRD and enables half-word and byte reads.

Figure 12 shows how to connect two 512K x 8-bit devices in parallel on NCS2.

Figure 12. Memory Connection for 2 x 8-bit Data Buses



Byte Select Access is used to connect 16-bit devices in a memory page.

- The signal A0/NLB is used as NLB and enables the lower byte for both read and write operations.
- The signal NWR1/NUB is used as NUB and enables the upper byte for both read and write operations.
- The signal NWR0/NWE is used as NWE and enables writing for byte or half-word.
- The signal NRD/NOE is used as NOE and enables reading for byte or half-word.

Figure 13 shows how to connect a 16-bit device with byte and half-word access (e.g., 16-bit SRAM) on NCS2.

Figure 13. Connection for a 16-bit Data Bus with Byte and Half-word Access

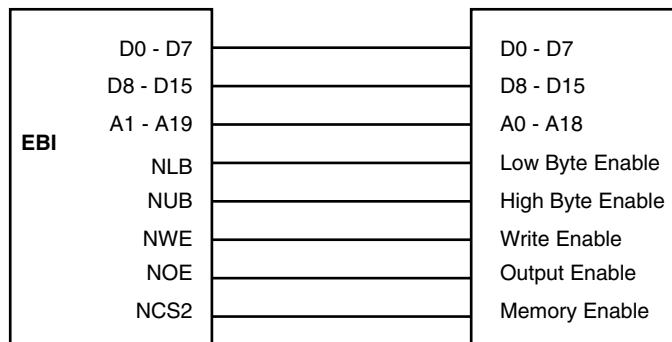
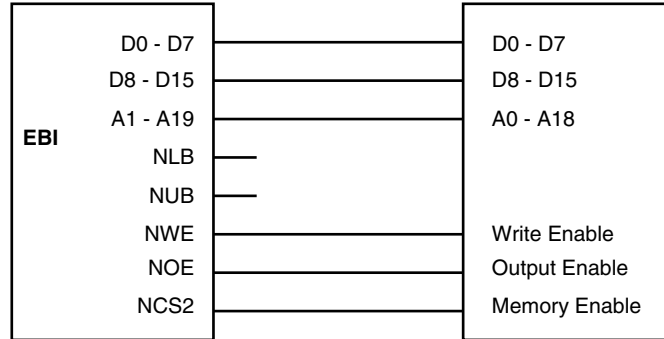


Figure 14 shows how to connect a 16-bit device without byte access (e.g., Flash) on NCS2.

Figure 14. Connection for a 16-bit Data Bus without Byte Write Capability



Boot on NCS0

Depending on the device and the BMS pin level during the reset, the user can select either an 8-bit or 16-bit external memory device connected on NCS0 as the Boot memory. In this case, EBI_CSR0 (Chip Select Register 0) is reset at the following configuration for chip select 0:

- 0 wait states (WSE = 0, NWS = 7)
- 8-bit or 16-bit data bus width, depending on BMS

Byte access type and number of data float time are set to Byte Write Access and 0, respectively.

Before the remap command, the user can modify the chip select 0 configuration, programming the EBI_CSR0 with the exact Boot memory characteristics. The base address becomes effective after the remap command.

Warning: In the internal oscillator bypass mode described in “Operating Modes” on page 11, the user must take the external oscillator frequency into account according to the minimum access time on the boot memory device.

As illustration, the following table gives examples of oscillator frequency limits according to the time access without using NWAIT pin at the boot.

| Chip Select Assertion to Output Data Valid Maximum Delay in Read Cycle (t_{CE} in ns) | External Oscillator Frequency Limit (MHz) |
|---|--|
| 110 | 7 |
| 90 | 9 |
| 70 | 11 |
| 55 | 14 |
| 25 | 24 |

Note: Values take only t_{CE} into account.

Read Protocols

The EBI provides two alternative protocols for external memory read access: standard and early read. The difference between the two protocols lies in the timing of the NRD (read cycle) waveform.

The protocol is selected by the DRP field in EBI_MCR (Memory Control Register) and is valid for all memory devices. Standard read protocol is the default protocol after reset.

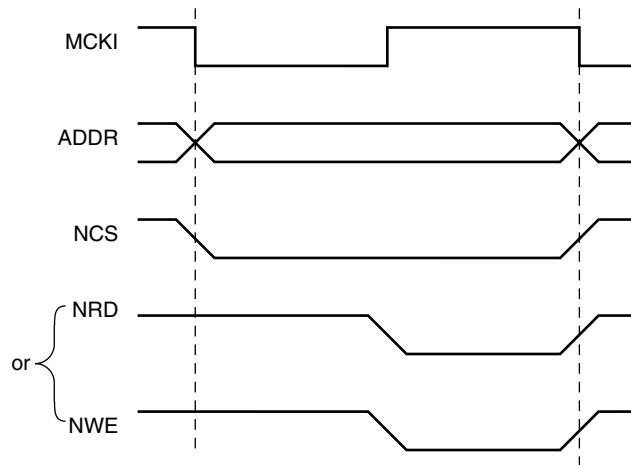
Note: In the following waveforms and descriptions, **NRD** represents NRD and NOE since the two signals have the same waveform. Likewise, **NWE** represents NWE, NWR0 and NWR1 unless NWR0 and NWR1 are otherwise represented. **ADDR** represents A0 - A23 and/or A1 - A23.

Standard Read Protocol

Standard read protocol implements a read cycle in which NRD and NWE are similar. Both are active during the second half of the clock cycle. The first half of the clock cycle allows time to ensure completion of the previous access as well as the output of address and NCS before the read cycle begins.

During a standard read protocol, external memory access, NCS is set low and ADDR is valid at the beginning of the access while NRD goes low only in the second half of the master clock cycle to avoid bus conflict (see Figure 15).

Figure 15. Standard Read Protocol

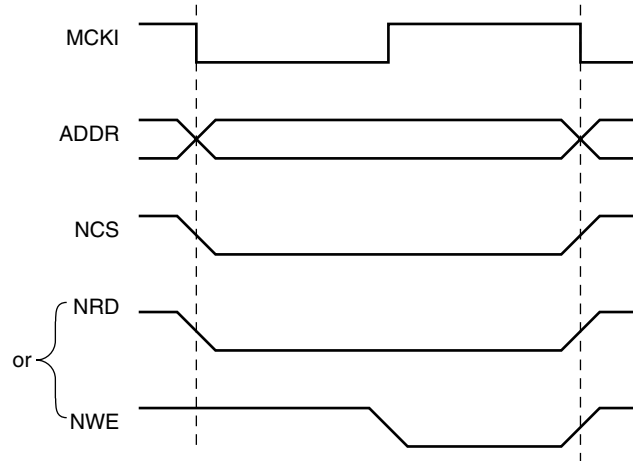


NWE is the same in both protocols. NWE always goes low in the second half of the master clock cycle (see Figure 17).

Early Read Protocol

Early read protocol provides more time for a read access from the memory by asserting NRD at the beginning of the clock cycle. In the case of successive read cycles in the same memory, NRD remains active continuously. Since a read cycle normally limits the speed of operation of the external memory system, early read protocol can allow a faster clock frequency to be used. However, an extra wait state is required in some cases to avoid contentions on the external bus.

Figure 16. Early Read Protocol



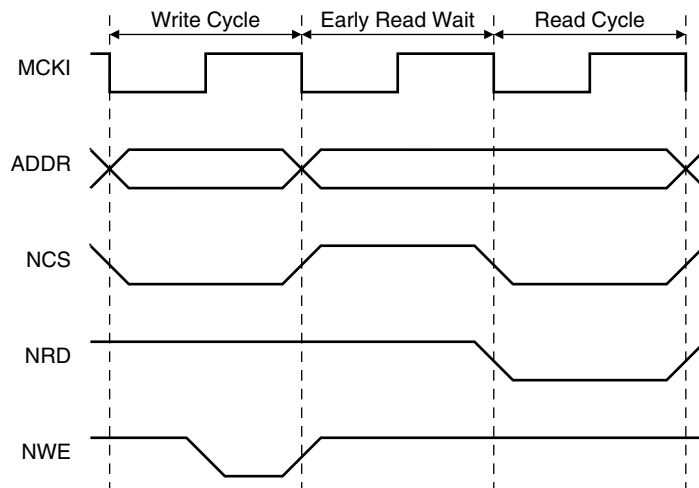
Early Read Wait State

In early read protocol, an early read wait state is automatically inserted when an external write cycle is followed by a read cycle to allow time for the write cycle to end before the subsequent read cycle begins (see Figure 17). This wait state is generated in addition to any other programmed wait states (i.e., data float wait).

No wait state is added when a read cycle is followed by a write cycle, between consecutive accesses of the same type or between external and internal memory accesses.

Early read wait states affect the external bus only. They do not affect internal bus timing.

Figure 17. Early Read Wait State

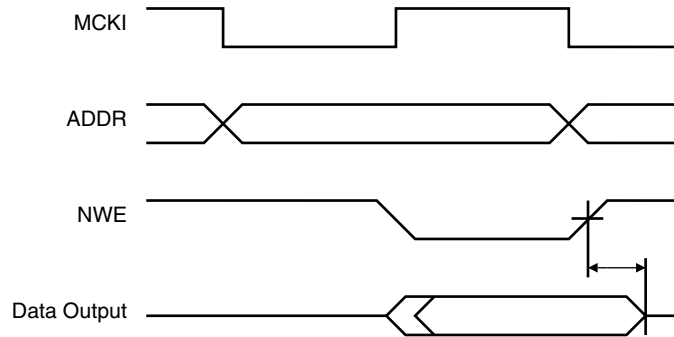


Write Data Hold Time

During write cycles in both protocols, output data becomes valid after the falling edge of the NWE signal and remains valid after the rising edge of NWE, as illustrated in Figure 18. The external NWE waveform (on the NWE pin) is used to control the output data timing to guarantee this operation.

It is therefore necessary to avoid excessive loading of the NWE pins, which could delay the write signal too long and cause a contention with a subsequent read cycle in standard protocol.

Figure 18. Data Hold Time



In early read protocol the data can remain valid longer than in standard read protocol due to the additional wait cycle which follows a write access.

Wait States

The EBI can automatically insert wait states. The different types of wait states are listed below:

- Standard wait states
- Data float wait states
- External wait states
- Chip select change wait states
- Early Read wait states (as described in “Read Protocols” on page 29)

Standard Wait States

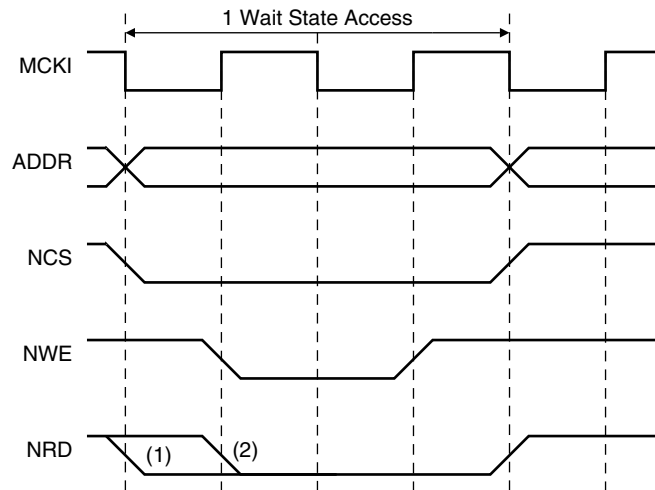
Each chip select can be programmed to insert one or more wait states during an access on the corresponding device. This is done by setting the WSE field in the corresponding EBI_CSR. The number of cycles to insert is programmed in the NWS field in the same register.

Below is the correspondence between the number of standard wait states programmed and the number of cycles during which the NWE pulse is held low:

| | |
|---------------|-----------|
| 0 wait states | 1/2 cycle |
| 1 wait state | 1 cycle |

For each additional wait state programmed, an additional cycle is added.

Figure 19. One Wait State Access



- Notes:
1. Early Read Protocol
 2. Standard Read Protocol

Data Float Wait State

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float waits) after a read access before starting a write access or a read access to a different external memory.

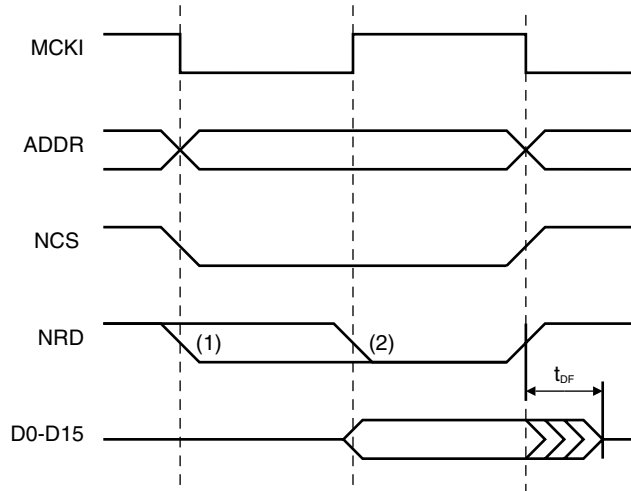
The data float output time (t_{DF}) for each external memory device is programmed in the TDF field of the EBI_CSR register for the corresponding chip select. The value (0 - 7 clock cycles) indicates the number of data float waits to be inserted and represents the time allowed for the data output to go high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The EBI keeps track of the programmed external data float time during internal accesses, to ensure that the external memory system is not accessed while it is still busy.

Internal memory accesses and consecutive accesses to the same external memory do not have added data float wait states.

Figure 20. Data Float Output Time



- Notes: 1. Early Read Protocol
2. Standard Read Protocol

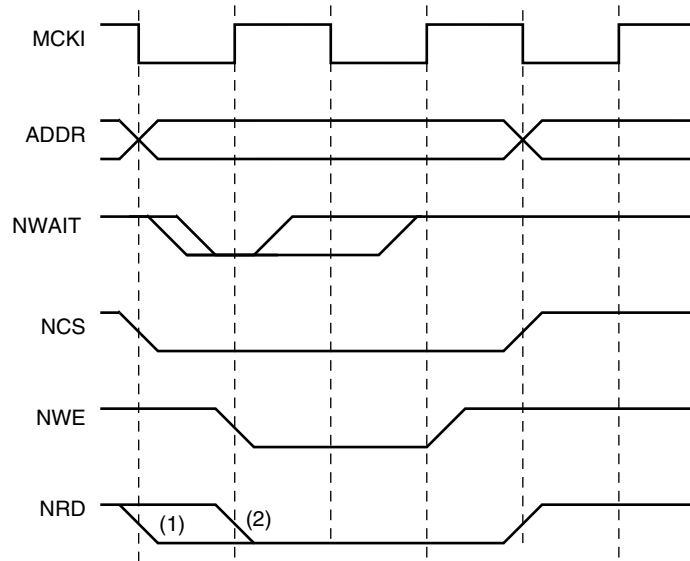
External Wait

The NWAIT input can be used to add wait states at any time. NWAIT is active low and is detected on the rising edge of the clock.

If NWAIT is low at the rising edge of the clock, the EBI adds a wait state and changes neither the output signals nor its internal counters and state. When NWAIT is de-asserted, the EBI finishes the access sequence.

The NWAIT signal must meet setup and hold requirements on the rising edge of the clock.

Figure 21. External Wait

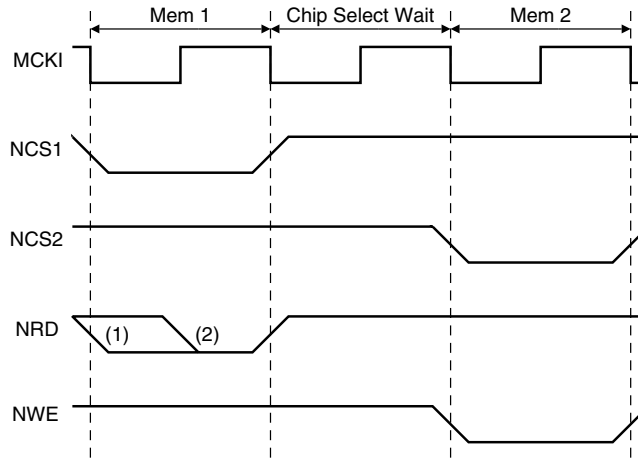


- Notes: 1. Early Read Protocol
2. Standard Read Protocol

Chip Select Change Wait States

A chip select wait state is automatically inserted when consecutive accesses are made to two different external memories (if no wait states have already been inserted). If any wait states have already been inserted, (e.g., data float wait) then none are added.

Figure 22. Chip Select Wait



- Notes: 1. Early Read Protocol
2. Standard Read Protocol

Memory Access Waveforms

Figures 23 through 26 show examples of the two alternative protocols for external memory read access.

Figure 23. Standard Read Protocol without t_{DF}

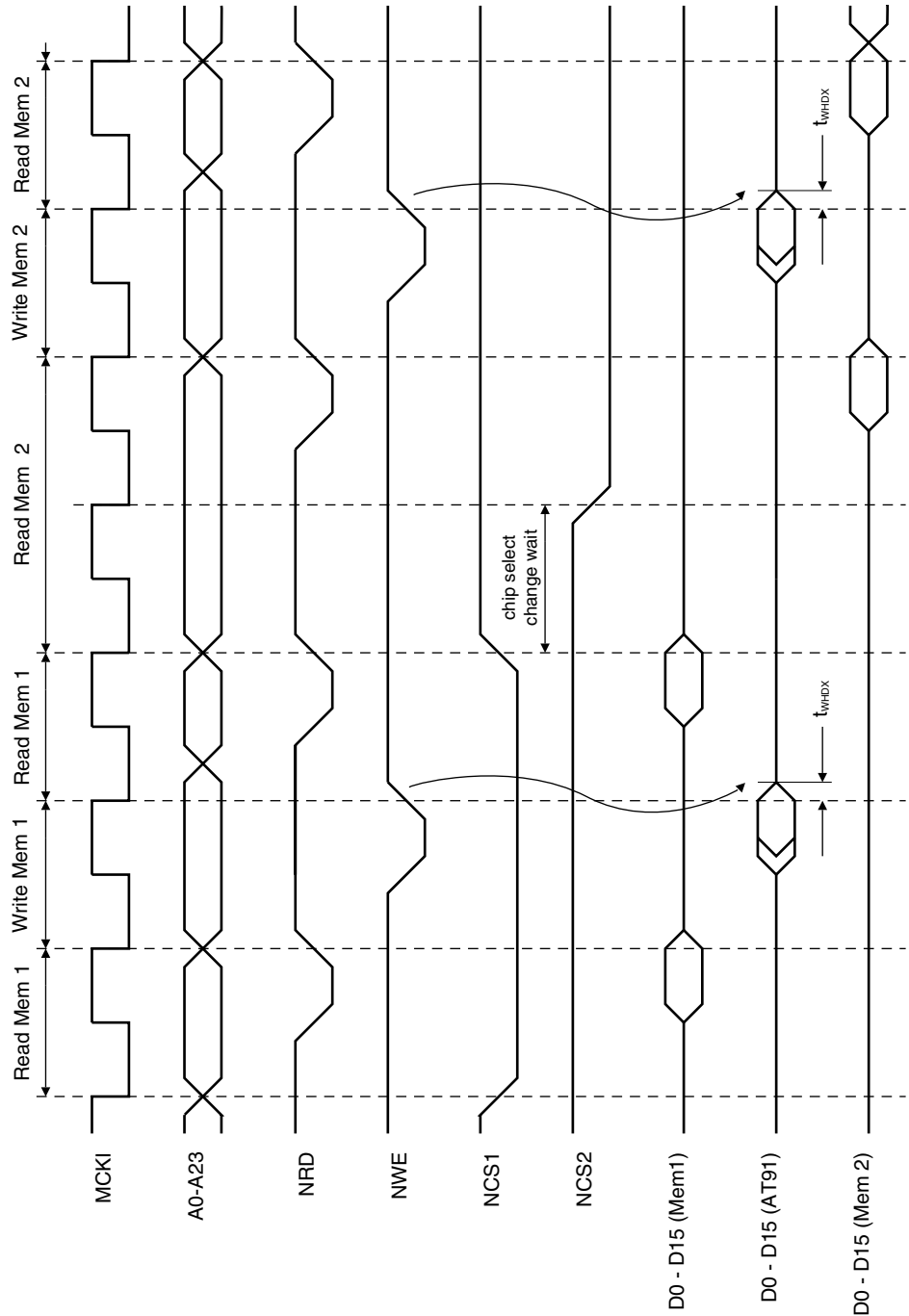


Figure 24. Early Read Protocol without t_{DF}

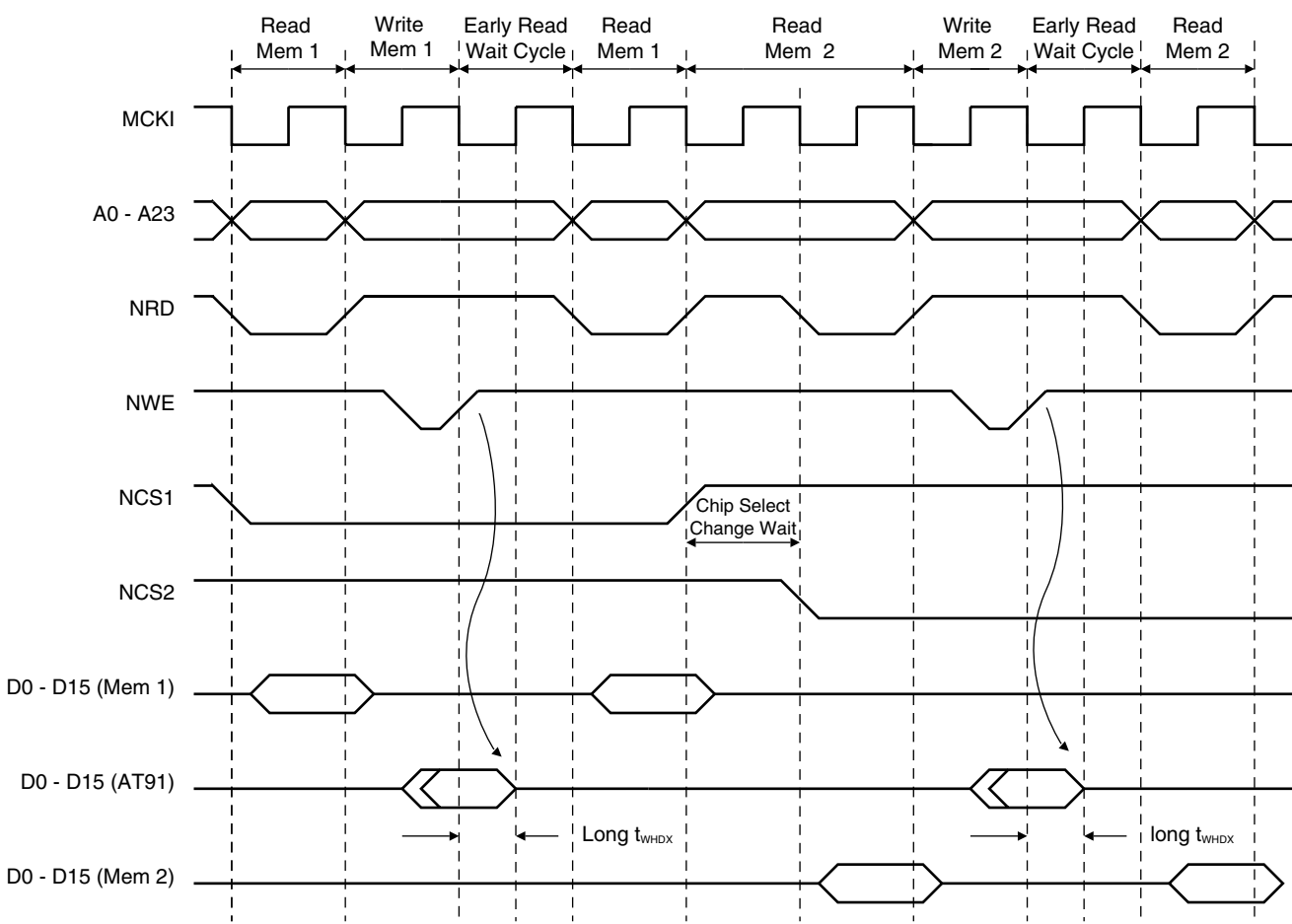


Figure 25. Standard Read Protocol with t_{DF}

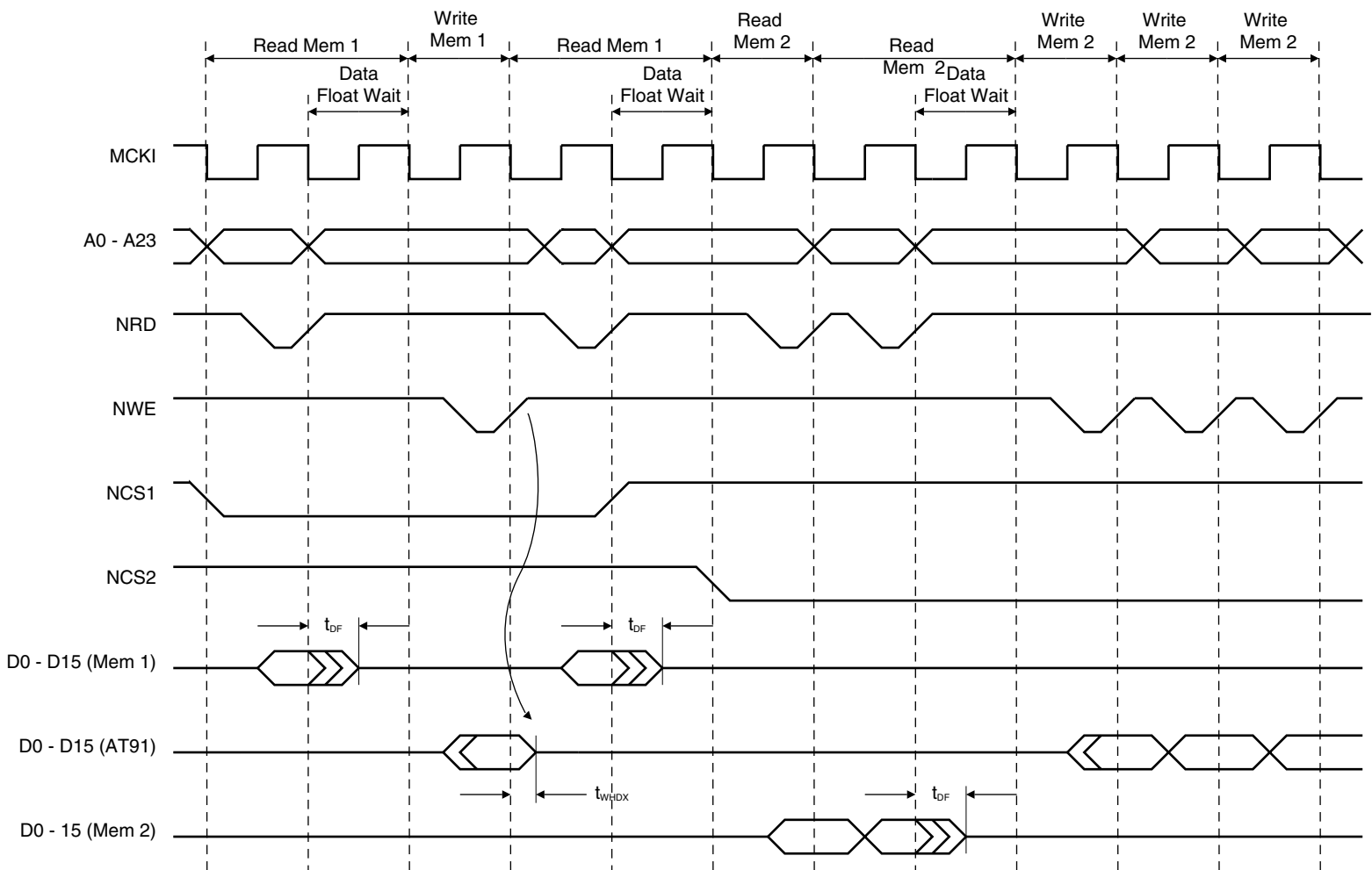
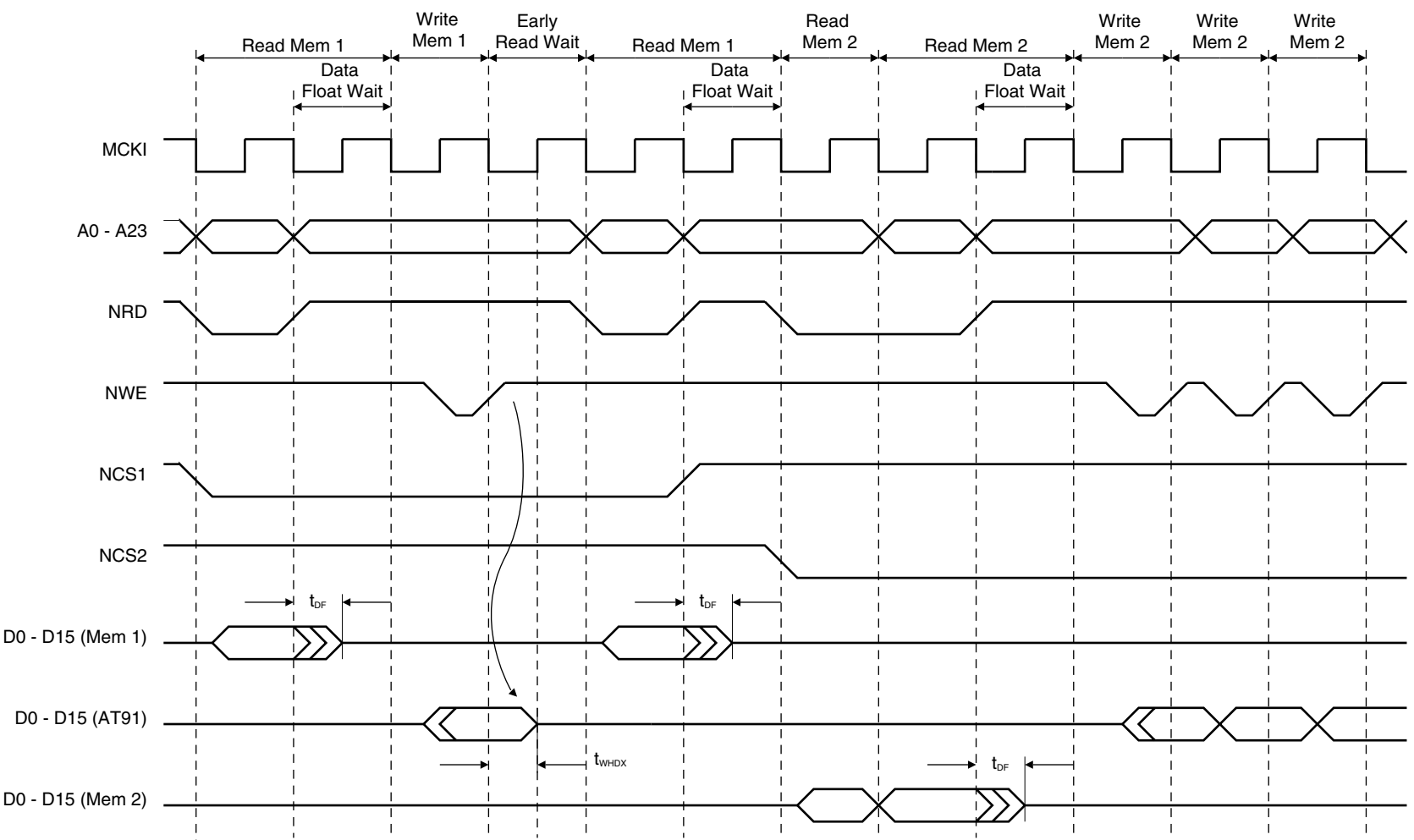


Figure 26. Early Read Protocol with t_{DF}



Figures 27 through 33 show the timing cycles and wait states for read and write access to the various AT91M42800A external memory devices. The configurations described are shown in the following table:

Table 8. Memory Access Waveforms

| Figure Number | Number of Wait States | Bus Width | Size of Data Transfer |
|---------------|-----------------------|-----------|-----------------------|
| 27 | 0 | 16 | Word |
| 28 | 1 | 16 | Word |
| 29 | 1 | 16 | Half-word |
| 30 | 0 | 8 | Word |
| 31 | 1 | 8 | Half-word |
| 32 | 1 | 8 | Byte |
| 33 | 0 | 16 | Byte |

Figure 27. 0 Wait States, 16-bit Bus Width, Word Transfer

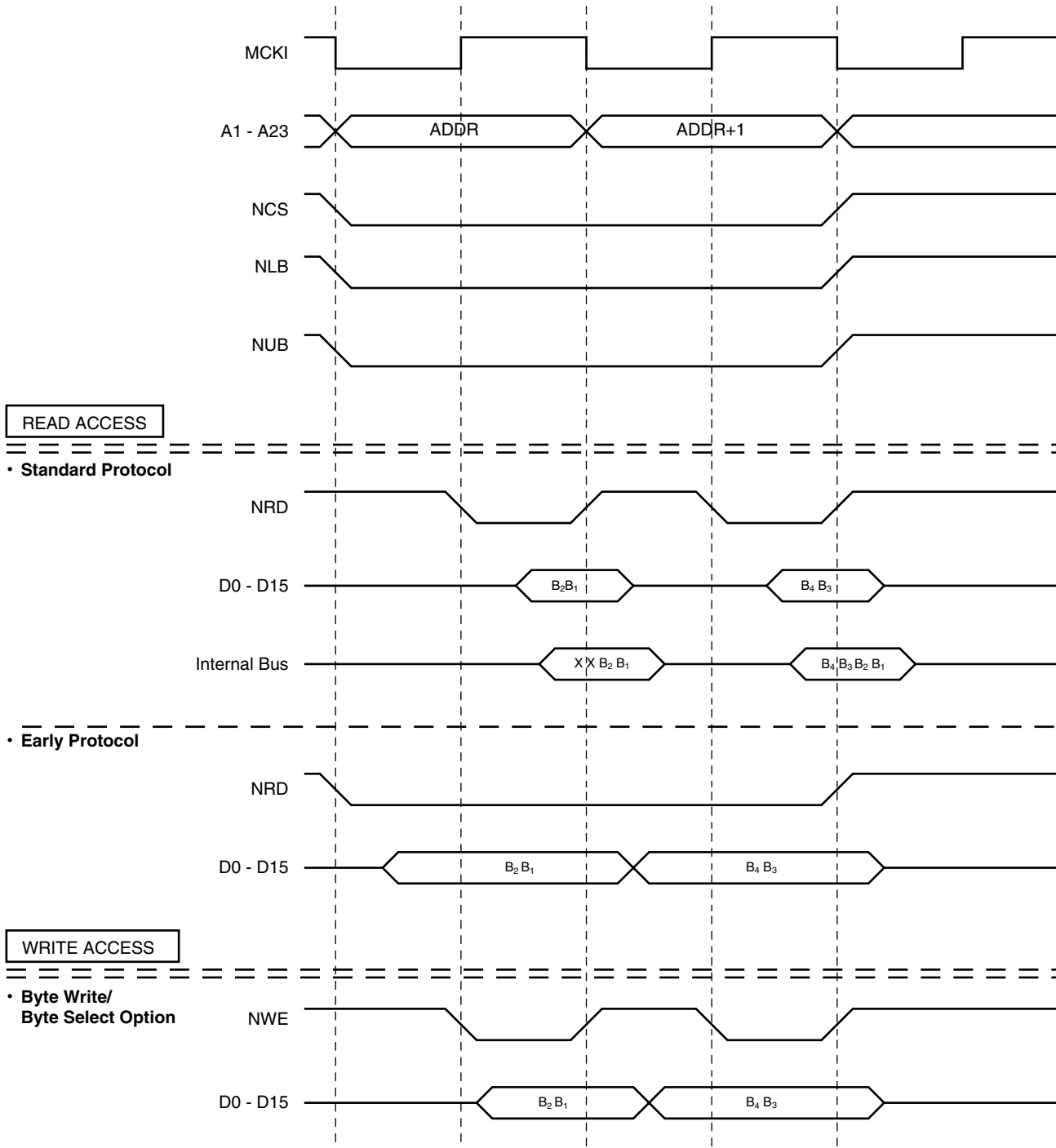


Figure 28. 1 Wait State, 16-bit Bus Width, Word Transfer

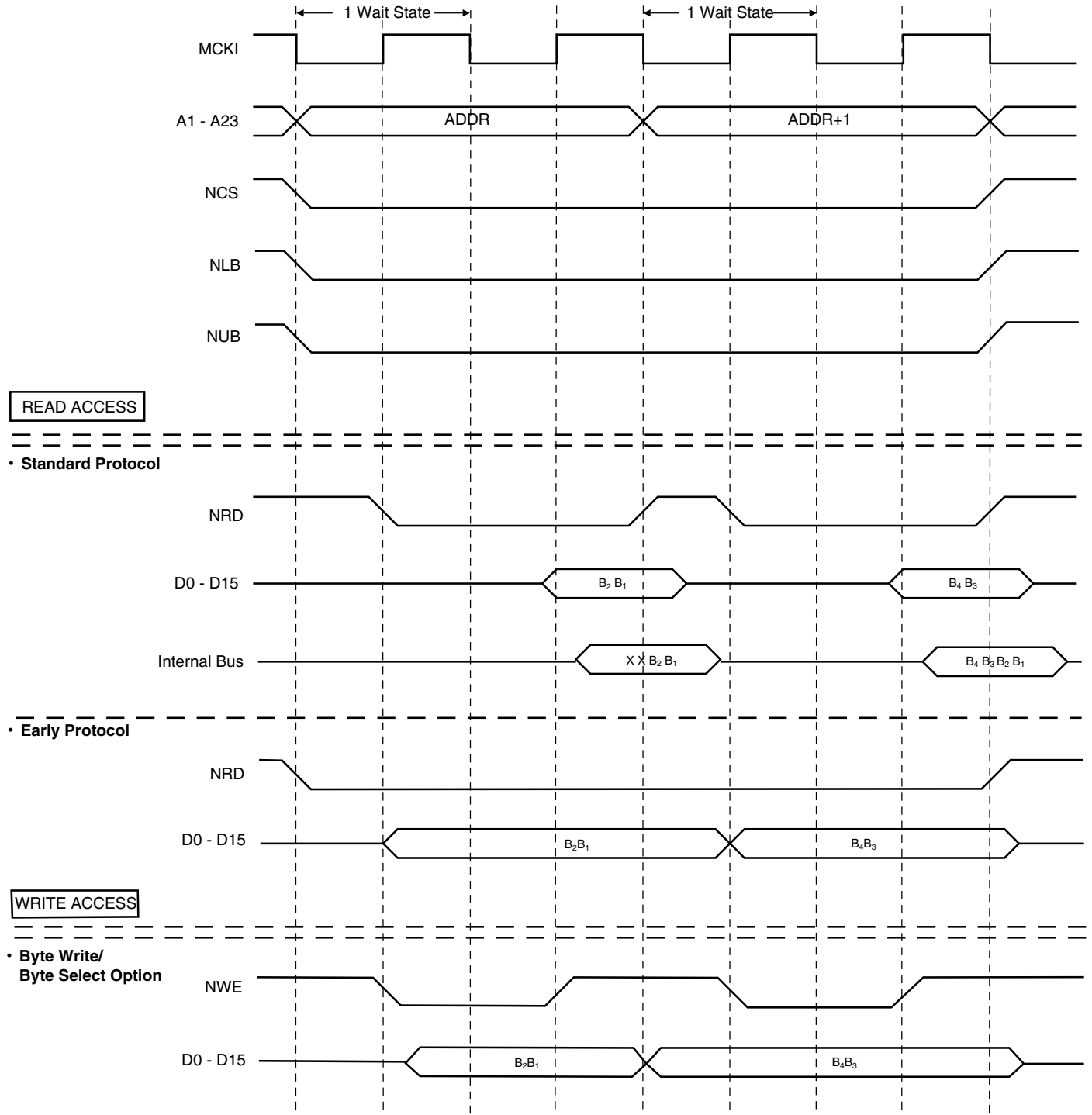


Figure 29. 1 Wait State, 16-bit Bus Width, Half-word Transfer

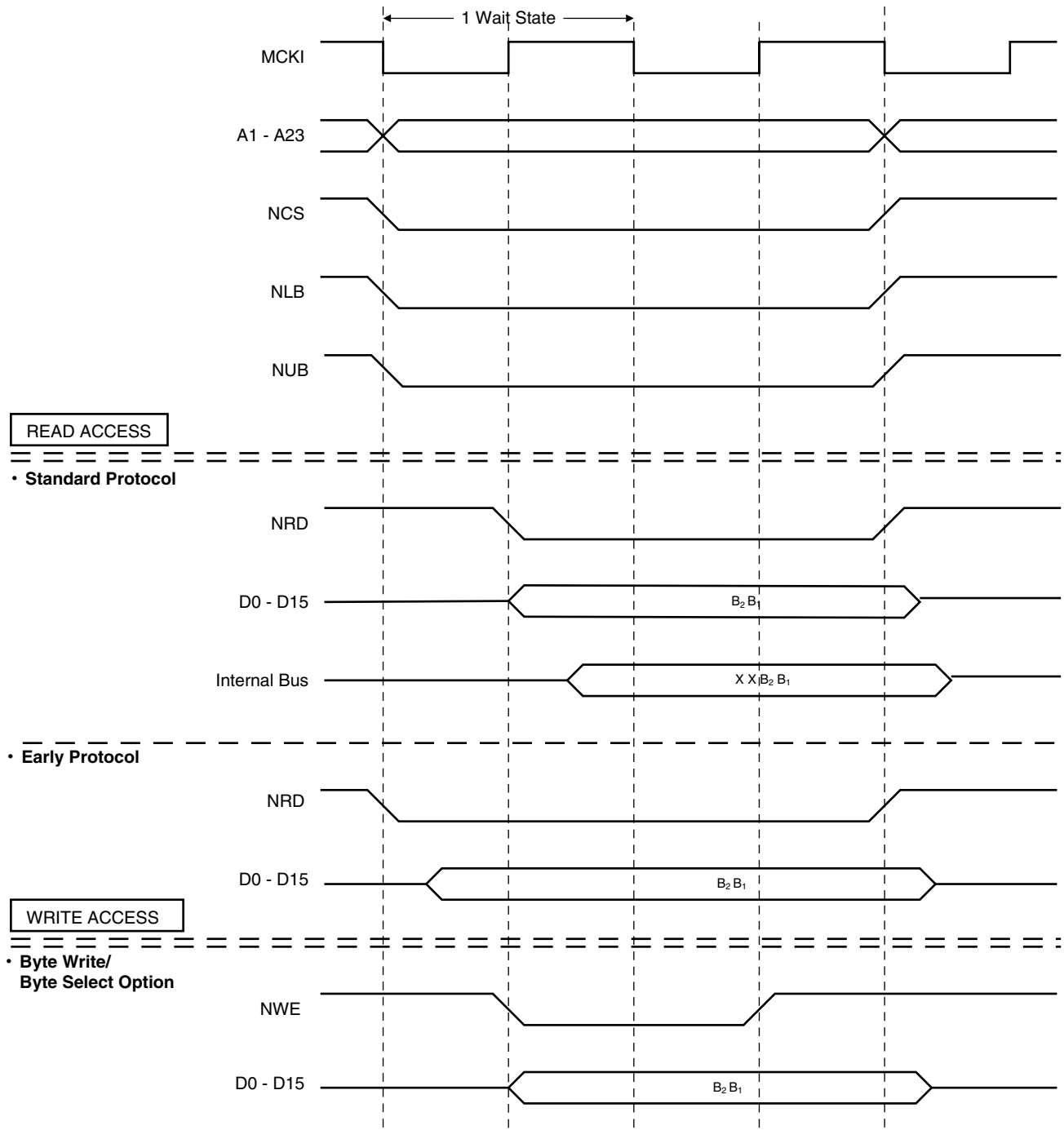


Figure 30. 0 Wait States, 8-bit Bus Width, Word Transfer

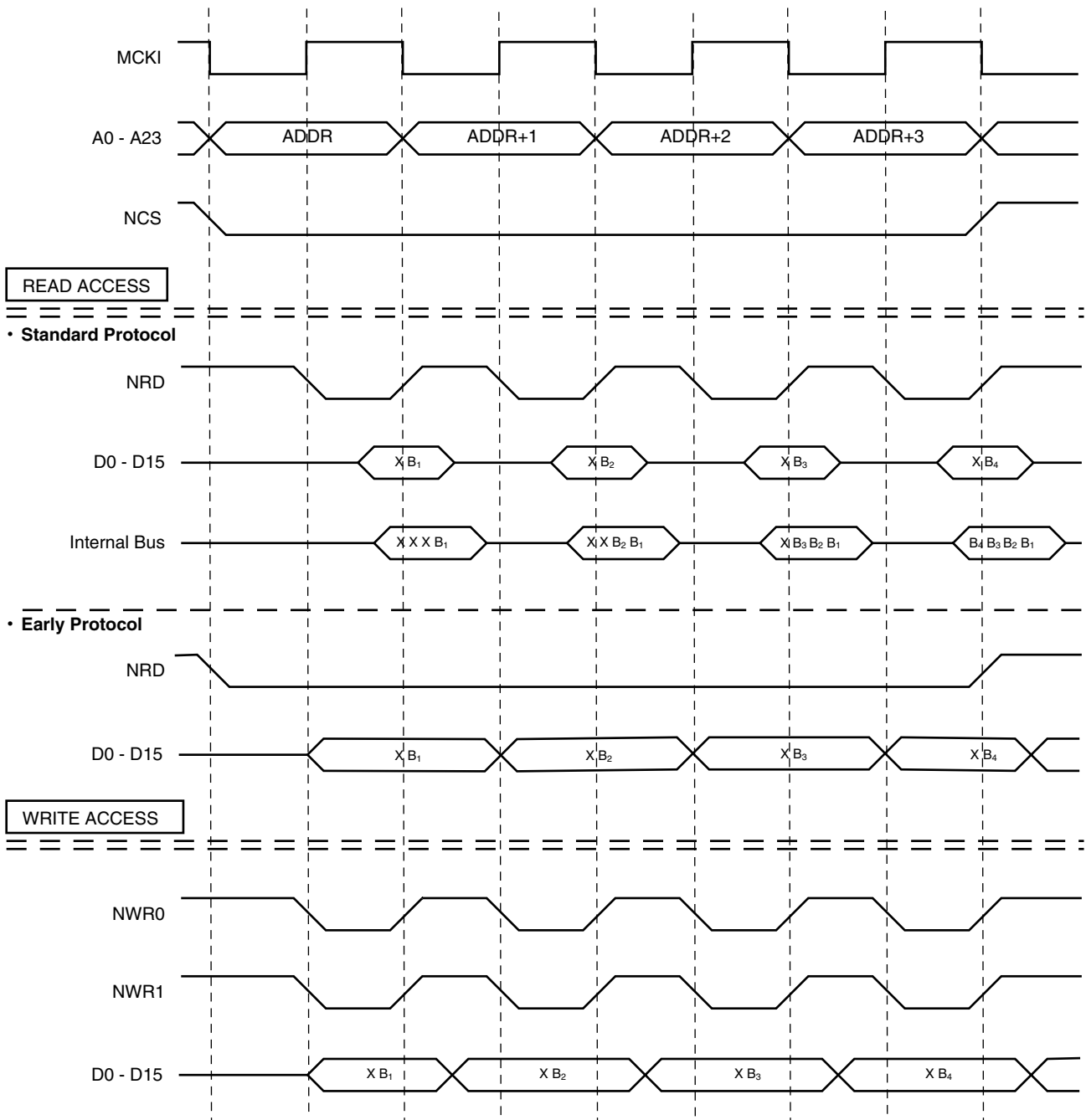


Figure 31. 1 Wait State, 8-bit Bus Width, Half-word Transfer

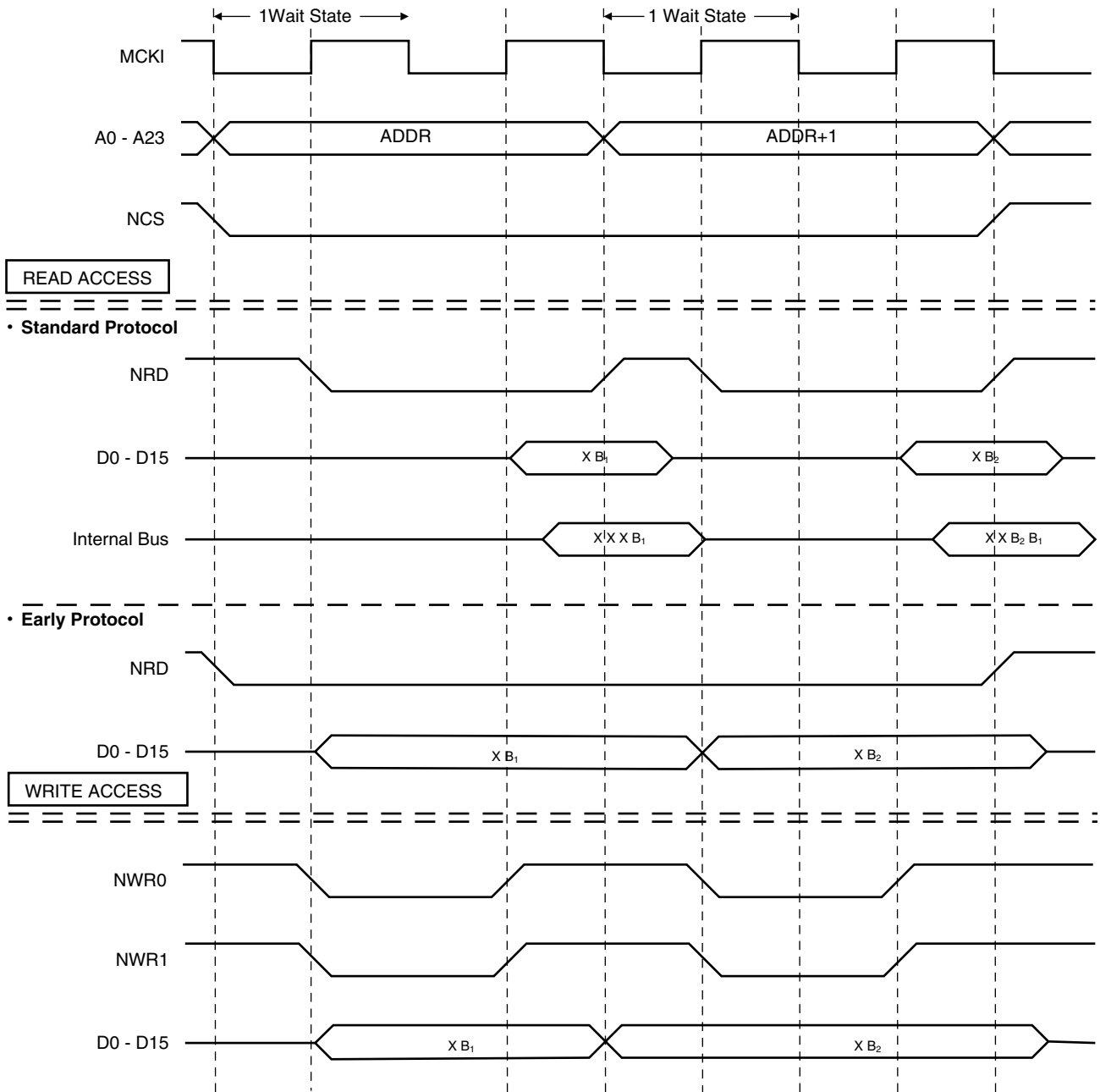


Figure 32. 1 Wait State, 8-bit Bus Width, Byte Transfer

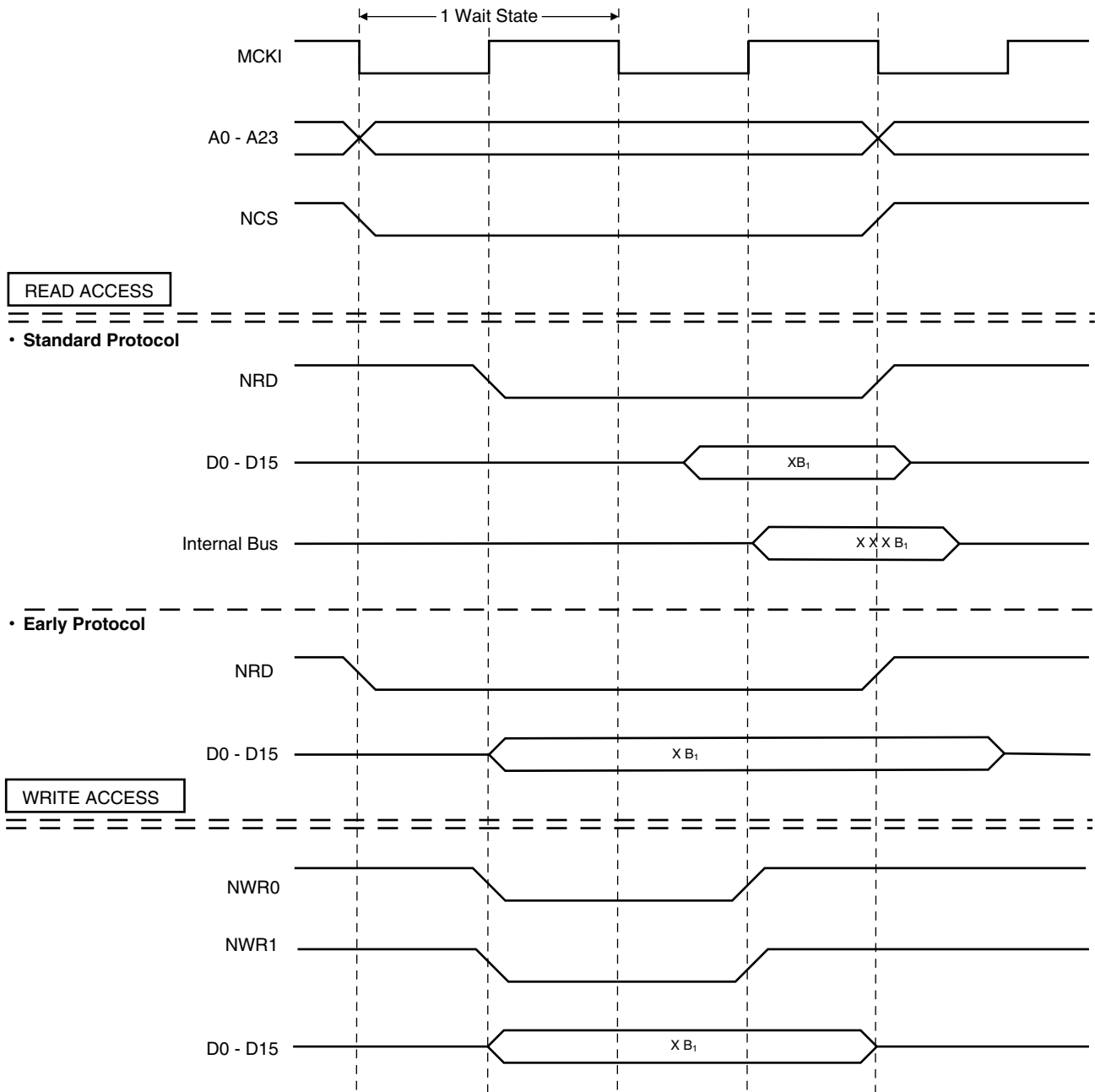
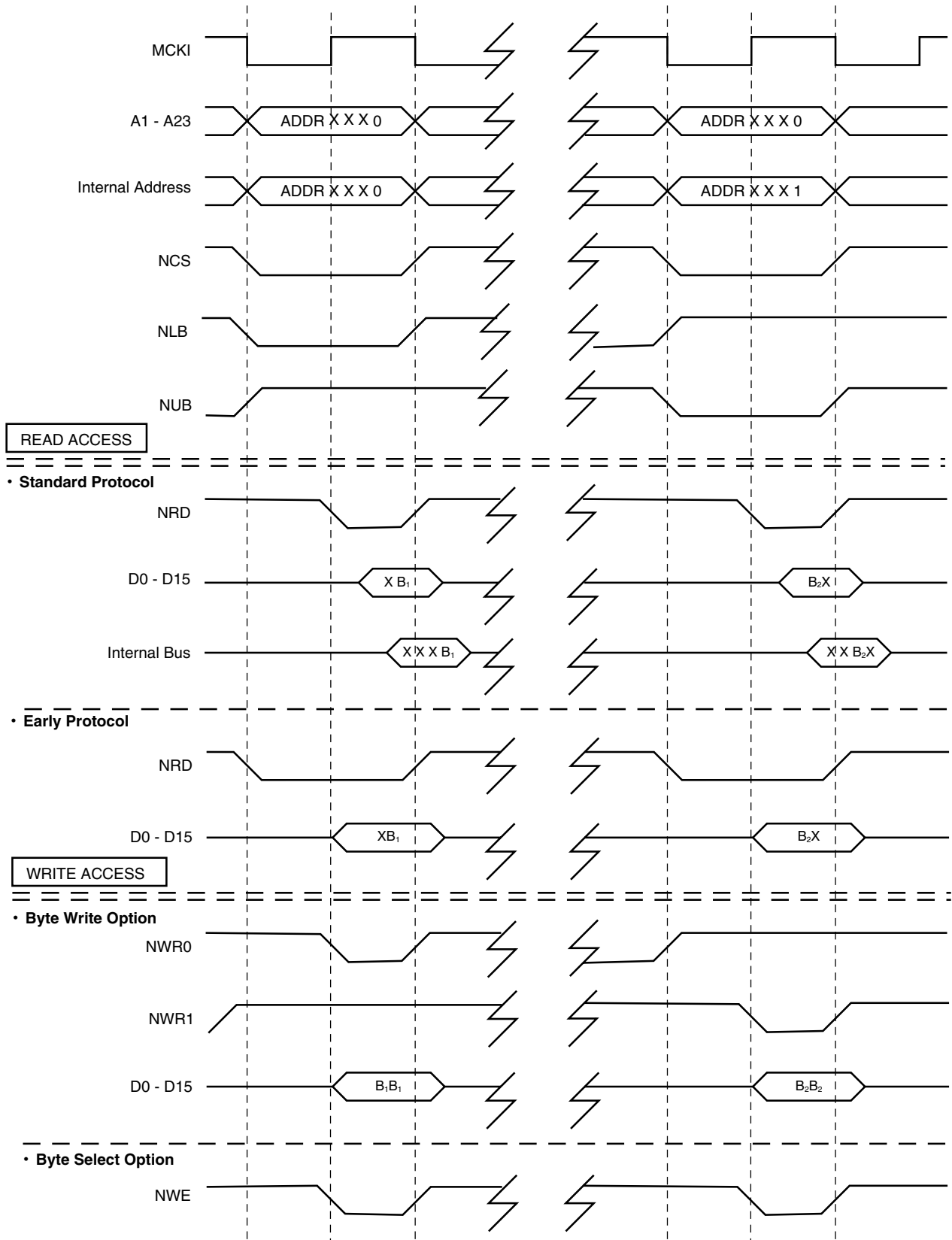


Figure 33. 0 Wait States, 16-bit Bus Width, Byte Transfer



EBI User Interface

The EBI is programmed using the registers listed in Table 9. The Remap Control Register (EBI_RCR) controls exit from Boot mode (see “Boot on NCS0” on page 28). The Memory Control Register (EBI_MCR) is used to program the number of active chip selects and data read protocol. Eight Chip Select Registers (EBI_CSR0 to EBI_CSR7) are used to program the parameters for the individual external memories. Each EBI_CSR must be programmed with a different base address, even for unused chip selects.

The Abort Status registers indicate the access address (EBI_AASR) and the reason for the abort (EBI_ASR).

Base Address: 0xFFE00000 (Code Label EBI_BASE)

Table 9. EBI Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|-------------------------------|----------|------------|--|
| 0x00 | Chip Select Register 0 | EBI_CSR0 | Read/Write | 0x0000203E ⁽¹⁾ 0x0000203D ⁽²⁾ |
| 0x04 | Chip Select Register 1 | EBI_CSR1 | Read/Write | 0x10000000 |
| 0x08 | Chip Select Register 2 | EBI_CSR2 | Read/Write | 0x20000000 |
| 0x0C | Chip Select Register 3 | EBI_CSR3 | Read/Write | 0x30000000 |
| 0x10 | Chip Select Register 4 | EBI_CSR4 | Read/Write | 0x40000000 |
| 0x14 | Chip Select Register 5 | EBI_CSR5 | Read/Write | 0x50000000 |
| 0x18 | Chip Select Register 6 | EBI_CSR6 | Read/Write | 0x60000000 |
| 0x1C | Chip Select Register 7 | EBI_CSR7 | Read/Write | 0x70000000 |
| 0x20 | Remap Control Register | EBI_RCR | Write-only | – |
| 0x24 | Memory Control Register | EBI_MCR | Read/Write | 0 |
| 0x28 | Reserved | – | – | – |
| 0x2C | Reserved | – | – | – |
| 0x30 | Abort Status Register | EBI_ASR | Read-only | 0x0 |
| 0x34 | Address Abort Status Register | EBI_AASR | Read-only | 0x0 |

Notes: 1. 8-bit boot (if BMS is detected high)
2. 16-bit boot (if BMS is detected low)

EBI Chip Select Register

Register Name: EBI_CSR0 - EBI_CSR7
Access Type: Read/Write
Reset Value: See Table 9
Absolute Address: 0xFFE00000 - 0xFFE0001C

| | | | | | | | |
|-------|----|------|-----|-----|-----|-------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| BA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BA | | | | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | CSEN | BAT | TDF | | PAGES | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAGES | – | WSE | NWS | | DBW | | |

• **DBW: Data Bus Width**

| DBW | | Data Bus Width | Code Label: EBI_DBW |
|-----|---|-----------------------|----------------------------|
| 0 | 0 | Reserved | – |
| 0 | 1 | 16-bit data bus width | EBI_DBW_16 |
| 1 | 0 | 8-bit data bus width | EBI_DBW_8 |
| 1 | 1 | Reserved | – |

• **NWS: Number of Wait States**

This field is valid only if WSE is set.

| NWS | | | Number of Standard Wait States | Code Label: EBI_NWS |
|-----|---|---|--------------------------------|----------------------------|
| 0 | 0 | 0 | 1 | EBI_NWS_1 |
| 0 | 0 | 1 | 2 | EBI_NWS_2 |
| 0 | 1 | 0 | 3 | EBI_NWS_3 |
| 0 | 1 | 1 | 4 | EBI_NWS_4 |
| 1 | 0 | 0 | 5 | EBI_NWS_5 |
| 1 | 0 | 1 | 6 | EBI_NWS_6 |
| 1 | 1 | 0 | 7 | EBI_NWS_7 |
| 1 | 1 | 1 | 8 | EBI_NWS_8 |

• **WSE: Wait State Enable (Code Label **EBI_WSE**)**

0 = Wait state generation is disabled. No wait states are inserted.
 1 = Wait state generation is enabled.

- **PAGES: Page Size**

| PAGES | | Page Size | Active Bits in Base Address | Code Label: EBI_PAGES |
|-------|---|-----------|-----------------------------|------------------------------|
| 0 | 0 | 1M Byte | 12 Bits (31 - 20) | EBI_PAGES_1M |
| 0 | 1 | 4M Bytes | 10 Bits (31 - 22) | EBI_PAGES_4M |
| 1 | 0 | 16M Bytes | 8 Bits (31 - 24) | EBI_PAGES_16M |
| 1 | 1 | 64M Bytes | 6 Bits (31 - 26) | EBI_PAGES_64M |

- **TDF: Data Float Output Time**

| TDF | | | Number of Cycles Added after the Transfer | Code Label: EBI_TDF |
|-----|---|---|---|----------------------------|
| 0 | 0 | 0 | 0 | EBI_TDF_0 |
| 0 | 0 | 1 | 1 | EBI_TDF_1 |
| 0 | 1 | 0 | 2 | EBI_TDF_2 |
| 0 | 1 | 1 | 3 | EBI_TDF_3 |
| 1 | 0 | 0 | 4 | EBI_TDF_4 |
| 1 | 0 | 1 | 5 | EBI_TDF_5 |
| 1 | 1 | 0 | 6 | EBI_TDF_6 |
| 1 | 1 | 1 | 7 | EBI_TDF_7 |

- **BAT: Byte Access Type**

| BAT | Selected BAT | Code Label: EBI_BAT |
|-----|-------------------------|----------------------------|
| 0 | Byte-write access type | EBI_BAT_BYTE_WRITE |
| 1 | Byte-select access type | EBI_BAT_BYTE_SELECT |

- **CSEN: Chip Select Enable (Code Label **EBI_CSEN**)**

0 = Chip select is disabled.

1 = Chip select is enabled.

- **BA: Base Address (Code Label **EBI_BA**)**

These bits contain the highest bits of the base address. If the page size is larger than 1M byte, the unused bits of the base address are ignored by the EBI decoder.

EBI Remap Control Register

Register Name: EBI_RCR
Access Type: Write-only
Absolute Address: 0xFFE00020

| | | | | | | | |
|----|----|----|----|----|----|----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | RCB |

- **RCB: Remap Command Bit (Code Label EBI_RCB)**

0 = No effect.

1 = Cancels the remapping (performed at reset) of the page zero memory devices.

EBI Memory Control Register

Register Name: EBI_MCR
Access Type: Read/Write
Reset Value: See Table 9
Absolute Address: 0xFFE00024

| | | | | | | | |
|----|----|----|-----|----|----|----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | DRP | – | – | – | ALE |

- **ALE: Address Line Enable**

This field determines the number of valid address lines and the number of valid chip select lines.

| ALE | | | Valid Address Bits | Maximum Addressable Space | Valid Chip Select | Code Label: EBI_ALE |
|-----|---|---|--------------------|---------------------------|--------------------|---------------------|
| 0 | X | X | A20, A21, A22, A23 | 16M Bytes | None | EBI_ALE_16M |
| 1 | 0 | 0 | A20, A21, A22 | 8M Bytes | CS4 | EBI_ALE_8M |
| 1 | 0 | 1 | A20, A21 | 4M Bytes | CS4, CS5 | EBI_ALE_4M |
| 1 | 1 | 0 | A20 | 2M Bytes | CS4, CS5, CS6 | EBI_ALE_2M |
| 1 | 1 | 1 | None | 1M Byte | CS4, CS5, CS6, CS7 | EBI_ALE_1M |



- **DRP: Data Read Protocol**

| DRP | Selected DRP | Code Label: EBI_DRP |
|------------|--|----------------------------|
| 0 | Standard read protocol for all external memory devices enabled | EBI_DRP_STANDARD |
| 1 | Early read protocol for all external memory devices enabled | EBI_DRP_EARLY |



Abort Status Register

Register Name: EBI_ASR
Access Type: Read-only
Offset: 0x30
Reset Value: 0x0

| | | | | | | | |
|----|----|-----|-----|--------|----|--------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | PDC | ARM | ABTTYP | | ABTSZ | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | MISADD | UNDADD |

- **UNDADD: Undefined Address Abort Status**

0 = The last abort is not due to the access of an undefined address in the EBI address space.
 1 = The last abort is due to the access of an undefined address in the EBI address space.

- **MISADD: Misaligned Address Abort Status**

0 = During the last aborted access, the address required by the core was not unaligned.
 1 = During the last aborted access, the address required by the core was unaligned.

- **ABTSZ: Abort Size Status**

This bit provides the size of the aborted access required by the core.

| ABTSZ | | Abort Size |
|-------|---|------------|
| 0 | 0 | Byte |
| 0 | 1 | Half-word |
| 1 | 0 | Word |
| 1 | 1 | Reserved |

- **ABTTYP: Abort Type Status**

This bit provides the type of the aborted access required by the core.

| ABTTYP | | Abort Size |
|--------|---|------------|
| 0 | 0 | Data read |
| 0 | 1 | Data write |
| 1 | 0 | Code fetch |
| 1 | 1 | Reserved |

- **ARM: Abort Induced by the ARM Core**

0 = The last abort is not due to the ARM core.
 1 = The last abort is due to the ARM core.

- **PDC: Abort Induced by the Peripheral Data Controller**

0 = The last abort is not due to the Peripheral Data controller.
 1 = The last abort is due to the Peripheral Data controller.



Abort Address Status Register

Register Name: EBI_AASR
Access Type: Read-only
Offset: 0x34
Reset Value: 0x0

| | | | | | | | |
|--------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| ABTADD | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ABTADD | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ABTADD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ABTADD | | | | | | | |

- **ABTADD: Abort Address**

This field contains the address required by the last aborted access.



PMC: Power Management Controller

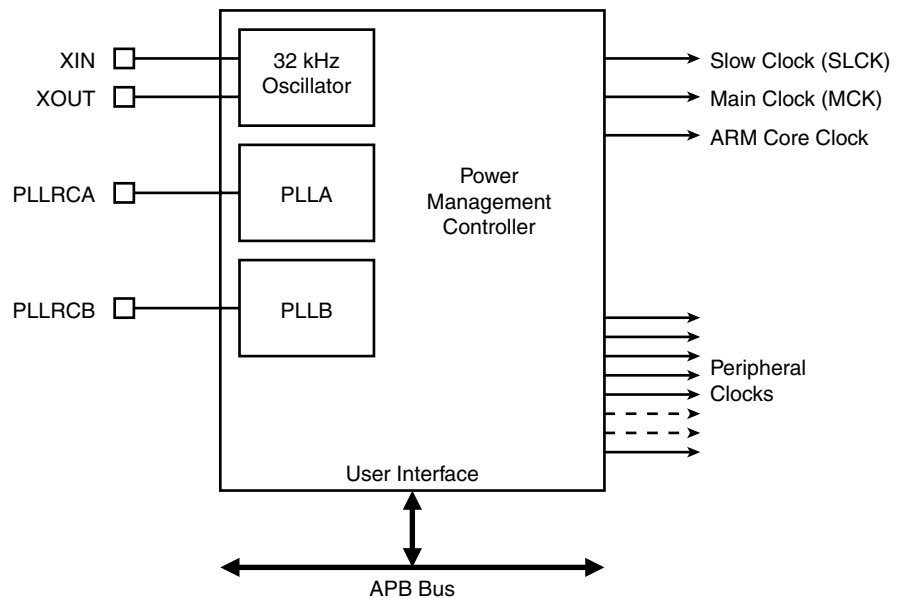
The AT91M42800A’s Power Management Controller optimizes the power consumption of the device. The PMC controls the clocking elements such as the oscillator and the PLLs, and the System and the Peripheral Clocks. It also controls the MCKO pin and enables the user to select four different signals to be driven on this pin.

The AT91M42800A has the following clock elements:

- The oscillator, which provides a clock that depends on the crystal fundamental frequency connected between the XIN and XOUT pins
- PLL A, which provides a low-to-middle frequency clock range
- PLL B, which provides a middle-to-high frequency range
- The Clock prescaler
- The System Clock controller
- The Peripheral Clock controller
- The Master Clock Output controller

The on-chip low-power oscillator together with the PLL-based frequency multiplier and the prescaler results in a programmable clock between 500 Hz and 66 MHz. It is the responsibility of the user to make sure that the PMC programming does not result in a clock over the acceptable limits.

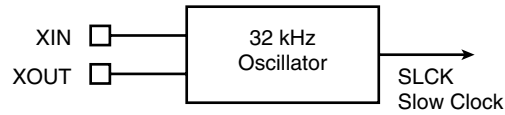
Figure 34. Oscillator, PLL and Clock Sources



Oscillator and Slow Clock

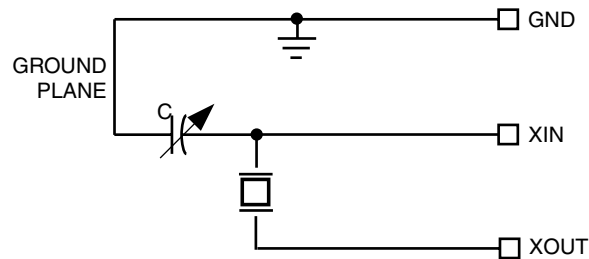
The integrated oscillator generates the Slow Clock. It is designed for use with a 32.768 kHz fundamental crystal. A 38.4 kHz crystal can be used. The bias resistor is on-chip and the oscillator integrates an equivalent load capacitance equal to 10 pF.

Figure 35. Slow Clock



To operate correctly, the crystal must be as close to the XIN and XOUT pins as possible. An external variable capacitor can be added to adjust the oscillator frequency.

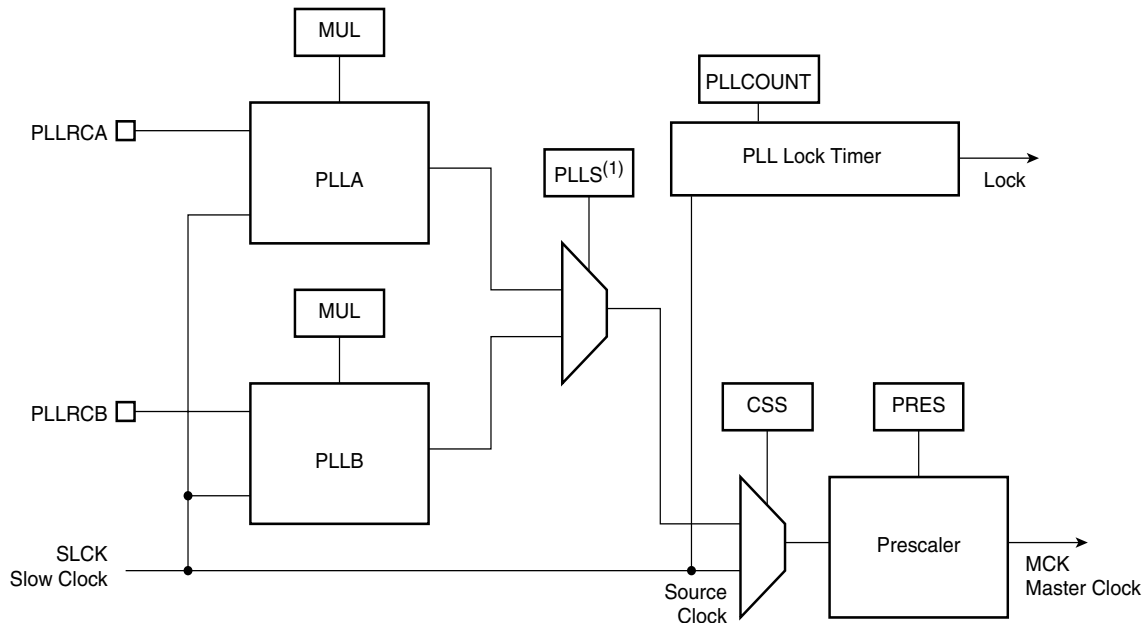
Figure 36. Crystal Location



Master Clock

The Master Clock (MCK) is generated from the Slow Clock by means of one of the two integrated PLLs and the prescaler.

Figure 37. Master Clock

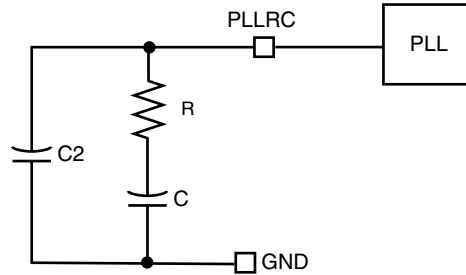


Note: 1. Value written at reset and not subsequently programmable.

Phase Locked Loops

Two PLLs are integrated in the AT91M42800A in order to cover a larger frequency range. Both PLLs have a Slow Clock input and a dedicated pin (PLLRC or PLLRCB), which must have appropriate capacitors and resistors. The capacitors and resistors serve as a second order filter. The PLLRC pin (A or B) that corresponds to the PLL that is disabled may be grounded if capacitors and resistors need to be saved.

Figure 38. PLL Capacitors and Resistors



Typical values for the two PLLs are shown below:

PLLA:

$$F_{SCLK} = 32.768 \text{ kHz}$$

$$F_{out_PLLA} = 16.776 \text{ MHz}$$

$$R = 1600 \text{ Ohm}$$

$$C = 100 \text{ nF}$$

$$C_2 = 10 \text{ nF}$$

With these parameters, the output frequency is stable ($\pm 10\%$) in 600 μs . This settling time is the value to be programmed in the PLLCOUNT field of PMC_CGMR. The maximum frequency overshoot during this phase is 22.5 MHz.

PLL B:

$$F_{SCLK} = 32.768 \text{ kHz}$$

$$F_{out_PLL B} = 33.554 \text{ MHz}$$

$$R = 800 \text{ Ohm}$$

$$C = 1 \mu\text{F}$$

$$C_2 = 100 \text{ nF}$$

With these parameters, the output frequency is stable ($\pm 10\%$) in 4 ms. This settling time is the value to be programmed in the PLLCOUNT field of PMC_CGMR. The maximum frequency overshoot during this phase is 38 MHz.

PLL Selection

The required PLL must be selected at the first writing access and cannot be changed after that. The PLLS bit in PMC_CGMR (Clock Generator Mode Register) determines which PLL module is activated. The other PLL is disabled in order to reduce power consumption and can only be activated by another reset. Writing in PMC_CGMR with a different value has no effect.

Source Clock Selection

The bit CSS in PMC_CGMR selects the Slow Clock or the output of the activated PLL as the Source Clock of the prescaler. After reset, the CSS field is 0, selecting the Slow Clock as Source Clock.

When switching from Slow Clock to PLL Output, the Source Clock takes effect after 3 Slow Clock cycles plus 2.5 PLL output signal cycles. This is a maximum value.

When switching from PLL Output to Slow Clock, the switch takes effect after 3.5 Slow Clock cycles plus 2.5 PLL output signal cycles. This is a maximum value.

PLL Programming

Once the PLL is selected, the output of the active PLL is a multiple of the Slow Clock, determined by the MUL field of the PMC_CGMR. The value of the multiply factor can be up to 2048. The multiplication factor is the programmed value plus one (MUL+1).

Each time PMC_CGMR is written with a MUL value different from the existing one, the LOCK bit in PMC_SR is automatically cleared and the PLL Lock Timer is started (see PLL Lock Timer). The LOCK bit is set when the PLL Lock Timer reaches 0.

If a null value is programmed in the MUL field, the PLL is automatically disabled and bypassed to save power. The LOCK bit in PMC_SR is also automatically cleared.

The time during which the LOCK bit is cleared is user programmable in the field PLLCOUNT in PMC_CGMR. The user must load this parameter with a value depending on the active PLL and its start-up time or the frequency shift performed.

As long as the LOCK bit is 0, the PLL is automatically bypassed and its output is the Slow Clock. This means:

- A switch from the PLL output to the Slow Clock and the associated delays, when the PLL is locked.
- A switch from the Slow Clock to the PLL output and the associated delays, when the LOCK bit is set.

PLL Lock Timer

The Power Management Controller of the AT91M42800A integrates a dedicated 8-bit timer for the locking time of the PLL. This timer is loaded with the value written in PLLCOUNT each time the value in the field MUL changes. At the same time, the LOCK bit in PMC_SR is cleared, and the PLL is bypassed.

The timer counts down the value written in PLLCOUNT on the Slow Clock. The count-down value ranges from 30 μ s to 7.8 ms.

When the PLL Lock Timer reaches 0, the LOCK bit is set and can provide an interrupt.

The PLLCOUNT field is defined by the user, and depends on the current state of the PLL (unlocked or locked), the targeted output frequency and the filter implemented on the PLLRC pin.

Prescaler

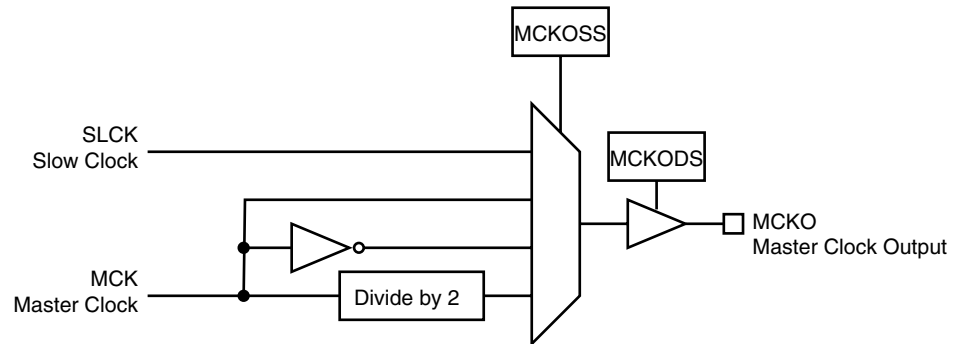
The Clock Source (Slow Clock or PLL output) selected through the CSS field (Clock Source Select) in PMC_CGMR can be divided by 1, 2, 4, 8, 16, 32 or 64. The default divider after a reset is 1. The output of the prescaler is called Master Clock (MCK).

When the prescaler value is modified, the new defined Master Clock is effective after a maximum delay of 64 Source Clock cycles.

Master Clock Output Controller

The clock output on MCKO pin can be selected to be the Slow Clock, the Master Clock, the Master Clock inverted or the Master Clock divided by two through the MCKOSS field (Master Clock Output Source Select) in PMC_CGMR. The MCKO pad can be put in Tri-state mode to save power consumption by setting the bit MCKODS (Master Clock Output Disable) in PMC_CGMR. After a reset the MCKO pin is enabled and is driven by the Slow Clock.

Figure 39. Master Clock Output



ARM Processor Clock Controller

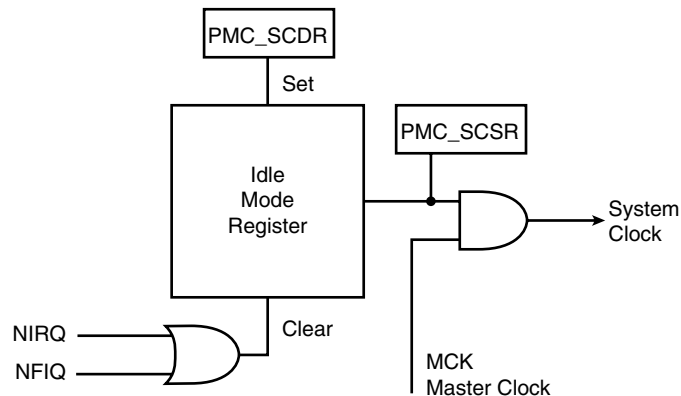
The AT91M42800A has only one System Clock. It can be enabled and disabled by writing the System Clock Enable (PMC_SCER) and System Clock Disable Registers (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

The system clock is enabled after a reset and is automatically re-enabled by any enabled interrupt.

When the system clock is disabled, the current instruction is finished before the clock is stopped.

Note: Stopping the ARM core does not prevent PDC transfers.

Figure 40. System Clock Control



Peripheral Clock Controller

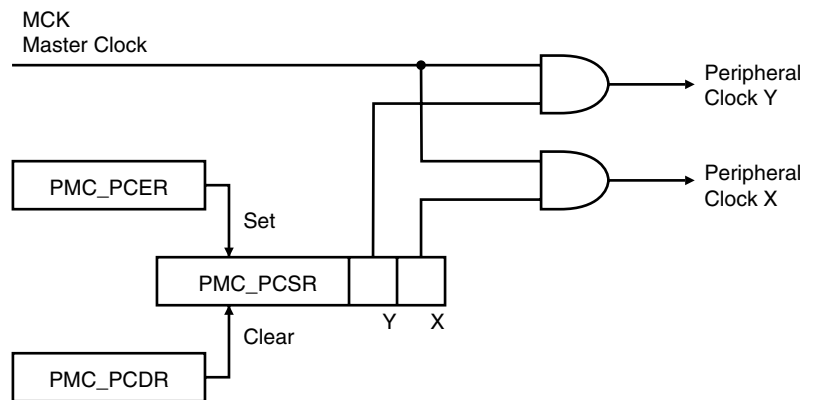
The clock of each peripheral integrated in the AT91M42800A can be individually enabled and disabled by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. When the clock is re-enabled, the peripheral resumes action where it left off. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software waits until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

Note: The bits defined to control the Peripheral Clocks correspond to the bits controlling the Interrupt Sources in the Interrupt Controller.

Figure 41. Peripheral Clock Control



PMC User Interface

Base Address: 0xFFFF4000 (Code Label PMC_BASE)

Table 10. PMC Registers

| Offset | Register Name | Register Mnemonic | Access | Reset Value |
|--------|-----------------------------------|-------------------|------------|-------------|
| 0x00 | System Clock Enable Register | PMC_SCER | Write-only | – |
| 0x04 | System Clock Disable Register | PMC_SCDR | Write-only | – |
| 0x08 | System Clock Status Register | PMC_SCSR | Read-only | 0x00000001 |
| 0x0C | Reserved | – | – | – |
| 0x10 | Peripheral Clock Enable Register | PMC_PCER | Write-only | – |
| 0x14 | Peripheral Clock Disable Register | PMC_PCDR | Write-only | – |
| 0x18 | Peripheral Clock Status Register | PMC_PCSR | Read-only | 0x00000000 |
| 0x1C | Reserved | – | – | – |
| 0x20 | Clock Generator Mode Register | PMC_CGMR | Read/Write | 0x00000000 |
| 0x24 | Reserved | – | – | – |
| 0x28 | Reserved | – | – | – |
| 0x2C | Reserved | – | – | – |
| 0x30 | Status Register | PMC_SR | Read-only | 0x00000000 |
| 0x34 | Interrupt Enable Register | PMC_IER | Write-only | – |
| 0x38 | Interrupt Disable Register | PMC_IDR | Write-only | – |
| 0x3C | Interrupt Mask Register | PMC_IMR | Read-only | 0x00000000 |

PMC System Clock Enable Register

Register Name: PMC_SCER
 Access Type: Write-only
 Offset: 0x00

| | | | | | | | |
|----|----|----|----|----|----|----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | CPU |

- CPU: System Clock Enable

0 = No effect.
 1 = Enables the System Clock.

PMC System Clock Disable Register

Register Name: PMC_SCDR
 Access Type: Write-only
 Offset: 0x04

| | | | | | | | |
|----|----|----|----|----|----|----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | CPU |

- CPU: System Clock Disable

0 = No effect.
 1 = Disables the System Clock.

PMC System Clock Status Register

Register Name: PMC_SCSR
Access Type: Read-only
Offset: 0x08
Reset Value: 0x01

| | | | | | | | |
|----|----|----|----|----|----|----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | CPU |

• **CPU: System Clock Status**

0 = System Clock is disabled.
 1 = System Clock is enabled.

PMC Peripheral Clock Enable Register

Register Name: PMC_PCER
Access Type: Write-only
Offset: 0x10

| | | | | | | | |
|-----|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | PIOB | PIOA | - | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | - | - |

• **Peripheral Clock Enable**

0 = No effect.
 1 = Enables the peripheral clock.

PMC Peripheral Clock Disable Register

Register Name: PMC_PCDR
Access Type: Write-only
Offset: 0x14

| | | | | | | | |
|-----|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | PIOB | PIOA | – | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | – | – |

- **Peripheral Clock Disable**

- 0 = No effect.
- 1 = Disables the peripheral clock.

PMC Peripheral Clock Status Register

Register Name: PMC_PCSR
Access Type: Read-only
Offset: 0x1C
Reset Value: 0x0

| | | | | | | | |
|-----|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | PIOB | PIOA | – | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | – | – |

- **Peripheral Clock Status**

- 0 = Peripheral clock is disabled.
- 1 = Peripheral clock is enabled.

PMC Clock Generator Mode Register

Register Name: PMC_CGMR
Access Type: Read/Write
Offset: 0x20
Reset Value: 0x0

| | | | | | | | |
|----------|--------|--------|----|------|------|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| PLLCOUNT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | MUL | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CSS | MCKODS | MCKOSS | | PLLS | PRES | | |

• PRES: Prescaler Selection

| PRES | | | Prescaler Selected | Code Label <code>PMC_PRES</code> |
|------|---|---|----------------------------------|----------------------------------|
| 0 | 0 | 0 | None. The Prescaler is bypassed. | <code>PMC_PRES_NONE</code> |
| 0 | 0 | 1 | Divide by 2 | <code>PMC_PRES_DIV2</code> |
| 0 | 1 | 0 | Divide by 4 | <code>PMC_PRES_DIV4</code> |
| 0 | 1 | 1 | Divide by 8 | <code>PMC_PRES_DIV8</code> |
| 1 | 0 | 0 | Divide by 16 | <code>PMC_PRES_DIV16</code> |
| 1 | 0 | 1 | Divide by 32 | <code>PMC_PRES_DIV32</code> |
| 1 | 1 | 0 | Divide by 64 | <code>PMC_PRES_DIV64</code> |
| 1 | 1 | 1 | Reserved | - |

• PLLS: PLL Selection

0 = The PLL A with 5 - 20 MHz output range is selected as PLL source. (Code Label `PMC_PLL_A`)

• 1 = The PLL B with 20 - 80 MHz output range is selected as PLL source. (Code Label `PMC_PLL_B`)

Note: This bit can be written only once after the reset. Any write of a different value than this one written the first time has no effect on the bit.

• MCKOSS: Master Clock Output Source Selection

| MCKOSS | | Master Clock Output Source Select | Code Label: <code>PMC_MCKOSS</code> |
|--------|---|-----------------------------------|-------------------------------------|
| 0 | 0 | Slow Clock | <code>PMC_MCKOSS_SLCK</code> |
| 0 | 1 | Master Clock | <code>PMC_MCKOSS_MCK</code> |
| 1 | 0 | Master Clock inverted | <code>PMC_MCKOSS_MCKINV</code> |
| 1 | 1 | Master Clock divided by 2 | <code>PMC_MCKOSS_MCK_DIV2</code> |

• MCKODS: Master Clock Output Disable (Code Label `PMC_MCKO_DIS`)

0 = The pin MCKO is driven with the clock selected by MCKOSS.

1 = The pin MCKO is tri-stated.

- **CSS: Clock Source Selection**

- 0 = The clock source is the Slow Clock.
- 1 = The clock source is the output of the PLL.

- **MUL: Phase Lock Loop Factor**

- 0 = The PLL is disabled, reducing at the minimum its power consumption.
- 1 up to 2047 = The PLL output is at frequency $(MUL+1) \times$ Slow Clock frequency when the LOCK bit is set.

- **PLLCOUNT: PLL Lock Counter**

Specifies the number of 32,768 Hz clock cycles for the PLL lock timer to count before the PLL is locked, after the PLL is started.

PMC Status Register

Register Name: PMC_SR
Access Type: Read-only
Offset: 0x30
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | LOCK |

- **LOCK: PLL Lock Status**

- 0 = The PLL output signal is not stabilized.
- 1 = The PLL output signal is stabilized.

PMC Interrupt Enable Register

Register Name: PMC_IER
Access Type: Write-only
Offset: 0x34

| | | | | | | | |
|----|----|----|----|----|----|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | LOCK |

- **LOCK: PLL Lock Interrupt Enable**
 0 = No effect.
 1 = Enables the PLL Lock Interrupt.

PMC Interrupt Disable Register

Register Name: PMC_IDR
Access Type: Write-only
Offset: 0x38

| | | | | | | | |
|----|----|----|----|----|----|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | LOCK |

- **LOCK: PLL Lock Interrupt Disable**
 0 = No effect.
 1 = Disables the PLL Lock Interrupt.



PMC Interrupt Mask Register

Register Name: PMC_IMR
Access Type: Read-only
Offset: 0x3C
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | LOCK |

- **LOCK: PLL Lock Interrupt Mask**

0 = The PLL Lock Interrupt is disabled.
 1 = The PLL Lock Interrupt is enabled.

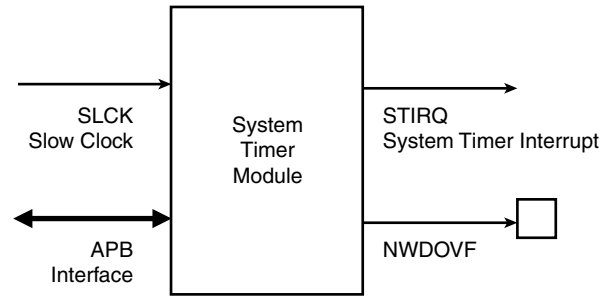
ST: System Timer

The System Timer module integrates three different free-running timers:

- A Period Interval Timer setting the base time for an Operating System.
- A Watchdog Timer having capabilities to reset the system in case of software deadlock.
- A Real-time Timer counting elapsed seconds.

These timers count using the Slow Clock. Typically, this clock has a frequency of 32.768 kHz.

Figure 42. System Timer Module

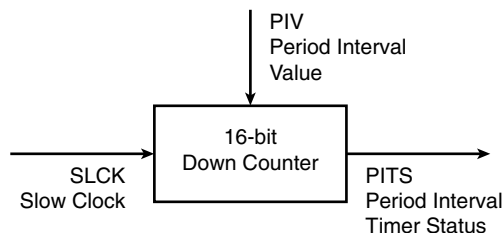


PIT: Period Interval Timer

The Period Interval Timer can be used to provide periodic interrupts for use by operating systems. It is built around a 16-bit down counter, which is preloaded by a value programmed in ST_PIMR (Period Interval Mode Register). When the PIT counter reaches 0, the bit PITS is set in ST_SR (Status Register), and an interrupt is generated, if it is enabled.

The counter is then automatically reloaded and restarted. Writing to the ST_PIMR at any time immediately reloads and restarts the down counter with the new programmed value.

Figure 43. Period Interval Timer



Note: If ST_PIMR is programmed with a period less or equal to the current MCK period, the update of the PITS status bit and its associated interrupt generation are unpredictable.

WDT: Watchdog Timer

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock.

It is built around a 16-bit down counter loaded with the value defined in ST_WDMR (Watchdog Mode Register). It uses the Slow Clock divided by 128. This allows the maximum watchdog period to be 256 seconds (with a typical Slow Clock of 32.768 kHz).

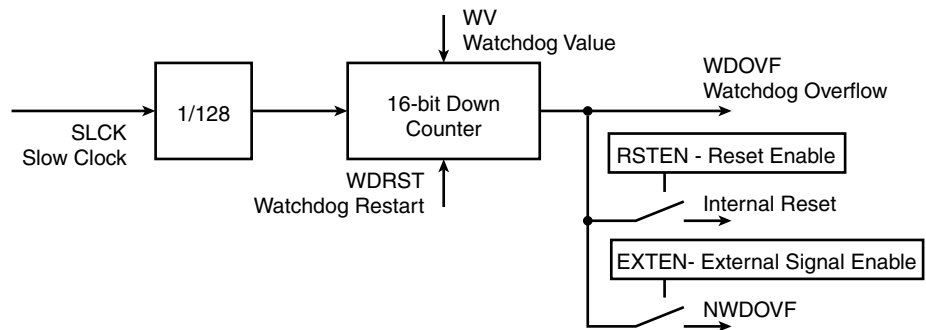
In normal operation, the user reloads the watchdog at regular intervals before the timer overflow occurs. This is done by writing to the ST_CR (Control Register) with the bit WDRST set.

If an overflow does occur, the Watchdog Timer:

- Sets the WDOVF in ST_SR (Status Register) from which an interrupt can be generated
- Generates a pulse for 8 slow clock cycles on the external signal NWDOVF if the bit EXTEN in ST_WDMR is set
- Generates an internal reset if the parameter RSTEN in ST_WDMR is set
- Reloads and restarts the down counter

Writing the ST_WDMR does not reload or restart the down counter. When the ST_CR is written the watchdog is immediately reloaded from ST_WDMR and restarted. The slow clock 128 divider is also immediately reset and restarted. When the ARM7TDMI enters debug mode, the output of the slow clock divider stops, preventing any internal or external reset during the debugging phase.

Figure 44. Watchdog Timer



RTT: Real-time Timer

The Real-time Timer can be used to count elapsed seconds. It is built around a 20-bit counter fed by the Slow Clock divided by a programmable value. At reset this value is set to 0x8000, corresponding to feeding the real-time counter with a 1 Hz signal when the Slow Clock is 32.768 Hz. The 20-bit counter can count up to 1048576 seconds, corresponding to more than 12 days, then roll over to 0.

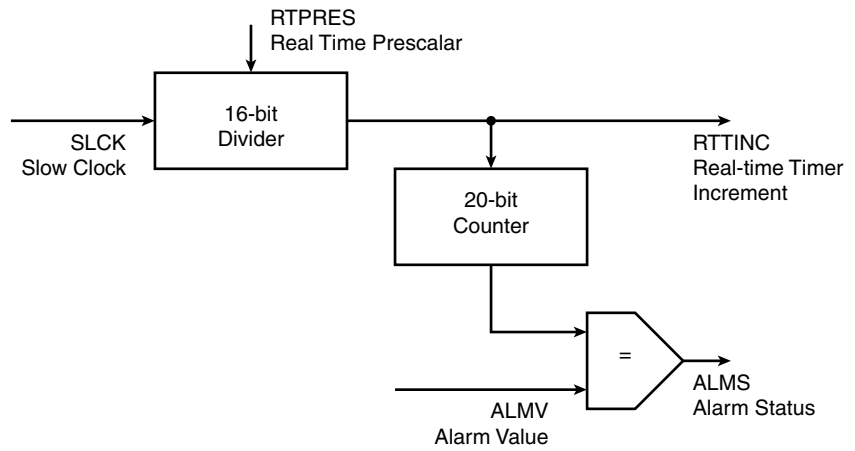
The Real-time Timer value can be read at any time in the register ST_CRTR (Current Real-time Register). As this value can be updated asynchronously to the Master Clock, it is advisable to read this register twice at the same value to improve accuracy of the returned value.

This current value of the counter is compared with the value written in the Alarm Register ST_RTAR (Real-time Alarm Register). If the counter value matches the alarm, the bit ALMS in ST_SR is set. The Alarm Register is set to its maximum value, corresponding to 0, after a reset.

The bit RTTINC in ST_SR is set each time the 20-bit counter is incremented. This bit can be used to start an interrupt, or generate a one-second signal.

Writing the ST_RTMR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 20-bit counter.

Figure 45. Real-time Timer



Note: If RTPRES is programmed with a period less or equal to the current MCK period, the update of the RTTINC and ALMS status bits and their associated interrupt generation are unpredictable.

System Timer User Interface

System Timer Base Address: 0xFFFF8000

Table 11. System Timer Registers

| Offset | Register Name | Register Mnemonic | Access | Reset Value |
|--------|-------------------------------|-------------------|--------|---------------------------|
| 0x00 | Control Register | ST_CR | W | – |
| 0x04 | Period Interval Mode Register | ST_PIMR | R/W | 0x00000000 ⁽¹⁾ |
| 0x08 | Watchdog Mode Register | ST_WDMR | R/W | 0x00020000 ⁽¹⁾ |
| 0x0C | Real-time Mode Register | ST_RTMR | R/W | 0x00008000 |
| 0x10 | Status Register | ST_SR | R | – |
| 0x14 | Interrupt Enable Register | ST_IER | W | – |
| 0x18 | Interrupt Disable Register | ST_IDR | W | – |
| 0x1C | Interrupt Mask Register | ST_IMR | R | 0x0 |
| 0x20 | Real-time Alarm Register | ST_RTAR | R/W | 0x0 |
| 0x24 | Current Real-time Register | ST_CRTR | R | 0x0 |

Note: 1. Corresponds to maximum value of the counter.

System Timer Control Register

Register Name: ST_CR
Access Type: Write-only
Offset: 0x00

| | | | | | | | |
|----|----|----|----|----|----|----|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | WDRST |

• **WDRST: Watchdog Timer Restart**

- 0 = No effect.
- 1 = Reload the start-up value in the Watchdog Timer.

System Timer Period Interval Mode Register

Register Name: ST_PIMR
Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

| | | | | | | | |
|-----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PIV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PIV | | | | | | | |

- **PIV: Period Interval Value**

Defines the value loaded in the 16-bit counter of the Period Interval Timer. The maximum period is obtained by programming PIV at 0x0 corresponding to 65536 Slow Clock cycles.

System Timer Watchdog Mode Register

Register Name: ST_WDMR
Access Type: Read/Write
Offset: 0x08
Reset Value: 0x0002 0000

| | | | | | | | |
|-----|----|----|----|----|----|-------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | EXTEN | RSTEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WDV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDV | | | | | | | |

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 16-bit counter. The maximum period is obtained by programming WDV to 0x0 corresponding to 65536 • 128 Slow Clock cycles.

- **RSTEN: Reset Enable**

0 = No reset is generated when a watchdog overflow occurs.
 1 = An internal reset is generated when a watchdog overflow occurs.

- **EXTEN: External Signal Assertion Enable**

0 = The NWDOVF is not tied low when a watchdog overflow occurs.
 1 = The NWDOVF is tied low during 8 Slow Clock cycles when a watchdog overflow occurs.

System Timer Real-time Mode Register

Register Name: ST_RTMR
Access Type: Read/Write
Offset: 0x0C
Reset Value: 0x0000 8000

| | | | | | | | |
|--------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RTPRES | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTPRES | | | | | | | |

• **RTPRES: Real-time Timer Prescaler Value**

Defines the number of SLCK periods required to increment the Real-time Timer. The maximum period is obtained by programming RTPRES to 0x0 corresponding to 65536 Slow Clock cycles.

System Timer Status Register

Register Name: ST_SR
Access Type: Read-only
Offset: 0x10

| | | | | | | | |
|----|----|----|----|------|--------|-------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | ALMS | RTTINC | WDOVF | PITS |

• **PITS: Period Interval Timer Status**

0 = The Period Interval Timer has not reached 0 since the last read of the Status Register.
 1 = The Period Interval Timer has reached 0 since the last read of the Status Register.

• **WDOVF: Watchdog Overflow**

0 = The Watchdog Timer has not reached 0 since the last read of the Status Register.
 1 = The Watchdog Timer has reached 0 since the last read of the Status Register.

• **RTTINC: Real-time Timer Increment**

0 = The Real-time Timer has not been incremented since the last read of the Status Register.
 1 = The Real-time Timer has been incremented since the last read of the Status Register.

• **ALMS: Alarm Status**

0 = No alarm compare has been detected since the last read of the Status Register.
 1 = Alarm compare has been detected since the last read of the Status Register.



System Timer Interrupt Enable Register

Register Name: ST_IER
Access Type: Write-only
Offset: 0x14

| | | | | | | | |
|----|----|----|----|------|--------|-------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | ALMS | RTTINC | WDOVF | PITS |

- **PITS: Period Interval Timer Status Interrupt Enable**
 0 = No effect.
 1 = Enables the Period Interval Timer Status Interrupt.
- **WDOVF: Watchdog Overflow Interrupt Enable**
 0 = No effect.
 1 = Enables the Watchdog Overflow Interrupt.
- **RTTINC: Real-time Timer Increment Interrupt Enable**
 0 = No effect.
 1 = Enables the Real-time Timer Increment Interrupt.
- **ALMS: Alarm Status Interrupt Enable**
 0 = No effect.
 1 = Enables the Alarm Status Interrupt.

System Timer Interrupt Disable Register

Register Name: ST_IDR
Access Type: Write-only
Offset: 0x18

| | | | | | | | |
|----|----|----|----|------|--------|-------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | ALMS | RTTINC | WDOVF | PITS |

- **PITS: Period Interval Timer Status Interrupt Disable**
 0 = No effect.
 1 = Disables the Period Interval Timer Status Interrupt.
- **WDOVF: Watchdog Overflow Interrupt Disable**
 0 = No effect.
 1 = Disables the Watchdog Overflow Interrupt.
- **RTTINC: Real-time Timer Increment Interrupt Disable**
 0 = No effect.
 1 = Disables the Real-time Timer Increment Interrupt.
- **ALMS: Alarm Status Interrupt Disable**
 0 = No effect.
 1 = Disables the Alarm Status Interrupt.

System Timer Interrupt Mask Register

Register Name: ST_IMR
Access Type: Read-only
Offset: 0x1C
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|------|--------|-------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | ALMS | RTTINC | WDOVF | PITS |

- PITS: Period Interval Timer Status Interrupt Mask**
 0 = Period Interval Timer Status Interrupt is disabled.
 1 = Period Interval Timer Status Interrupt is enabled.
- WDOVF: Watchdog Overflow Interrupt Mask**
 0 = Watchdog Overflow Interrupt is disabled.
 1 = Watchdog Overflow Interrupt is enabled.
- RTTINC: Real-time Timer Increment Interrupt Mask**
 0 = Real-time Timer Increment Interrupt is disabled.
 1 = Real-time Timer Increment Interrupt is enabled.
- ALMS: Alarm Status Interrupt Mask**
 0 = Alarm Status Interrupt is disabled.
 1 = Alarm Status Interrupt is enabled.

System Timer Real-time Alarm Register

Register Name: ST_RTAR
Access Type: Read/Write
Offset: 0x20
Reset Value: 0x0

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ALMV | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ALMV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ALMV | | | | | | | |

- ALMV: Alarm Value**
 Defines the Alarm value compared with the Real-time Timer. The maximum delay before ALMS status bit activation is obtained by programming ALMV to 0x0 corresponding to 1048576 seconds.

System Timer Current Real-time Register

Register Name: ST_CRTR
Access Type: Read-only
Offset: 0x24
Reset Value: 0x0

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRTV | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CRTV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CRTV | | | | | | | |

- **CRTV: Current Real-time Value**
Returns the current value of the Real-time Timer.

AIC: Advanced Interrupt Controller

The AT91M42800A has an 8-level priority, individually maskable, vectored interrupt controller. This feature substantially reduces the software and real-time overhead in handling internal and external interrupts.

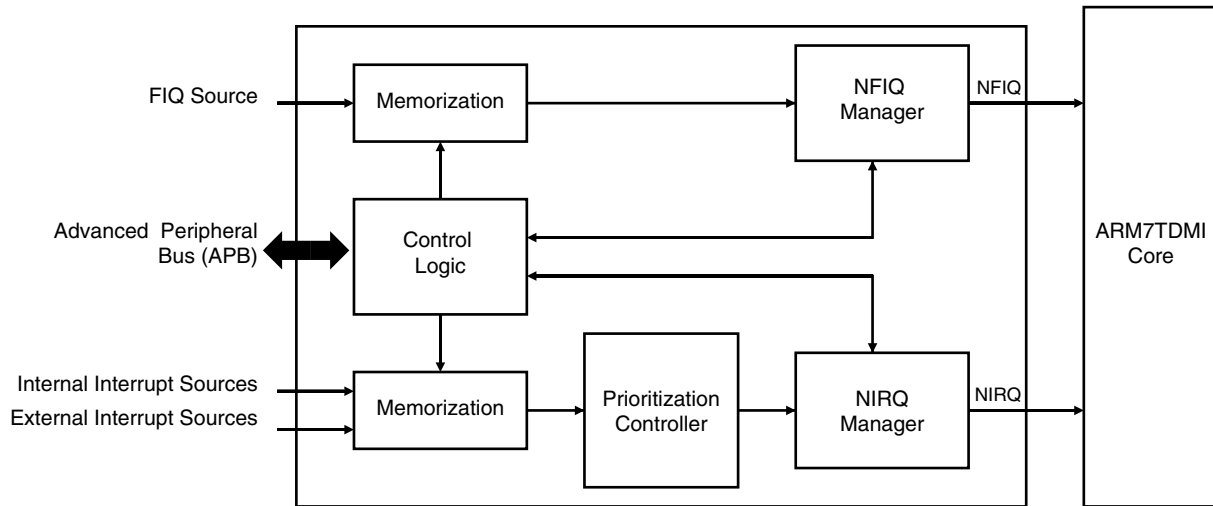
The interrupt controller is connected to the NFIQ (fast interrupt request) and the NIRQ (standard interrupt request) inputs of the ARM7TDMI processor. The processor's NFIQ line can only be asserted by the external fast interrupt request input: FIQ. The NIRQ line can be asserted by the interrupts generated by the on-chip peripherals and the external interrupt request lines: IRQ0 to IRQ3.

The 8-level priority encoder allows the customer to define the priority between the different NIRQ interrupt sources.

Internal sources are programmed to be level sensitive or edge triggered. External sources can be programmed to be positive or negative edge triggered or high- or low-level sensitive.

The interrupt sources are listed in Table 12 and the AIC programmable registers in Table 13.

Figure 46. Interrupt Controller Block Diagram



Note: After a hardware reset, the external interrupt sources pins are controlled by the Controller. They must be configured to be controlled by the peripheral before being used.

Table 12. AIC Interrupt Sources

| Interrupt Source | Interrupt Name | Interrupt Description |
|------------------|----------------|---------------------------------------|
| 0 | FIQ | Fast Interrupt |
| 1 | SW | Soft Interrupt (generated by the AIC) |
| 2 | US0 | USART Channel 0 interrupt |
| 3 | US1 | USART Channel 1 interrupt |
| 4 | SPIA | SPI Channel A Interrupt |
| 5 | SPIB | SPI Channel B Interrupt |
| 6 | TC0 | Timer Channel 0 Interrupt |
| 7 | TC1 | Timer Channel 1 Interrupt |
| 8 | TC2 | Timer Channel 2 Interrupt |
| 9 | TC3 | Timer Channel 3 Interrupt |
| 10 | TC4 | Timer Channel 4 Interrupt |
| 11 | TC5 | Timer Channel 5 Interrupt |
| 12 | ST | System Timer Interrupt |
| 13 | PIOA | Parallel I/O Controller A Interrupt |
| 14 | PIOB | Parallel I/O Controller B Interrupt |
| 15 | PMC | Power Management Controller Interrupt |
| 16 | – | Reserved |
| 17 | – | Reserved |
| 18 | – | Reserved |
| 19 | – | Reserved |
| 20 | – | Reserved |
| 21 | – | Reserved |
| 22 | – | Reserved |
| 23 | – | Reserved |
| 24 | – | Reserved |
| 25 | – | Reserved |
| 26 | – | Reserved |
| 27 | – | Reserved |
| 28 | IRQ3 | External Interrupt 3 |
| 29 | IRQ2 | External Interrupt 2 |
| 30 | IRQ1 | External Interrupt 1 |
| 31 | IRQ0 | External Interrupt 0 |

Hardware Interrupt Vectoring

The hardware interrupt vectoring reduces the number of instructions to reach the interrupt handler to only one. By storing the following instruction at address 0x00000018, the processor loads the program counter with the interrupt handler address stored in the AIC_IVR register. Execution is then vectored to the interrupt handler corresponding to the current interrupt.

```
ldr PC, [PC, # -&F20]
```

The current interrupt is the interrupt with the highest priority when the Interrupt Vector Register (AIC_IVR) is read. The value read in the AIC_IVR corresponds to the address stored in the Source Vector Register (AIC_SVR) of the current interrupt. Each interrupt source has its corresponding AIC_SVR. In order to take advantage of the hardware interrupt vectoring it is necessary to store the address of each interrupt handler in the corresponding AIC_SVR, at system initialization.

Priority Controller

The NIRQ line is controlled by an 8-level priority encoder. Each source has a programmable priority level of 7 to 0. Level 7 is the highest priority and level 0 the lowest.

When the AIC receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the lowest interrupt source number (see Table 12) is serviced first.

The current priority level is defined as the priority level of the current interrupt at the time the register AIC_IVR is read (the interrupt which will be serviced).

In the case when a higher priority unmasked interrupt occurs while an interrupt already exists, there are two possible outcomes depending on whether the AIC_IVR has been read.

- If the NIRQ line has been asserted but the AIC_IVR has not been read, then the processor will read the new higher priority interrupt handler address in the AIC_IVR register and the current interrupt level is updated.
- If the processor has already read the AIC_IVR then the NIRQ line is reasserted. When the processor has authorized nested interrupts to occur and reads the AIC_IVR again, it reads the new, higher priority interrupt handler address. At the same time the current priority value is pushed onto a first-in last-out stack and the current priority is updated to the higher priority.

When the end of interrupt command register (AIC_EOICR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Hence at the end of a higher priority interrupt, the AIC returns to the previous state corresponding to the preceding lower priority interrupt which had been interrupted.

Interrupt Handling

The interrupt handler must read the AIC_IVR as soon as possible. This de-asserts the NIRQ request to the processor and clears the interrupt in case it is programmed to be edge triggered. This permits the AIC to assert the NIRQ line again when a higher priority unmasked interrupt occurs.

At the end of the interrupt service routine, the end of interrupt command register (AIC_EOICR) must be written. This allows pending interrupts to be serviced.

Interrupt Masking

Each interrupt source, including FIQ, can be enabled or disabled using the command registers AIC_IECR and AIC_IDCR. The interrupt mask can be read in the Read-only register AIC_IMR. A disabled interrupt does not affect the servicing of other interrupts.

Interrupt Clearing and Setting

All interrupt sources which are programmed to be edge triggered (including FIQ) can be individually set or cleared by respectively writing to the registers AIC_ISCR and AIC_ICCR. This function of the interrupt controller is available for auto-test or software debug purposes.

Fast Interrupt Request

The external FIQ line is the only source which can raise a fast interrupt request to the processor. Therefore, it has no priority controller.

The external FIQ line can be programmed to be positive or negative edge triggered or high- or low-level sensitive in the AIC_SMR0 register.

The fast interrupt handler address can be stored in the AIC_SVR0 register. The value written into this register is available by reading the AIC_FVR register when an FIQ interrupt is raised. By storing the following instruction at address 0x0000001C, the processor will load the program counter with the interrupt handler address stored in the AIC_FVR register.

```
ldr PC, [PC, # -&F20]
```

Alternatively the interrupt handler can be stored starting from address 0x0000001C as described in the ARM7TDMI datasheet.

Software Interrupt

Interrupt source 1 of the advanced interrupt controller is a software interrupt. It must be programmed to be edge triggered in order to set or clear it by writing to the AIC_ISCR and AIC_ICCR.

This is totally independent of the SWI instruction of the ARM7TDMI processor.

Spurious Interrupt

When the AIC asserts the NIRQ line, the ARM7TDMI enters IRQ mode and the interrupt handler reads the IVR. It may happen that the AIC de-asserts the NIRQ line after the core has taken into account the NIRQ assertion and before the read of the IVR.

This behavior is called a Spurious Interrupt.

The AIC is able to detect these Spurious Interrupts and returns the Spurious Vector when the IVR is read. The Spurious Vector can be programmed by the user when the vector table is initialized.

A spurious interrupt may occur in the following cases:

- With any sources programmed to be level sensitive, if the interrupt signal of the AIC input is de-asserted at the same time as it is taken into account by the ARM7TDMI.
- If an interrupt is asserted at the same time as the software is disabling the corresponding source through AIC_IDCR (this can happen due to the pipelining of the ARM core).

The same mechanism of spurious interrupt occurs if the ARM7TDMI reads the IVR (application software or ICE) when there is no interrupt pending. This mechanism is also valid for the FIQ interrupts.

Once the AIC enters the spurious interrupt management, it asserts neither the NIRQ nor the NFIQ lines to the ARM7TDMI as long as the spurious interrupt is not acknowledged. Therefore, it is mandatory for the Spurious Interrupt Service Routine to acknowledge the “spurious” behavior by writing to the AIC_EOICR (End of Interrupt) before returning to the interrupted software. It also can perform other operation(s), e.g., trace possible undesirable behavior.

Protect Mode

The Protect Mode permits reading of the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system.

When a Debug Monitor or an ICE reads the AIC User Interface, the IVR could be read. This would have the following consequences in normal mode.

- If an enabled interrupt with a higher priority than the current one is pending, it is stacked.
- If there is no enabled pending interrupt, the spurious vector is returned.

In either case, an End of Interrupt command would be necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system. Hence the debug system would become strongly intrusive, and could cause the application to enter an undesired state.

This is avoided by using Protect mode.

The Protect mode is enabled by setting the AIC bit in the SF Protect Mode Register (see “SF: Special Function Registers” on page 116).

When Protect mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it.

The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR.

Extra AIC_IVR reads performed in between the read and the write can cause unpredictable results. Therefore, it is strongly recommended not to set a breakpoint between these two actions, nor to stop the software.

The debug system must not write to the AIC_IVR as this would cause undesirable effects.

The following table shows the main steps of an interrupt and the order in which they are performed according to the mode:

| Action | Normal Mode | Protect Mode |
|--|---------------|---------------|
| Calculate active interrupt (higher than current or spurious) | Read AIC_IVR | Read AIC_IVR |
| Determine and return the vector of the active interrupt | Read AIC_IVR | Read AIC_IVR |
| Memorize interrupt | Read AIC_IVR | Read AIC_IVR |
| Push on internal stack the current priority level | Read AIC_IVR | Write AIC_IVR |
| Acknowledge the interrupt ⁽¹⁾ | Read AIC_IVR | Write AIC_IVR |
| No effect ⁽²⁾ | Write AIC_IVR | – |

- Notes:
1. NIRQ de-assertion and automatic interrupt clearing if the source is programmed as level sensitive.
 2. Software that has been written and debugged using Protect mode will run correctly in Normal mode without modification. However, in Normal mode, the AIC_IVR write has no effect and can be removed to optimize the code.

AIC User Interface

Base Address: 0xFFFFF000 (Code Label AIC_BASE)

Table 13. AIC Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|------------------------------------|-----------|------------|--------------|
| 0x000 | Source Mode Register 0 | AIC_SMR0 | Read/Write | 0 |
| 0x004 | Source Mode Register 1 | AIC_SMR1 | Read/Write | 0 |
| – | – | – | Read/Write | 0 |
| 0x07C | Source Mode Register 31 | AIC_SMR31 | Read/Write | 0 |
| 0x080 | Source Vector Register 0 | AIC_SVR0 | Read/Write | 0 |
| 0x084 | Source Vector Register 1 | AIC_SVR1 | Read/Write | 0 |
| – | – | – | Read/Write | 0 |
| 0x0FC | Source Vector Register 31 | AIC_SVR31 | Read/Write | 0 |
| 0x100 | IRQ Vector Register | AIC_IVR | Read-only | – |
| 0x104 | FIQ Vector Register | AIC_FVR | Read-only | – |
| 0x108 | Interrupt Status Register | AIC_ISR | Read-only | – |
| 0x10C | Interrupt Pending Register | AIC_IPR | Read-only | (see Note 1) |
| 0x110 | Interrupt Mask Register | AIC_IMR | Read-only | 0 |
| 0x114 | Core Interrupt Status Register | AIC_CISR | Read-only | – |
| 0x118 | Reserved | – | – | – |
| 0x11C | Reserved | – | – | – |
| 0x120 | Interrupt Enable Command Register | AIC_IECR | Write-only | – |
| 0x124 | Interrupt Disable Command Register | AIC_IDCR | Write-only | – |
| 0x128 | Interrupt Clear Command Register | AIC_ICCR | Write-only | – |
| 0x12C | Interrupt Set Command Register | AIC_ISCR | Write-only | – |
| 0x130 | End of Interrupt Command Register | AIC_EOICR | Write-only | – |
| 0x134 | Spurious Vector Register | AIC_SPU | Read/Write | 0 |

Note: 1. The reset value of this register depends on the level of the External IRQ lines. All other sources are cleared at reset.

AIC Source Mode Register

Register Name: AIC_SMR0..AIC_SMR31

Access Type: Read/Write

Reset Value: 0

| | | | | | | | |
|----|---------|----|----|----|-------|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | SRCTYPE | | – | – | PRIOR | | |

- **PRIOR: Priority Level (Code Label AIC_PRIOR)**

Program the priority level for all sources except source 0 (FIQ).
The priority level can be between 0 (lowest) and 7 (highest).
The priority level is not used for the FIQ, in the SMR0.

- **SRCTYPE: Interrupt Source Type (Code Label AIC_SRCTYPE)**

Program the input to be positive or negative edge-triggered or positive or negative level sensitive.
The active level or edge is not programmable for the internal sources.

| SRCTYPE | | External Sources | Code Label |
|---------|---|-------------------------|-------------------------------|
| 0 | 0 | Low-level Sensitive | AIC_SRCTYPE_EXT_LOW_LEVEL |
| 0 | 1 | Negative Edge triggered | AIC_SRCTYPE_EXT_NEGATIVE_EDGE |
| 1 | 0 | High-level Sensitive | AIC_SRCTYPE_EXT_HIGH_LEVEL |
| 1 | 1 | Positive Edge triggered | AIC_SRCTYPE_EXT_POSITIVE_EDGE |

| SRCTYPE | | Internal Sources | Code Label |
|---------|---|------------------|---------------------------------|
| X | 0 | Level Sensitive | AIC_SRCTYPE_INT_LEVEL_SENSITIVE |
| X | 1 | Edge triggered | AIC_SRCTYPE_INT_EDGE_TRIGGERED |

AIC Source Vector Register

Register Name: AIC_SVR0..AIC_SVR31

Access Type: Read/Write

Reset Value: 0

| | | | | | | | |
|--------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| VECTOR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VECTOR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VECTOR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VECTOR | | | | | | | |

- **VECTOR: Interrupt Handler Address**

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

AIC Interrupt Vector Register

Register Name: AIC_IVR
Access Type: Read-only
Offset: 0x100

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQV | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IRQV | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IRQV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IRQV | | | | | | | |

- **IRQV: Interrupt Vector Register**

The IRQ Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register (1 to 31) is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the IRQ Vector Register reads the value stored in AIC_SPU.

AIC FIQ Vector Register

Register Name: AIC_FVR
Access Type: Read-only
Offset: 0x104

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FIQV | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FIQV | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FIQV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIQV | | | | | | | |

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0 which corresponds to FIQ.

AIC Interrupt Status Register

Register Name: AIC_ISR
Access Type: Read-only
Offset: 0x108

| | | | | | | | | |
|----|----|----|-------|----|----|----|----|--|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
| - | - | - | - | - | - | - | - | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| - | - | - | - | - | - | - | - | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| - | - | - | - | - | - | - | - | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| - | - | - | IRQID | | | | | |

- **IRQID: Current IRQ Identifier (Code Label AIC_IRQID)**
 The Interrupt Status Register returns the current interrupt source number.

AIC Interrupt Pending Register

Register Name: AIC_IPR
Access Type: Read-only
Offset: 0x10C

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

• Interrupt Pending

0 = Corresponding interrupt is inactive.
 1 = Corresponding interrupt is pending.

AIC Interrupt Mask Register

Register Name: AIC_IMR
Access Type: Read-only
Offset: 0x110
Reset Value: 0x0

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

• Interrupt Mask

0 = Corresponding interrupt is disabled.
 1 = Corresponding interrupt is enabled.

AIC Core Interrupt Status Register

Register Name: AIC_CISR
Access Type: Read-only
Offset: 0x114

| | | | | | | | |
|----|----|----|----|----|----|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | NIRQ | NFIQ |

- **NFIQ: NFIQ Status (Code Label AIC_NFIQ)**

0 = NFIQ line inactive.
 1 = NFIQ line active.

- **NIRQ: NIRQ Status (Code Label AIC_NIRQ)**

0 = NIRQ line inactive.
 1 = NIRQ line active.

AIC Interrupt Enable Command Register

Register Name: AIC_IECR
Access Type: Write-only
Offset: 0x120

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

- **Interrupt Enable**

0 = No effect.
 1 = Enables corresponding interrupt.

AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Access Type: Write-only

Offset: 0x124

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

- Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

AIC Interrupt Clear Command Register

Register Name: AIC_ICCR
Access Type: Write-only
Offset: 0x128

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

• **Interrupt Clear**

0 = No effect.
 1 = Clears corresponding interrupt.

AIC Interrupt Set Command Register

Register Name: AIC_ISCR
Access Type: Write-only
Offset: 0x12C

| | | | | | | | |
|------|------|------|------|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| IRQ0 | IRQ1 | IRQ2 | IRQ3 | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PMC | PIOB | PIOA | ST | TC5 | TC4 | TC3 | TC2 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TC1 | TC0 | SPIB | SPIA | US1 | US0 | SW | FIQ |

• **Interrupt Set**

0 = No effect.
 1 = Sets corresponding interrupt.

AIC End of Interrupt Command Register

Register Name: AIC_EOICR
Access Type: Write-only
Offset: 0x130

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | – |

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

AIC Spurious Vector Register

Register Name: AIC_SPU
Access Type: Read/Write
Offset: 0x134
Reset Value: 0

| | | | | | | | |
|--------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| SPUVEC | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SPUVEC | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SPUVEC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPUVEC | | | | | | | |

- **SPUVEC: Spurious Interrupt Vector Handler Address**

The user may store the address of the spurious interrupt handler in this register.

Standard Interrupt Sequence

It is assumed that:

- The Advanced Interrupt Controller has been programmed, AIC_SVR are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
- The Instruction at address 0x18 (IRQ exception vector address) is

```
ldr pc, [pc, #-&F20]
```

When NIRQ is asserted, if the bit I of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the IRQ link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by 4.
2. The ARM core enters IRQ mode, if it is not already.
3. When the instruction loaded at address 0x18 is executed, the Program Counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
 - Set the current interrupt to be the pending one with the highest priority. The current level is the priority level of the current interrupt.
 - De-assert the NIRQ line on the processor. (Even if vectoring is not used, AIC_IVR must be read in order to de-assert NIRQ)
 - Automatically clear the interrupt, if it has been programmed to be edge triggered.
 - Push the current level on to the stack.
 - Return the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has effect to branch to the corresponding interrupt service routine. This should start by saving the Link Register (R14_irq) and the SPSR (SPSR_irq). Note that the Link Register must be decremented by 4 when it is saved, if it is to be restored directly into the Program Counter at the end of the interrupt.
5. Further interrupts can then be unmasked by clearing the I-bit in the CPSR, allowing re-assertion of the NIRQ to be taken into account by the core. This can occur if an interrupt with a higher priority than the current one occurs.
6. The Interrupt Handler can then proceed as required, saving the registers which will be used and restoring them at the end. During this phase, an interrupt of priority higher than the current level will restart the sequence from step 1. Note that if the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.
7. The I-bit in the CPSR must be set in order to mask interrupts before exiting, to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than old current level but with higher priority than the new current level, the NIRQ line is re-asserted, but the interrupt sequence does not immediately start because the I-bit is set in the core.

9. The SPSR (SPSR_irq) is restored. Finally, the saved value of the Link Register is restored directly into the PC. This has effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in the SPSR (the previous state of the ARM core).

Note: The I-bit in the SPSR is significant. If it is set, it indicates that the ARM core was just about to mask IRQ interrupts when the mask instruction was interrupted. Hence, when the SPSR is restored, the mask instruction is completed (IRQ is masked).

Fast Interrupt Sequence

It is assumed that:

- The Advanced Interrupt Controller has been programmed, AIC_SVR[0] is loaded with fast interrupt service routine address and the fast interrupt is enabled.
- The Instruction at address 0x1C(FIQ exception vector address) is:

```
ldr pc, [pc, #-&F20]
```
- Nested Fast Interrupts are not needed by the user.

When NFIQ is asserted, if the F-bit of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the Program Counter is loaded in the FIQ link register (R14_fiq) and the Program Counter (R15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by 4.
2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the Program Counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt (source 0 connected to the FIQ line), if it has been programmed to be edge triggered. In this case only, it de-asserts the NFIQ line on the processor.
4. The previous step has effect to branch to the corresponding interrupt service routine. It is not necessary to save the Link Register(R14_fiq) and the SPSR(SPSR_fiq) if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the NFIQ line.
6. Finally, the Link Register (R14_fiq) is restored into the PC after decrementing it by 4 (with instruction `sub pc, lr, #4` for example). This has effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the SPSR, masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The F-bit in the SPSR is significant. If it is set, it indicates that the ARM core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

PIO: Parallel I/O Controller

The AT91M42800A has 54 programmable I/O lines. I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins (see Tables 14 and 15). These lines are controlled by two separate and identical PIO Controllers called PIOA and PIOB. Each PIO controller also provides an internal interrupt signal to the Advanced Interrupt Controller.

Note: After a hardware reset, the PIO clock is disabled by default (see Power Management Controller on page 55). The user must configure the Power Management Controller before any access to the User Interface of the PIO.

Multiplexed I/O Lines

When a peripheral signal is not used in an application, the corresponding pin can be used as a parallel I/O. Each parallel I/O line is bi-directional, whether the peripheral defines the signal as input or output. Figure 47 shows the multiplexing of the peripheral signals with Parallel I/O signals.

A pin is controlled by the registers PIO_PER (PIO Enable) and PIO_PDR (PIO Disable). The register PIO_PSR (PIO Status) indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller.

When the PIO is selected, the peripheral input line is connected to zero.

Output Selection

The user can enable each individual I/O signal as an output with the registers PIO_OER (Output Enable) and PIO_ODR (Output Disable). The output status of the I/O signals can be read in the register PIO_OSR (Output Status). The direction defined has effect only if the pin is configured to be controlled by the PIO Controller.

I/O Levels

Each pin can be configured to be driven high or low. The level is defined in four different ways, according to the following conditions.

- If a pin is controlled by the PIO Controller and is defined as an output (see Output Selection above), the level is programmed using the registers PIO_SODR (Set Output Data) and PIO_CODR (Clear Output Data). In this case, the programmed value can be read in PIO_ODSR (Output Data Status).
- If a pin is controlled by the PIO Controller and is not defined as an output, the level is determined by the external circuit.
- If a pin is not controlled by the PIO Controller, the state of the pin is defined by the peripheral (see peripheral datasheets).

In all cases, the level on the pin can be read in the register PIO_PDSR (Pin Data Status).

Filters

Optional input glitch filtering is available on each pin and is controlled by the registers PIO_IFER (Input Filter Enable) and PIO_IFDR (Input Filter Disable). The input glitch filtering can be selected whether the pin is used for its peripheral function or as a parallel I/O line. The register PIO_IFSR (Input Filter Status) indicates whether or not the filter is activated for each pin.

Interrupts

Each parallel I/O can be programmed to generate an interrupt when a level change occurs. This is controlled by the PIO_IER (Interrupt Enable) and PIO_IDR (Interrupt Disable) registers which enable/disable the I/O interrupt by setting/clearing the corresponding bit in the PIO_IMR. When a change in level occurs, the corresponding bit in the PIO_ISR (Interrupt Status) is set whether the pin is used as a PIO or a peripheral and whether it is defined as input or output. If the corresponding interrupt in PIO_IMR (Interrupt Mask) is enabled, the PIO interrupt is asserted.

When PIO_ISR is read, the register is automatically cleared.

User Interface

Each individual I/O is associated with a bit position in the Parallel I/O user interface registers. Each of these registers are 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero.

Multi-driver (Open Drain)

Each I/O can be programmed for multi-driver option. This means that the I/O is configured as open drain (can only drive a low level) in order to support external drivers on the same pin. An external pull-up is necessary to guarantee a logic level of one when the pin is not being driven.

Registers PIO_MDER (Multi-Driver Enable) and PIO_MDDR (Multi-Driver Disable) control this option. Multi-driver can be selected whether the I/O pin is controlled by the PIO Controller or the peripheral. PIO_MDSR (Multi-Driver Status) indicates which pins are configured to support external drivers.

PIO Connection Tables

Table 14. PIO Controller A Connection Table

| PIO Controller | | Peripheral | | | | Reset State | Pin Number |
|----------------|-----------|-------------|-------------------------------|------------------|--------------------------|-------------|------------|
| Bit Number | Port Name | Port Name | Signal Description | Signal Direction | OFF ⁽¹⁾ Value | | |
| 0 | PA0 | IRQ0 | External Interrupt 0 | Input | 0 | PIO Input | 77 |
| 1 | PA1 | IRQ1 | External Interrupt 1 | Input | 0 | PIO Input | 78 |
| 2 | PA2 | IRQ2 | External Interrupt 2 | Input | 0 | PIO Input | 79 |
| 3 | PA3 | IRQ3 | External Interrupt 3 | Input | 0 | PIO Input | 80 |
| 4 | PA4 | FIQ | Fast Interrupt | Input | 0 | PIO Input | 81 |
| 5 | PA5 | SCK0 | USART0 Clock Signal | Bi-directional | 0 | PIO Input | 82 |
| 6 | PA6 | TXD0 | USART0 Transmit Data Signal | Output | – | PIO Input | 83 |
| 7 | PA7 | RXD0 | USART0 Receive Data Signal | Input | 0 | PIO Input | 86 |
| 8 | PA8 | SCK1 | USART1 Clock Signal | Bi-directional | 0 | PIO Input | 87 |
| 9 | PA9 | TXD1/NTRI | USART1 Transmit Data Signal | Output | – | PIO Input | 88 |
| 10 | PA10 | RXD1 | USART1 Receive Data Signal | Input | 0 | PIO Input | 89 |
| 11 | PA11 | SPCKA | SPIA Clock Signal | Bi-directional | 0 | PIO Input | 90 |
| 12 | PA12 | MISOA | SPIA Master In Slave Out | Bi-directional | 0 | PIO Input | 91 |
| 13 | PA13 | MOSIA | SPIA Master Out Slave In | Bi-directional | 0 | PIO Input | 92 |
| 14 | PA14 | NPCSA0/NSSA | SPIA Peripheral Chip Select 0 | Bi-directional | 1 | PIO Input | 93 |
| 15 | PA15 | NPCSA1 | SPIA Peripheral Chip Select 1 | Output | – | PIO Input | 94 |
| 16 | PA16 | NPCSA2 | SPIA Peripheral Chip Select 2 | Output | – | PIO Input | 95 |
| 17 | PA17 | NPCSA3 | SPIA Peripheral Chip Select 3 | Output | – | PIO Input | 98 |
| 18 | PA18 | SPCKB | SPIB Clock Signal | Bi-directional | 0 | PIO Input | 99 |
| 19 | PA19 | MISOB | SPIB Master In Slave Out | Bi-directional | 0 | PIO Input | 100 |
| 20 | PA20 | MOSIB | SPIB Master Out Slave In | Bi-directional | 0 | PIO Input | 101 |
| 21 | PA21 | NPCSB0/NSSB | SPIB Peripheral Chip Select 0 | Bi-directional | 1 | PIO Input | 102 |
| 22 | PA22 | NPCSB1 | SPIB Peripheral Chip Select 1 | Output | – | PIO Input | 103 |
| 23 | PA23 | NPCSB2 | SPIB Peripheral Chip Select 2 | Output | – | PIO Input | 104 |
| 24 | PA24 | NPCSB3 | SPIB Peripheral Chip Select 3 | Output | – | PIO Input | 105 |
| 25 | PA25 | MCKO | Master Clock Output | Output | – | MCKO | 106 |
| 26 | PA26 | – | – | – | – | PIO Input | 111 |
| 27 | PA27 | BMS | Boot Mode Select | Input | 0 | PIO Input | 123 |
| 28 | PA28 | – | – | Output | – | PIO Input | 131 |
| 29 | PA29 | PME | Protect Mode Enable | Input | 0 | PIO Input | 134 |

Note: 1. The OFF value is the default level seen on the peripheral input when the PIO line is enabled.

Table 15. PIO Controller B Connection Table

| PIO Controller | | Peripheral | | | | Reset State | Pin Number |
|----------------|-----------|------------|--------------------------|------------------|--------------------------|-------------|------------|
| Bit Number | Port Name | Port Name | Signal Description | Signal Direction | OFF ⁽¹⁾ Value | | |
| 0 | PB0 | NCS2 | Chip Select 2 | Output | – | NCS2 | 141 |
| 1 | PB1 | NCS3 | Chip Select 3 | Output | – | NCS3 | 142 |
| 2 | PB2 | A20/CS7 | Address 20/Chip Select 7 | Output | – | A20 | 27 |
| 3 | PB3 | A21/CS6 | Address 21/Chip Select 6 | Output | – | A21 | 28 |
| 4 | PB4 | A22/CS5 | Address 22/Chip Select 5 | Output | – | A22 | 29 |
| 5 | PB5 | A23/CS4 | Address 23/Chip Select 4 | Output | – | A23 | 30 |
| 6 | PB6 | TCLK0 | Timer0 Clock Signal | Input | 0 | PIO Input | 53 |
| 7 | PB7 | TIOA0 | Timer0 Signal A | Bi-directional | 0 | PIO Input | 54 |
| 8 | PB8 | TIOB0 | Timer0 Signal B | Bi-directional | 0 | PIO Input | 55 |
| 9 | PB9 | TCLK1 | Timer1 Clock Signal | Input | 0 | PIO Input | 56 |
| 10 | PB10 | TIOA1 | Timer1 Signal A | Bi-directional | 0 | PIO Input | 57 |
| 11 | PB11 | TIOB1 | Timer1 Signal B | Bi-directional | 0 | PIO Input | 58 |
| 12 | PB12 | TCLK2 | Timer2 Clock Signal | Input | 0 | PIO Input | 59 |
| 13 | PB13 | TIOA2 | Timer2 Signal A | Bi-directional | 0 | PIO Input | 62 |
| 14 | PB14 | TIOB2 | Timer2 Signal B | Bi-directional | 0 | PIO Input | 63 |
| 15 | PB15 | TCLK3 | Timer3 Clock Signal | Input | 0 | PIO Input | 64 |
| 16 | PB16 | TIOA3 | Timer3 Signal A | Bi-directional | 0 | PIO Input | 65 |
| 17 | PB17 | TIOB3 | Timer3 Signal B | Bi-directional | 0 | PIO Input | 66 |
| 18 | PB18 | TCLK4 | Timer4 Clock Signal | Input | 0 | PIO Input | 67 |
| 19 | PB19 | TIOA4 | Timer4 Signal A | Bi-directional | 0 | PIO Input | 68 |
| 20 | PB20 | TIOB4 | Timer4 Signal B | Bi-directional | 0 | PIO Input | 69 |
| 21 | PB21 | TCLK5 | Timer5 Clock Signal | Input | 0 | PIO Input | 70 |
| 22 | PB22 | TIOA5 | Timer5 Signal A | Bi-directional | 0 | PIO Input | 75 |
| 23 | PB23 | TIOB5 | Timer5 Signal B | Bi-directional | 0 | PIO Input | 76 |

Note: 1. The OFF value is the default level seen on the peripheral input when the PIO line is enabled.

PIO User Interface

PIO Controller A Base Address: 0xFFFFEC000 (Code Label `PIOA_BASE`)

PIO Controller B Base Address: 0xFFFF0000 (Code Label `PIOB_BASE`)

Table 16. PIO Controller Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|-------------------------------|----------|------------|----------------------------------|
| 0x00 | PIO Enable Register | PIO_PER | Write-only | – |
| 0x04 | PIO Disable Register | PIO_PDR | Write-only | – |
| 0x08 | PIO Status Register | PIO_PSR | Read-only | 0x3DFFFFFF (A) 0x00FFFFC0 (B) |
| 0x0C | Reserved | – | – | – |
| 0x10 | Output Enable Register | PIO_OER | Write-only | – |
| 0x14 | Output Disable Register | PIO_ODR | Write-only | – |
| 0x18 | Output Status Register | PIO_OSR | Read-only | 0 |
| 0x1C | Reserved | – | – | – |
| 0x20 | Input Filter Enable Register | PIO_IFER | Write-only | – |
| 0x24 | Input Filter Disable Register | PIO_IFDR | Write-only | – |
| 0x28 | Input Filter Status Register | PIO_IFSR | Read-only | 0 |
| 0x2C | Reserved | – | – | – |
| 0x30 | Set Output Data Register | PIO_SODR | Write-only | – |
| 0x34 | Clear Output Data Register | PIO_CODR | Write-only | – |
| 0x38 | Output Data Status Register | PIO_ODSR | Read-only | 0 |
| 0x3C | Pin Data Status Register | PIO_PDSR | Read-only | (see Note 1) |
| 0x40 | Interrupt Enable Register | PIO_IER | Write-only | – |
| 0x44 | Interrupt Disable Register | PIO_IDR | Write-only | – |
| 0x48 | Interrupt Mask Register | PIO_IMR | Read-only | 0 |
| 0x4C | Interrupt Status Register | PIO_ISR | Read-only | (see Note 2) |
| 0x50 | Multi-driver Enable Register | PIO_MDER | Write-only | – |
| 0x54 | Multi-driver Disable Register | PIO_MDDR | Write-only | – |
| 0x58 | Multi-driver Status Register | PIO_MDSR | Read-only | 0 |
| 0x5C | Reserved | – | – | – |

- Notes:
1. The reset value of this register depends on the level of the external pins at reset.
 2. This register is cleared at reset. However, the first read of the register can give a value not equal to zero if any changes have occurred on any pins between the reset and the read.

PIO Enable Register

Register Name: PIO_PER
Access Type: Write-only
Offset: 0x00

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to enable individual pins to be controlled by the PIO Controller instead of the associated peripheral. When the PIO is enabled, the associated peripheral (if any) is held at logic zero.

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

PIO Disable Register

Register Name: PIO_PDR
Access Type: Write-only
Offset: 0x04

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to disable PIO control of individual pins. When the PIO control is disabled, the normal peripheral function is enabled on the corresponding pin.

0 = No effect.

1 = Disables PIO control (enables peripheral control) on the corresponding pin.

PIO Status Register

Register Name: PIO_PSR
Access Type: Read-only
Offset: 0x08
Reset Value: 0x3DFFFFFF (A)
 0x00FFFFFFC0 (B)

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register indicates which pins are enabled for PIO control. This register is updated when PIO lines are enabled or disabled.

- 0 = PIO is inactive on the corresponding line (peripheral is active).
- 1 = PIO is active on the corresponding line (peripheral is inactive).

PIO Output Enable Register

Register Name: PIO_OER
Access Type: Write-only
Offset: 0x10

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to enable PIO output drivers. If the pin is driven by a peripheral, this has no effect on the pin, but the information is stored. The register is programmed as follows:

- 0 = No effect.
- 1 = Enables the PIO output on the corresponding pin.

PIO Output Disable Register

Register Name: PIO_ODR
Access Type: Write-only
Offset: 0x14

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to disable PIO output drivers. If the pin is driven by the peripheral, this has no effect on the pin, but the information is stored. The register is programmed as follows:

- 0 = No effect.
- 1 = Disables the PIO output on the corresponding pin.

PIO Output Status Register

Register Name: PIO_OSR
Access Type: Read-only
Offset: 0x18
Reset Value: 0

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register shows the PIO pin control (output enable) status which is programmed in PIO_OER and PIO ODR. The defined value is effective only if the pin is controlled by the PIO. The register reads as follows:

- 0 = The corresponding PIO is input on this line.
- 1 = The corresponding PIO is output on this line.

PIO Input Filter Enable Register

Register Name: PIO_IFER
Access Type: Write-only
Offset: 0x20

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to enable input glitch filters. It affects the pin whether or not the PIO is enabled. The register is programmed as follows:

- 0 = No effect.
- 1 = Enables the glitch filter on the corresponding pin.

PIO Input Filter Disable Register

Register Name: IO_IFDR
Access Type: Write-only
Offset: 0x24

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to disable input glitch filters. It affects the pin whether or not the PIO is enabled. The register is programmed as follows:

- 0 = No effect.
- 1 = Disables the glitch filter on the corresponding pin.

PIO Input Filter Status Register

Register Name: PIO_IFSR
Access Type: Read-only
Offset: 0x28
Reset Value: 0

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register indicates which pins have glitch filters selected. It is updated when PIO outputs are enabled or disabled by writing to PIO_IFER or PIO_IFDR.

- 0 = Filter is not selected on the corresponding input.
- 1 = Filter is selected on the corresponding input (peripheral and PIO).

PIO Set Output Data Register

Register Name: PIO_SODR
Access Type: Write-only
Offset: 0x30

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to set PIO output data. It affects the pin only if the corresponding PIO output line is enabled and if the pin is controlled by the PIO. Otherwise, the information is stored.

- 0 = No effect.
- 1 = PIO output data on the corresponding pin is set.

PIO Clear Output Data Register

Register Name: PIO_CODR
Access Type: Write-only
Offset: 0x34

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to clear PIO output data. It affects the pin only if the corresponding PIO output line is enabled and if the pin is controlled by the PIO. Otherwise, the information is stored.

- 0 = No effect.
- 1 = PIO output data on the corresponding pin is cleared.

PIO Output Data Status Register

Register Name: PIO_ODSR
Access Type: Read-only
Offset: 0x38
Reset Value: 0

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register shows the output data status which is programmed in PIO_SODR or PIO_CODR. The defined value is effective only if the pin is controlled by the PIO Controller and only if the pin is defined as an output.

- 0 = The output data for the corresponding line is programmed to 0.
- 1 = The output data for the corresponding line is programmed to 1.

PIO Pin Data Status Register

Register Name: PIO_PDSR
Access Type: Read-only
Offset: 0x3C
Reset Value: Undefined

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register shows the state of the physical pin of the chip. The pin values are always valid, regardless of whether the pins are enabled as PIO, peripheral, input or output. The register reads as follows:

- 0 = The corresponding pin is at logic 0.
- 1 = The corresponding pin is at logic 1.



PIO Interrupt Enable Register

Register Name: PIO_IER
Access Type: Write-only
Offset: 0x40

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to enable PIO interrupts on the corresponding pin. It has effect whether PIO is enabled or not.

0 = No effect.

1 = Enables an interrupt when a change of logic level is detected on the corresponding pin.

PIO Interrupt Disable Register

Register Name: PIO_IDR
Access Type: Write-only
Offset: 0x44

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to disable PIO interrupts on the corresponding pin. It has effect whether the PIO is enabled or not.

0 = No effect.

1 = Disables the interrupt on the corresponding pin. Logic level changes are still detected.

PIO Interrupt Mask Register

Register Name: PIO_IMR
Access Type: Read-only
Offset: 0x48
Reset Value: 0

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register shows which pins have interrupts enabled. It is updated when interrupts are enabled or disabled by writing to PIO_IER or PIO_IDR.

- 0 = Interrupt is not enabled on the corresponding input pin.
- 1 = Interrupt is enabled on the corresponding input pin.

PIO Interrupt Status Register

Register Name: PIO_ISR
Access Type: Read-only
Offset: 0x4C

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register indicates for each pin when a logic value change has been detected (rising or falling edge). This is valid whether the PIO is selected for the pin or not and whether the pin is an input or an output.

The register is reset to zero following a read, and at reset.

- 0 = No input change has been detected on the corresponding pin since the register was last read.
- 1 = At least one input change has been detected on the corresponding pin since the register was last read.

PIO Multi-drive Enable Register

Register Name: PIO_MDER
Access Type: Write-only
Offset: 0x50

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to enable PIO output drivers to be configured as open drain to support external drivers on the same pin.

- 0 = No effect.
- 1 = Enables multi-drive option on the corresponding pin.

PIO Multi-drive Disable Register

Register Name: PIO_MDDR
Access Type: Write-only
Offset: 0x54

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register is used to disable the open drain configuration of the output buffer.

- 0 = No effect.
- 1 = Disables the multi-driver option on the corresponding pin.

PIO Multi-drive Status Register

Register Name: PIO_MDSR
Access Type: Read-only
Offset: 0x58
Reset Value: 0

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register indicates which pins are configured with open drain drivers.

- 0 = PIO is not configured as an open drain.
- 1 = PIO is configured as an open drain.

SF: Special Function Registers

The AT91M42800A provides registers that implement the following special functions:

- Chip Identification
- RESET Status
- Protect Mode (see "Protect Mode" on page 84)

Chip Identification

The AT91M42800A chip identifier is 0x14280041.

SF User Interface

Chip ID Base Address: 0xFFFF0000 (Code Label `SF_BASE`)

Table 17. SF Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|----------------------------|---------|------------|--------------------------|
| 0x00 | Chip ID Register | SF_CIDR | Read-only | Hardwired |
| 0x04 | Chip ID Extension Register | SF_EXID | Read-only | Hardwired |
| 0x08 | Reset Status Register | SF_RSR | Read-only | See register description |
| 0x0C | Reserved | – | – | – |
| 0x10 | Reserved | – | – | – |
| 0x14 | Reserved | – | – | – |
| 0x18 | Protect Mode Register | SF_PMR | Read/Write | 0x0 |

Chip ID Register

Register Name: SF_CIDR
Access Type: Read-only
Offset: 0x00

| | | | | | | | | |
|--------|--------|----|---------|--------|----|----|----|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
| EXT | NVPTYP | | | ARCH | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| ARCH | | | | VDSIZ | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| NVDSIZ | | | | NVPSIZ | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0 | 1 | 0 | VERSION | | | | | 0 |

• **VERSION: Version of the Chip (Code Label SF_VERSION)**

This value is incremented by one with each new version of the chip (from zero to a maximum value of 31).

• **NVPSIZ: Nonvolatile Program Memory Size**

| NVPSIZ | | | | Size | Code Label: SF_NVPSIZ |
|--------|---|---|---|------------|-----------------------|
| 0 | 0 | 0 | 0 | None | SF_NVPSIZ_NONE |
| 0 | 0 | 1 | 1 | 32K Bytes | SF_NVPSIZ_32K |
| 0 | 1 | 0 | 1 | 64K Bytes | SF_NVP_SIZ_64K |
| 0 | 1 | 1 | 1 | 128K Bytes | SF_NVP_SIZ_128K |
| 1 | 0 | 0 | 1 | 256K Bytes | SF_NVP_SIZ_256K |
| Others | | | | Reserved | – |

• **NVDSIZ: Nonvolatile Data Memory Size**

| NVDSIZ | | | | Size | Code Label: SF_NVDSIZ |
|--------|---|---|---|----------|-----------------------|
| 0 | 0 | 0 | 0 | None | SF_NVDSIZ_NONE |
| Others | | | | Reserved | – |

• **VDSIZ: Volatile Data Memory Size**

| VDSIZ | | | | Size | Code Label: SF_VDSIZ |
|--------|---|---|---|----------|----------------------|
| 0 | 0 | 0 | 0 | None | SF_VDSIZ_NONE |
| 0 | 0 | 0 | 1 | 1K Byte | SF_VDSIZ_1K |
| 0 | 0 | 1 | 0 | 2K Bytes | SF_VDSIZ_2K |
| 0 | 1 | 0 | 0 | 4K Bytes | SF_VDSIZ_4K |
| 1 | 0 | 0 | 0 | 8K Bytes | SF_VDSIZ_8K |
| Others | | | | Reserved | – |



- **ARCH: Chip Architecture**

Code of Architecture: Two BCD digits

| ARCH | Selected ARCH | Code Label: SF_ARCH |
|-----------|---------------|----------------------------|
| 0110 0011 | AT91x63yyy | SF_ARCH_AT91x63 |
| 0100 0000 | AT91x40yyy | SF_ARCH_AT91x40 |
| 0101 0101 | AT91x55yyy | SF_ARCH_AT91x55 |

- **NVPTYP: Nonvolatile Program Memory Type**

| NVPTYP | | | Type | Code Label: SF_NVPTYP |
|--------|---|---|--------------------------|------------------------------|
| 0 | 0 | 1 | "M" Series or "F" Series | SF_NVPTYP_M |
| 1 | 0 | 0 | "R" Series | SF_NVPTYP_R |

- **EXT: Extension Flag (Code Label **SF_EXT**)**

0 = Chip ID has a single register definition without extensions.

1 = An extended Chip ID exists (to be defined in the future).

Chip ID Extension Register

Register Name: SF_EXID

Access Type: Read-only

Offset: 0x04

This register is reserved for future use. It will be defined when needed.

Reset Status Register

Register Name: SF_RSR
Access Type: Read-only
Offset: 0x08

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESET | | | | | | | |

• **RESET: Reset Status Information**

This field indicates whether the reset was demanded by the external system (via NRST) or by the Watchdog internal reset request.

| Reset | Cause of Reset | Code Label |
|-------|-------------------|--------------|
| 0x6C | External Pin | SF_EXT_RESET |
| 0x53 | Internal Watchdog | SF_WD_RESET |

SF Protect Mode Register

Register Name: SF_PMR
Offset: 0x18
Reset Value: 0x00000000

| | | | | | | | |
|--------|----|-----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| PMRKEY | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PMRKEY | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | AIC | – | – | – | – | – |

• **AIC: AIC Protect Mode Enable (Code Label SF_AIC)**

0 = The Advanced Interrupt Controller runs in Normal Mode.
 1 = The Advanced Interrupt Controller runs in Protect Mode.
 See "Protect Mode" on page 84.

• **PMRKEY: Protect Mode Register Key**

Used only when writing SF_PMR. PMRKEY reads 0.
 0x27A8: Write access in SF_PMR is allowed.
 Other value: Write access in SF_PMR is prohibited.



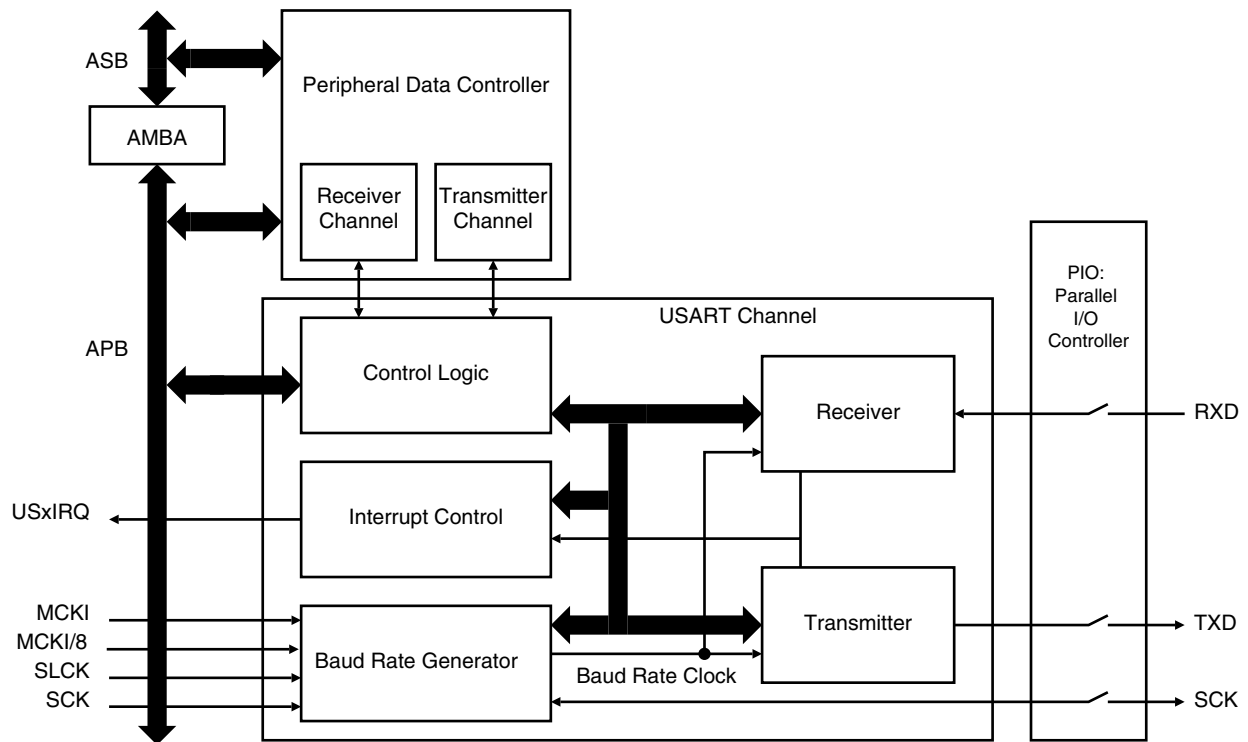
USART: Universal Synchronous/Asynchronous Receiver/Transmitter

The AT91M42800A provides two identical, full-duplex, universal synchronous/asynchronous receiver/transmitters that interface to the APB and are connected to the Peripheral Data Controller.

The main features are:

- Programmable Baud Rate Generator with External or Internal Clock, as well as Slow Clock
- Parity, Framing and Overrun Error Detection
- Line Break Generation and Detection
- Automatic Echo, Local Loopback and Remote Loopback channel modes
- Multi-drop Mode: Address Detection and Generation
- Interrupt Generation
- Two Dedicated Peripheral Data Controller channels
- 5-, 6-, 7-, 8- and 9-bit character length

Figure 48. USART Block Diagram



Pin Description

Each USART channel has the following external signals:

| Name | Description |
|------|---|
| SCK | USART Serial clock can be configured as input or output: SCK is configured as input if an External clock is selected (USCLKS = 3) SCK is driven as output if the External Clock is disabled (USCLKS ≠ 3) and Clock output is enabled (CLKO = 1) |
| TXD | Transmit Serial Data is an output |
| RXD | Receive Serial Data is an input |

- Notes:
1. After a hardware reset, the USART clock is disabled by default (see “PMC: Power Management Controller” on page 55). The user must configure the Power Management Controller before any access to the User Interface of the USART.
 2. After a hardware reset, the USART pins are deselected by default (see “PIO: Parallel I/O Controller” on page 98). The user must configure the PIO Controller before enabling the transmitter or receiver. If the user selects one of the internal clocks, SCK can be configured as a PIO.

Baud Rate Generator

The Baud Rate Generator provides the bit period clock (the Baud Rate clock) to both the Receiver and the Transmitter.

The Baud Rate Generator can select between external and internal clock sources. The external clock source is SCK. The internal clock sources can be either the master clock MCK or the master clock divided by 8 (MCK/8).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the system clock (MCK) period. The external clock frequency must be at least 2.5 times lower than the system clock.

When the USART is programmed to operate in Asynchronous Mode (SYNC = 0 in the Mode Register US_MR), the selected clock is divided by 16 times the value (CD) written in US_BRGR (Baud Rate Generator Register). If US_BRGR is set to 0, the Baud Rate Clock is disabled.

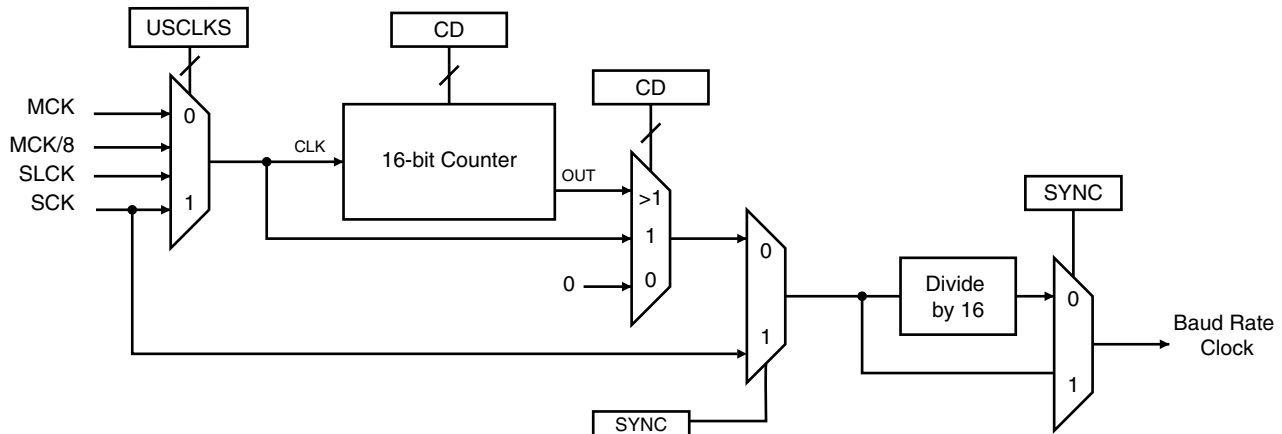
$$\text{Baud Rate} = \frac{\text{Selected Clock}}{16 \times \text{CD}}$$

When the USART is programmed to operate in Synchronous Mode (SYNC = 1) and the selected clock is internal (USCLKS ≠ 3 in the Mode Register US_MR), the Baud Rate Clock is the internal selected clock divided by the value written in US_BRGR. If US_BRGR is set to 0, the Baud Rate Clock is disabled.

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{\text{CD}}$$

In Synchronous Mode with external clock selected (USCLKS = 3), the clock is provided directly by the signal on the SCK pin. No division is active. The value written in US_BRGR has no effect.

Figure 49. Baud Rate Generator



Receiver

Asynchronous Receiver

The USART is configured for asynchronous operation when SYNC = 0 (bit 7 of US_MR). In asynchronous mode, the USART detects the start of a received character by sampling the RXD signal until it detects a valid start bit. A low level (space) on RXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence a space which is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the RXD at the theoretical mid-point of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (one bit period) so the sampling point is 8 cycles (0.5 bit periods) after the start of the bit. The first sampling point is therefore 24 cycles (1.5 bit periods) after the falling edge of the start bit was detected. Each subsequent bit is sampled 16 cycles (1 bit period) after the previous one.

Figure 50. Asynchronous Mode: Start Bit Detection

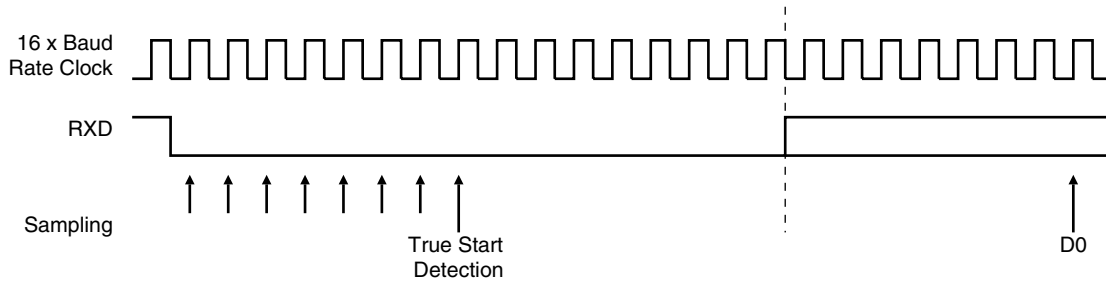
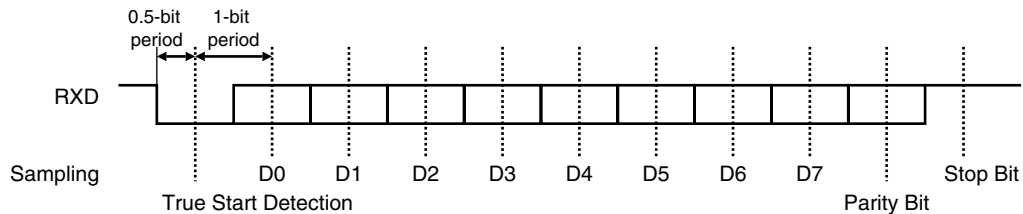


Figure 51. Asynchronous Mode: Character Reception

Example: 8-bit, parity enabled 1 stop

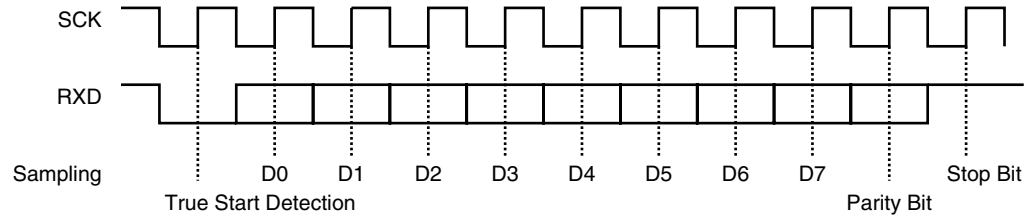


Synchronous Receiver

When configured for synchronous operation (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate clock. If a low level is detected, it is considered as a start. Data bits, parity bit and stop bit are sampled and the receiver waits for the next start bit. See example in Figure 52.

Figure 52. Synchronous Mode: Character Reception

Example: 8-bit, parity enabled 1 stop



Receiver Ready

When a complete character is received, it is transferred to the US_RHR and the RXRDY status bit in US_CSR is set. If US_RHR has not been read since the last transfer, the OVRE status bit in US_CSR is set.

Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in US_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in US_CSR is set.

Framing Error

If a character is received with a stop bit at low level and with at least one data bit at high level, a framing error is generated. This sets FRAME in US_CSR.

Time-out

This function allows an idle condition on the RXD line to be detected. The maximum delay for which the USART should wait for a new character to arrive while the RXD line is inactive (high level) is programmed in US_RTOR (Receiver Tim-out). When this register is set to 0, no time-out is detected. Otherwise, the receiver waits for a first character and then initializes a counter which is decremented at each bit period and reloaded at each byte reception. When the counter reaches 0, the TIMEOUT bit in US_CSR is set. The user can restart the wait for a first character with the STTTO (Start Time-out) bit in US_CR.

Calculation of time-out duration:

$$\text{Duration} = \text{Value} \times 4 \times \text{Bit Period}$$

Transmitter

The transmitter has the same behavior in both synchronous and asynchronous operating modes. Start bit, data bits, parity bit and stop bits are serially shifted, lowest significant bit first, on the falling edge of the serial clock. See example in Figure 53.

The number of data bits is selected in the CHRL field in US_MR.

The parity bit is set according to the PAR field in US_MR.

The number of stop bits is selected in the NBSTOP field in US_MR.

When a character is written to US_THR (Transmit Holding), it is transferred to the Shift Register as soon as it is empty. When the transfer occurs, the TXRDY bit in US_CSR is set until a new character is written to US_THR. If Transmit Shift Register and US_THR are both empty, the TXEMPTY bit in US_CSR is set.

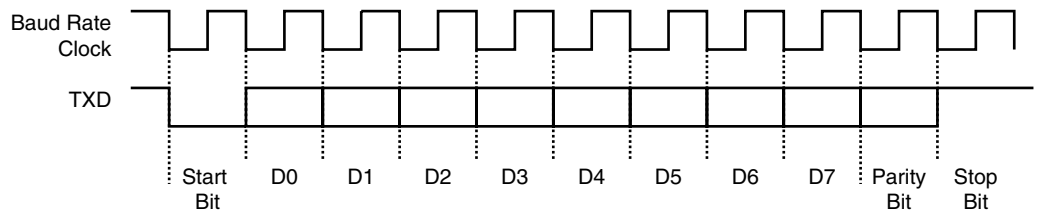
Time-guard

The Time-guard function allows the transmitter to insert an idle state on the TXD line between two characters. The duration of the idle state is programmed in US_TTGR (Transmitter Time-guard). When this register is set to zero, no time-guard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in US_TTGR.

$$\text{Idle state duration between two characters} = \frac{\text{Time-guard Value}}{\text{Bit Period}} \times \text{Bit Period}$$

Figure 53. Synchronous and Asynchronous Modes: Character Transmission

Example: 8-bit, parity enabled 1 stop



Multi-drop Mode

When the field PAR in US_MR equals 11X (binary value), the USART is configured to run in Multi-drop mode. In this case, the parity error bit PARE in US_CSR is set when data is detected with a parity bit set to identify an address byte. PARE is cleared with the Reset Status Bits Command (RSTSTA) in US_CR. If the parity bit is detected low, identifying a data byte, PARE is not set.

The transmitter sends an address byte (parity bit set) when a Send Address Command (SENDA) is written to US_CR. In this case, the next byte written to US_THR will be transmitted as an address. After this any byte transmitted will have the parity bit cleared.

Break

A break condition is a low signal level that has a duration of at least one character (including start/stop bits and parity).

Transmit Break

The transmitter generates a break condition on the TXD line when STTBRK is set in US_CR (Control Register). In this case, the character present in the Transmit Shift Register is completed before the line is held low.

To cancel a break condition on the TXD line, the STPBRK command in US_CR must be set. The USART completes a minimum break duration of one character length. The TXD line then returns to high level (idle state) for at least 12 bit periods, or the value of the

Time-guard register if it is greater than 12, to ensure that the end of break is correctly detected. Then the transmitter resumes normal operation.

The BREAK is managed like a character:

- The STTBRK and the STPBRK commands are performed only if the transmitter is ready (bit TXRDY = 1 in US_CSR)
- The STTBRK command blocks the transmitter holding register (bit TXRDY is cleared in US_CSR) until the break has started
- A break is started when the Shift Register is empty (any previous character is fully transmitted). US_CSR.TXEMPTY is cleared. The break blocks the transmitter shift register until it is completed (high level for at least 12 bit periods after the STPBRK command is requested)

In order to avoid unpredictable states:

- STTBRK and STPBRK commands must not be requested at the same time
- Once an STTBRK command is requested, further STTBRK commands are ignored until the BREAK is ended (high level for at least 12 bit periods)
- All STPBRK commands requested without a previous STTBRK command are ignored
- A byte written into the Transmit Holding Register while a break is pending but not started (bit TXRDY = 0 in US_CSR) is ignored
- It is *not permitted* to write new data in the Transmit Holding Register while a break is in progress (STPBRK has not been requested), even though TXRDY = 1 in US_CSR.
- A new STTBRK command *must not* be issued until an existing break has ended (TXEMPTY=1 in US_CSR).

The standard break transmission sequence is:

1. Wait for the transmitter ready (US_CSR.TXRDY = 1)
2. Send the STTBRK command (write 0x0200 to US_CR)
3. Wait for the transmitter ready (bit TXRDY = 1 in US_CSR)
4. Send the STPBRK command (write 0x0400 to US_CR)

The next byte can then be sent:

5. Wait for the transmitter ready (bit TXRDY = 1 in US_CSR)
6. Send the next byte (write byte to US_THR)

Each of these steps can be scheduled by using the interrupt if the bit TXRDY in US_IMR is set.

For character transmission, the USART channel must be enabled before sending a break.

Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or at least one sample in synchronous operating mode. RXBRK is also asserted when an end of break is detected.

Both the beginning and the end of a break can be detected by interrupt if the bit US_IMR.RXBRK is set.

Peripheral Data Controller

Each USART channel is closely connected to a corresponding Peripheral Data Controller channel. One is dedicated to the receiver. The other is dedicated to the transmitter.

The PDC is disabled if 9-bit character length is selected (MODE9 = 1) in US_MR.

The PDC channel is programmed using US_TPR (Transmit Pointer) and US_TCR (Transmit Counter) for the transmitter and US_RPR (Receive Pointer) and US_RCR (Receive Counter) for the receiver. The status of the PDC is given in US_CSR by the ENDTX bit for the transmitter and by the ENDRX bit for the receiver.

The pointer registers (US_TPR and US_RPR) are used to store the address of the transmit or receive buffers. The counter registers (US_TCR and US_RCR) are used to store the size of these buffers.

The receiver data transfer is triggered by the RXRDY bit and the transmitter data transfer is triggered by TXRDY. When a transfer is performed, the counter is decremented and the pointer is incremented. When the counter reaches 0, the status bit is set (ENDRX for the receiver, ENDTX for the transmitter in US_CSR) and can be programmed to generate an interrupt. Transfers are then disabled until a new non-zero counter value is programmed.

Interrupt Generation

Each status bit in US_CSR has a corresponding bit in US_IER (Interrupt Enable) and US_IDR (Interrupt Disable) which controls the generation of interrupts by asserting the USART interrupt line connected to the Advanced Interrupt Controller. US_IMR (Interrupt Mask Register) indicates the status of the corresponding bits.

When a bit is set in US_CSR and the same bit is set in US_IMR, the interrupt line is asserted.

Channel Modes

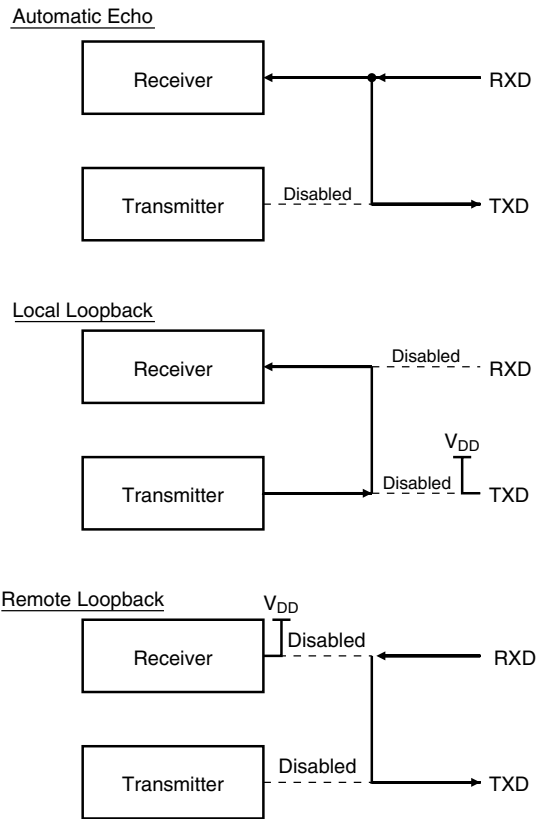
The USART can be programmed to operate in three different test modes, using the field CHMODE in US_MR.

Automatic echo mode allows bit by bit re-transmission. When a bit is received on the RXD line, it is sent to the TXD line. Programming the transmitter has no effect.

Local loopback mode allows the transmitted characters to be received. TXD and RXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The RXD pin level has no effect and the TXD pin is held high, as in idle state.

Remote loopback mode directly connects the RXD pin to the TXD pin. The Transmitter and the Receiver are disabled and have no effect. This mode allows bit-by-bit re-transmission.

Figure 54. Channel Modes



USART User Interface

Base Address USART0: 0xFFFC0000 (Code Label USART0_BASE)

Base Address USART1: 0xFFFC4000 (Code Label USART1_BASE)

Table 1. USART Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|---------------------------------|---------|------------|-------------|
| 0x00 | Control Register | US_CR | Write-only | – |
| 0x04 | Mode Register | US_MR | Read/Write | 0 |
| 0x08 | Interrupt Enable Register | US_IER | Write-only | – |
| 0x0C | Interrupt Disable Register | US_IDR | Write-only | – |
| 0x10 | Interrupt Mask Register | US_IMR | Read-only | 0 |
| 0x14 | Channel Status Register | US_CSR | Read-only | 0x18 |
| 0x18 | Receiver Holding Register | US_RHR | Read-only | 0 |
| 0x1C | Transmitter Holding Register | US_THR | Write-only | – |
| 0x20 | Baud Rate Generator Register | US_BRGR | Read/Write | 0 |
| 0x24 | Receiver Time-out Register | US_RTOR | Read/Write | 0 |
| 0x28 | Transmitter Time-guard Register | US_TTGR | Read/Write | 0 |
| 0x2C | Reserved | – | – | – |
| 0x30 | Receive Pointer Register | US_RPR | Read/Write | 0 |
| 0x34 | Receive Counter Register | US_RCR | Read/Write | 0 |
| 0x38 | Transmit Pointer Register | US_TPR | Read/Write | 0 |
| 0x3C | Transmit Counter Register | US_TCR | Read/Write | 0 |

USART Control Register

Name: US_CR
Access Type: Write-only
Offset: 0x00

| | | | | | | | |
|-------|------|-------|-------|-------|--------|--------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | SENDA | STTTO | STPBRK | STTBRK | RSTSTA |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDIS | TXEN | RXDIS | RXEN | RSTTX | RSTRX | - | - |

- **RSTRX: Reset Receiver (Code Label US_RSTRX)**
 0 = No effect.
 1 = The receiver logic is reset.
- **RSTTX: Reset Transmitter (Code Label US_RSTTX)**
 0 = No effect.
 1 = The transmitter logic is reset.
- **RXEN: Receiver Enable (Code Label US_RXEN)**
 0 = No effect.
 1 = The receiver is enabled if RXDIS is 0.
- **RXDIS: Receiver Disable (Code Label US_RXDIS)**
 0 = No effect.
 1 = The receiver is disabled.
- **TXEN: Transmitter Enable (Code Label US_TXEN)**
 0 = No effect.
 1 = The transmitter is enabled if TXDIS is 0.
- **TXDIS: Transmitter Disable (Code Label US_TXDIS)**
 0 = No effect.
 1 = The transmitter is disabled.
- **RSTSTA: Reset Status Bits (Code Label US_RSTSTA)**
 0 = No effect.
 1 = Resets the status bits PARE, FRAME, OVRE and RXBRK in the US_CSR.
- **STTBRK: Start Break (Code Label US_STTBRK)**
 0 = No effect.
 1 = If break is not being transmitted, start transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted.
- **STPBRK: Stop Break (Code Label US_STPBRK)**
 0 = No effect.
 1 = If a break is being transmitted, stop transmission of the break after a minimum of one character length and transmit a high level during 12 bit periods.
- **STTTO: Start Time-out (Code Label US_STTTO)**
 0 = No effect.
 1 = Start waiting for a character before clocking the time-out counter.
- **SENDA: Send Address (Code Label US_SENDA)**
 0 = No effect.
 1 = In Multi-drop Mode only, the next character written to the US_THR is sent with the address bit set.

USART Mode Register

Name: US_MR
Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

| | | | | | | | |
|--------|----|--------|----|-----|------|-------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | CLKO | MODE9 | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHMODE | | NBSTOP | | PAR | | SYNC | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHRL | | USCLKS | | – | – | – | – |

- **USCLKS: Clock Selection (Baud Rate Generator Input Clock)**

| USCLKS | | Selected Clock | Code Label: us_clks |
|--------|---|----------------|----------------------------|
| 0 | 0 | MCK | US_CLKS_MCK |
| 0 | 1 | MCK/8 | US_CLKS_MCK8 |
| 1 | 0 | Slow Clock | US_CLKS_SLCK |
| 1 | 1 | External (SCK) | US_CLKS_SCK |

- **CHRL: Character Length**

| CHRL | | Character Length | Code Label: us_chrl |
|------|---|------------------|----------------------------|
| 0 | 0 | Five bits | US_CHRL_5 |
| 0 | 1 | Six bits | US_CHRL_6 |
| 1 | 0 | Seven bits | US_CHRL_7 |
| 1 | 1 | Eight bits | US_CHRL_8 |

Start, stop and parity bits are added to the character length.

- **SYNC: Synchronous Mode Select (Code Label **us_sync**)**

0 = USART operates in Asynchronous Mode.
 1 = USART operates in Synchronous Mode.

- **PAR: Parity Type**

| PAR | | | Parity Type | Code Label: us_par |
|-----|---|---|----------------------------|---------------------------|
| 0 | 0 | 0 | Even Parity | US_PAR_EVEN |
| 0 | 0 | 1 | Odd Parity | US_PAR_ODD |
| 0 | 1 | 0 | Parity forced to 0 (Space) | US_PAR_SPACE |
| 0 | 1 | 1 | Parity forced to 1 (Mark) | US_PAR_MARK |
| 1 | 0 | x | No parity | US_PAR_NO |
| 1 | 1 | x | Multi-drop mode | US_PAR_MULTIDROP |

- **NBSTOP: Number of Stop Bits**

The interpretation of the number of stop bits depends on SYNC.

Table 2.

| NBSTOP | | Asynchronous (SYNC = 0) | Synchronous (SYNC = 1) | Code Label: <code>US_NBSTOP</code> |
|--------|---|-------------------------|------------------------|------------------------------------|
| 0 | 0 | 1 stop bit | 1 stop bit | <code>US_NBSTOP_1</code> |
| 0 | 1 | 1.5 stop bits | Reserved | <code>US_NBSTOP_1_5</code> |
| 1 | 0 | 2 stop bits | 2 stop bits | <code>US_NBSTOP_2</code> |
| 1 | 1 | Reserved | Reserved | – |

Note: 1.5 or 2 stop bits are reserved for the TX function. The RX function uses only the 1 stop bit (there is no check on the 2 stop bit timeslot if NBSTO P= 10).

- **CHMODE: Channel Mode**

| CHMODE | | Mode Description | Code Label: <code>US_CHMODE</code> |
|--------|---|--|--|
| 0 | 0 | Normal Mode The USART Channel operates as an Rx/Tx USART. | <code>US_CHMODE_NORMAL</code> |
| 0 | 1 | Automatic Echo Receiver Data Input is connected to TXD pin. | <code>US_CHMODE_AUTOMATIC_ECHO</code> |
| 1 | 0 | Local Loopback Transmitter Output Signal is connected to Receiver Input Signal. | <code>US_CHMODE_LOCAL_LOOPBACK</code> |
| 1 | 1 | Remote Loopback RXD pin is internally connected to TXD pin. | <code>US_CHMODE_REMOTE_LOOPBACK</code> |

- **MODE9: 9-Bit Character Length (Code Label `US_MODE9`)**

0 = CHRL defines character length.

1 = 9-Bit character length.

- **CKLO: Clock Output Select (Code Label `US_CLKO`)**

0 = The USART does not drive the SCK pin.

1 = The USART drives the SCK pin.

USART Interrupt Enable Register

Name: US_IER
Access Type: Write-only
Offset: 0x08

| | | | | | | | |
|--------|--------|------|-------|-------|-------|---------|---------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| COMMRX | COMMTX | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: Enable RXRDY Interrupt (Code Label US_RXRDY)**
 0 = No effect.
 1 = Enables RXRDY Interrupt.
- **TXRDY: Enable TXRDY Interrupt (Code Label US_TXRDY)**
 0 = No effect.
 1 = Enables TXRDY Interrupt.
- **RXBRK: Enable Receiver Break Interrupt (Code Label US_RXBRK)**
 0 = No effect.
 1 = Enables Receiver Break Interrupt.
- **ENDRX: Enable End of Receive Transfer Interrupt (Code Label US_ENDRX)**
 0 = No effect.
 1 = Enables End of Receive Transfer Interrupt.
- **ENDTX: Enable End of Transmit Transfer Interrupt (Code Label US_ENDTX)**
 0 = No effect.
 1 = Enables End of Transmit Transfer Interrupt.
- **OVRE: Enable Overrun Error Interrupt (Code Label US_OVRE)**
 0 = No effect.
 1 = Enables Overrun Error Interrupt.
- **FRAME: Enable Framing Error Interrupt (Code Label US_FRAME)**
 0 = No effect.
 1 = Enables Framing Error Interrupt.
- **PARE: Enable Parity Error Interrupt (Code Label US_PARE)**
 0 = No effect.
 1 = Enables Parity Error Interrupt.
- **TIMEOUT: Enable Time-out Interrupt (Code Label US_TIMEOUT)**
 0 = No effect.
 1 = Enables Reception Time-out Interrupt.
- **TXEMPTY: Enable TXEMPTY Interrupt (Code Label US_TXEMPTY)**
 0 = No effect.
 1 = Enables TXEMPTY Interrupt.
- **COMMTX: Enable ARM7TDMI ICE Debug Communication Channel Transmit Interrupt**
 This bit is implemented for USART0 only.
 0 = No effect.
 1 = Enables COMMTX Interrupt
- **COMMRX: Enable ARM7TDMI ICE Debug Communication Channel Receive Interrupt**
 This bit is implemented for USART0 only.
 0 = No effect.
 1 = Enables COMMRX Interrupt



USART Interrupt Disable Register

Name: US_IDR
Access Type: Write-only
Offset: 0x0C

| | | | | | | | |
|--------|--------|------|-------|-------|-------|---------|---------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: Disable RXRDY Interrupt (Code Label US_RXRDY)**
 0 = No effect.
 1 = Disables RXRDY Interrupt.
- **TXRDY: Disable TXRDY Interrupt (Code Label US_TXRDY)**
 0 = No effect.
 1 = Disables TXRDY Interrupt.
- **RXBRK: Disable Receiver Break Interrupt (Code Label US_RXBRK)**
 0 = No effect.
 1 = Disables Receiver Break Interrupt.
- **ENDRX: Disable End of Receive Transfer Interrupt (Code Label US_ENDRX)**
 0 = No effect.
 1 = Disables End of Receive Transfer Interrupt.
- **ENDTX: Disable End of Transmit Transfer Interrupt (Code Label US_ENDTX)**
 0 = No effect.
 1 = Disables End of Transmit Transfer Interrupt.
- **OVRE: Disable Overrun Error Interrupt (Code Label US_OVRE)**
 0 = No effect.
 1 = Disables Overrun Error Interrupt.
- **FRAME: Disable Framing Error Interrupt (Code Label US_FRAME)**
 0 = No effect.
 1 = Disables Framing Error Interrupt.
- **PARE: Disable Parity Error Interrupt (Code Label US_PARE)**
 0 = No effect.
 1 = Disables Parity Error Interrupt.
- **TIMEOUT: Disable Time-out Interrupt (Code Label US_TIMEOUT)**
 0 = No effect.
 1 = Disables Receiver Time-out Interrupt.
- **TXEMPTY: Disable TXEMPTY Interrupt (Code Label US_TXEMPTY)**
 0 = No effect.
 1 = Disables TXEMPTY Interrupt.
- **COMMTX: Disable ARM7TDMI ICE Debug Communication Channel Transmit Interrupt**
 This bit is implemented for USART0 only.
 0 = No effect.
 1 = Disables COMMTX Interrupt.
- **COMMRX: Disable ARM7TDMI ICE Debug Communication Channel Receive Interrupt**
 This bit is implemented for USART0 only.
 0 = No effect.
 1 = Disables COMMRX Interrupt.

USART Interrupt Mask Register

Name: US_IMR
Access Type: Read-only
Offset: 0x10
Reset Value: 0x0

| | | | | | | | |
|--------|--------|------|-------|-------|-------|---------|---------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: RXRDY Interrupt Mask (Code Label US_RXRDY)**
 0 = RXRDY Interrupt is Disabled.
 1 = RXRDY Interrupt is Enabled.
- **TXRDY: TXRDY Interrupt Mask (Code Label US_TXRDY)**
 0 = TXRDY Interrupt is Disabled.
 1 = TXRDY Interrupt is Enabled.
- **RXBRK: Receiver Break Interrupt Mask (Code Label US_RXBRK)**
 0 = Receiver Break Interrupt is Disabled.
 1 = Receiver Break Interrupt is Enabled.
- **ENDRX: End of Receive Transfer Interrupt Mask (Code Label US_ENDRX)**
 0 = End of Receive Transfer Interrupt is Disabled.
 1 = End of Receive Transfer Interrupt is Enabled.
- **ENDTX: End of Transmit Transfer Interrupt Mask (Code Label US_ENDTX)**
 0 = End of Transmit Transfer Interrupt is Disabled.
 1 = End of Transmit Transfer Interrupt is Enabled.
- **OVRE: Overrun Error Interrupt Mask (Code Label US_OVRE)**
 0 = Overrun Error Interrupt is Disabled.
 1 = Overrun Error Interrupt is Enabled.
- **FRAME: Framing Error Interrupt Mask (Code Label US_FRAME)**
 0 = Framing Error Interrupt is Disabled.
 1 = Framing Error Interrupt is Enabled.
- **PARE: Parity Error Interrupt Mask (Code Label US_PARE)**
 0 = Parity Error Interrupt is Disabled.
 1 = Parity Error Interrupt is Enabled.
- **TIMEOUT: Time-out Interrupt Mask (Code Label US_TIMEOUT)**
 0 = Receive Time-out Interrupt is Disabled.
 1 = Receive Time-out Interrupt is Enabled.
- **TXEMPTY: TXEMPTY Interrupt Mask (Code Label US_TXEMPTY)**
 0 = TXEMPTY Interrupt is Disabled.
 1 = TXEMPTY Interrupt is Enabled.
- **COMMTX: ARM7TDMI ICE Debug Communication Channel Transmit Interrupt Mask**
 This bit is implemented for USART0 only.
 0 = COMMTX Interrupt is Disabled
 1 = COMMTX Interrupt is Enabled
- **COMMRX: ARM7TDMI ICE Debug Communication Channel Receive Interrupt Mask**
 This bit is implemented for USART0 only.
 0 = COMMRX Interrupt is Disabled
 1 = COMMRX Interrupt is Enabled

USART Channel Status Register

Name: US_CSR
Access Type: Read-only
Offset: 0x14
Reset Value: 0x18

| | | | | | | | |
|--------|--------|------|-------|-------|-------|---------|---------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: Receiver Ready (Code Label `US_RXRDY`)**

0 = No complete character has been received since the last read of the US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.
 1 = At least one complete character has been received and the US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (Code Label `US_TXRDY`)**

0 = A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.
 1 = US_THR is empty and there is no Break request pending TSR availability.
 Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one.

- **RXBRK: Break Received/End of Break (Code Label `US_RXBRK`)**

0 = No Break Received nor End of Break detected since the last “Reset Status Bits” command in the Control Register.
 1 = Break Received or End of Break detected since the last “Reset Status Bits” command in the Control Register.

- **ENDRX: End of Receive Transfer (Code Label `US_ENDRX`)**

0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is inactive.
 1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is active.

- **ENDTX: End of Transmit Transfer (Code Label `US_ENDTX`)**

0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is inactive.
 1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is active.

- **OVRE: Overrun Error (Code Label `US_OVRE`)**

0 = No byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last “Reset Status Bits” command.
 1 = At least one byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last “Reset Status Bits” command.

- **FRAME: Framing Error (Code Label `US_FRAME`)**

0 = No stop bit has been detected low since the last “Reset Status Bits” command.
 1 = At least one stop bit has been detected low since the last “Reset Status Bits” command.

- **PARE: Parity Error (Code Label `US_PARE`)**

1 = At least one parity bit has been detected false (or a parity bit high in multi-drop mode) since the last “Reset Status Bits” command.
 0 = No parity bit has been detected false (or a parity bit high in multi-drop mode) since the last “Reset Status Bits” command.

- **TIMEOUT: Receiver Time-out (Code Label `US_TIMEOUT`)**

0 = There has not been a time-out since the last “Start Time-out” command or the Time-out Register is 0.
 1 = There has been a time-out since the last “Start Time-out” command.

• **TXEMPTY: Transmitter Empty (Code Label US_TXEMPTY)**

0 = There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.
 1 = There are no characters in either US_THR or the Transmit Shift Register. TXEMPTY is 1 after Parity, Stop Bit and Time-guard have been transmitted. TXEMPTY is 1 after stop bit has been sent, or after Time-guard has been sent if US_TTGR is not 0.
 Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one, if the transmitter is disabled.

• **COMMTX: ARM7TDMI ICE Debug Communication Channel Transmit Status**

For USART0 only. Refer to the ARM7TDMI Datasheet for a complete description of this flag.

• **COMMRX: ARM7TDMI ICE Debug Communication Channel Receive Status**

For USART0 only. Refer to the ARM7TDMI Datasheet for a complete description of this flag.

USART Receiver Holding Register

Name: US_RHR
Access Type: Read-only
Offset: 0x18
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | RXCHR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXCHR | | | | | | | |

• **RXCHR: Received Character**

Last character received if RXRDY is set. When number of data bits is less than 9 bits, the bits are right-aligned. All unused bits read zero.

USART Transmitter Holding Register

Name: US_THR
Access Type: Write-only
Offset: 0x1C

| | | | | | | | |
|-------|----|----|----|----|----|----|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | TXCHR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXCHR | | | | | | | |

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set. When number of data bits is less than 9 bits, the bits are right-aligned.

USART Baud Rate Generator Register

Name: US_BRGR
Access Type: Read/Write
Offset: 0x20
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CD | | | | | | | |

- **CD: Clock Divisor**

This register has no effect if Synchronous Mode is selected with an external clock.

| CD | |
|------------|--|
| 0 | Disables Clock |
| 1 | Clock Divisor Bypass ⁽¹⁾ |
| 2 to 65535 | Baud Rate (Asynchronous Mode ⁽²⁾) = Selected Clock/(16 x CD) Baud Rate (Synchronous Mode) = Selected Clock/CD |

Notes: 1. In Synchronous mode, the value programmed must be even to ensure a 50:50 mark:space ratio.
 2. Clock divisor bypass (CD = 1) must not be used when internal clock MCK is selected (USCLKS = 0).

USART Receiver Time-out Register

Name: US_RTOR
Access Type: Read/Write
Offset: 0x24
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TO | | | | | | | |

• **TO: Time-out Value**

When a value is written to this register, a Start Time-out Command is automatically performed.

| TO | |
|--------|--|
| 0 | Disables the RX Time-out function. |
| 1- 255 | The Time-out counter is loaded with TO when the Start Time-out Command is given or when each new data character is received (after reception has started). |

Time-out duration = TO x 4 x Bit period

USART Transmitter Time-guard Register

Name: US_TTGR
Access Type: Read/Write
Offset: 0x28
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TG | | | | | | | |

• **TG: Time-guard Value**

| TG | |
|---------|--|
| 0 | Disables the TX Time-guard function. |
| 1 - 255 | TXD is inactive high after the transmission of each character for the time-guard duration. |

Time-guard duration = TG x Bit period

USART Receive Pointer Register

Name: US_RPR
Access Type: Read/Write
Offset: 0x30
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| RXPTR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXPTR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXPTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXPTR | | | | | | | |

- **RXPTR: Receive Pointer**

RXPTR must be loaded with the address of the receive buffer.

USART Receive Counter Register

Name: US_RCR
Access Type: Read/Write
Offset: 0x34
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXCTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXCTR | | | | | | | |

- **RXCTR: Receive Counter**

RXCTR must be loaded with the size of the receive buffer.
 0: Stop Peripheral Data Transfer dedicated to the receiver.
 1-65535: Start Peripheral Data transfer if RXRDY is active.

USART Transmit Pointer Register

Name: US_TPR
Access Type: Read/Write
Offset: 0x38
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| TXPTR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXPTR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXPTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXPTR | | | | | | | |

- **TXPTR: Transmit Pointer**
 TXPTR must be loaded with the address of the transmit buffer.

USART Transmit Counter Register

Name: US_TCR
Access Type: Read/Write
Offset: 0x3C
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXCTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXCTR | | | | | | | |

- **TXCTR: Transmit Counter**
 TXCTR must be loaded with the size of the transmit buffer.
 0: Stop Peripheral Data Transfer dedicated to the transmitter.
 1-65535: Start Peripheral Data transfer if TXRDY is active.

TC: Timer/Counter

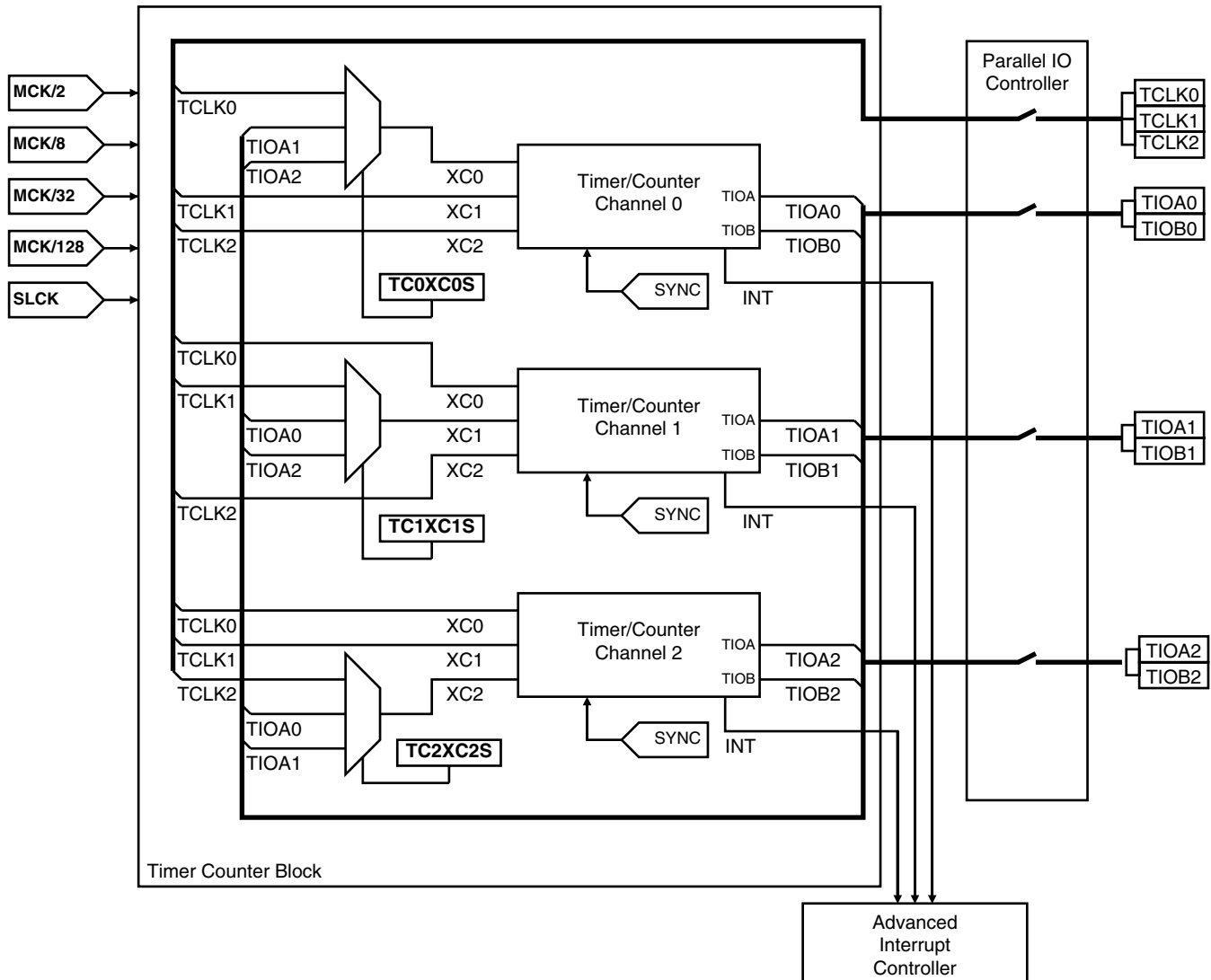
The AT91M42800A features two Timer/Counter blocks, each containing three identical 16-bit Timer/Counter channels. Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each Timer/Counter (TC) channel has 3 external clock inputs, 5 internal clock inputs, and 2 multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts via the AIC (Advanced Interrupt Controller).

The Timer/Counter block has two global registers which act upon all three TC channels. The Block Control Register allows the three channels to be started simultaneously with the same instruction. The Block Mode Register defines the external clock inputs for each Timer/Counter channel, allowing them to be chained.

Each Timer/Counter block operates independently and has a complete set of block and channel registers. Since they are identical in operation, only one block is described below (see Timer/Counter Description on page 144). The internal configuration of a single Timer/Counter Block is shown in Figure 55.

Figure 55. TC Block Diagram



Signal Name Description^(1, 2)

| Channel Signals | Description |
|---------------------|--|
| XC0, XC1, XC2 | External Clock Inputs |
| TIOA | Capture Mode: General-purpose Input Waveform Mode: General-purpose Output |
| TIOB | Capture Mode: General-purpose Input Waveform Mode: General-purpose Input/Output |
| INT | Interrupt Signal Output |
| SYNC | Synchronization Input Signal |
| Block 0 Signals | Description |
| TCLK0, TCLK1, TCLK2 | External Clock Inputs for Channels 0, 1, 2 |
| TIOA0 | TIOA Signal for Channel 0 |
| TIOB0 | TIOB Signal for Channel 0 |
| TIOA1 | TIOA Signal for Channel 1 |
| TIOB1 | TIOB Signal for Channel 1 |
| TIOA2 | TIOA Signal for Channel 2 |
| TIOB2 | TIOB Signal for Channel 2 |
| Block 1 Signals | Description |
| TCLK3, TCLK4, TCLK5 | External Clock Inputs for Channels 3, 4, 5 |
| TIOA3 | TIOA Signal for Channel 3 |
| TIOB3 | TIOB Signal for Channel 3 |
| TIOA4 | TIOA Signal for Channel 4 |
| TIOB4 | TIOB Signal for Channel 4 |
| TIOA5 | TIOA Signal for Channel 5 |
| TIOB5 | TIOB Signal for Channel 5 |

- Notes:
1. After a hardware reset, the TC clock is disabled by default (see “PMC: Power Management Controller” on page 55). The user must configure the Power Management Controller before any access to the User Interface of the TC.
 2. After a hardware reset, the Timer/Counter block pins are controlled by the PIO Controller. They must be configured to be controlled by the peripheral before being used.

Timer/Counter Description

Counter

Each Timer/Counter channel is identical in operation. The registers for channel programming are listed in Table 19.

Each Timer/Counter channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the input clock. When the counter reaches the value 0xFFFF and passes to 0x0000, an overflow occurs and the bit COVFS in TC_SR (Status Register) is set.

The current value of the counter is accessible in real time by reading TC_CV. The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the clock.

Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block mode).

Each channel can independently select an internal or external clock source for its counter:

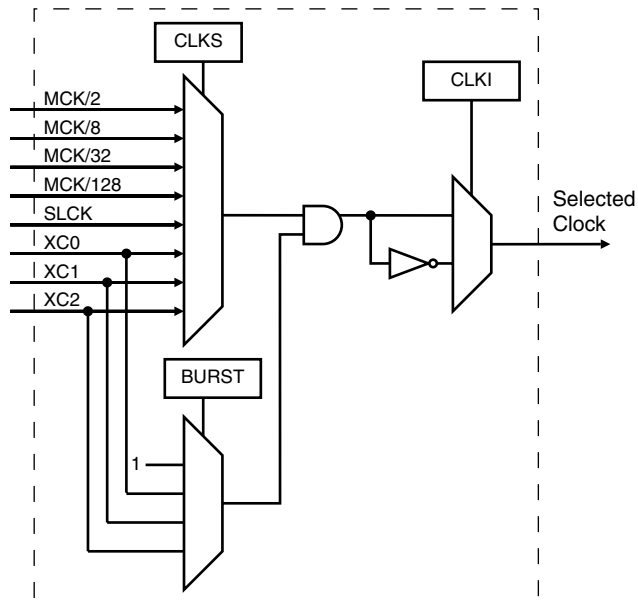
- Internal clock signals: MCK/2, MCK/8, MCK/32, MCK/128 and Slow Clock SLCK
- External clock signals: XC0, XC1 or XC2

The selected clock can be inverted with the CLKI bit in TC_CMR (Channel mode). This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the system clock (MCK) period. The external clock frequency must be at least 2.5 times lower than the system clock.

Figure 56. Clock Selection

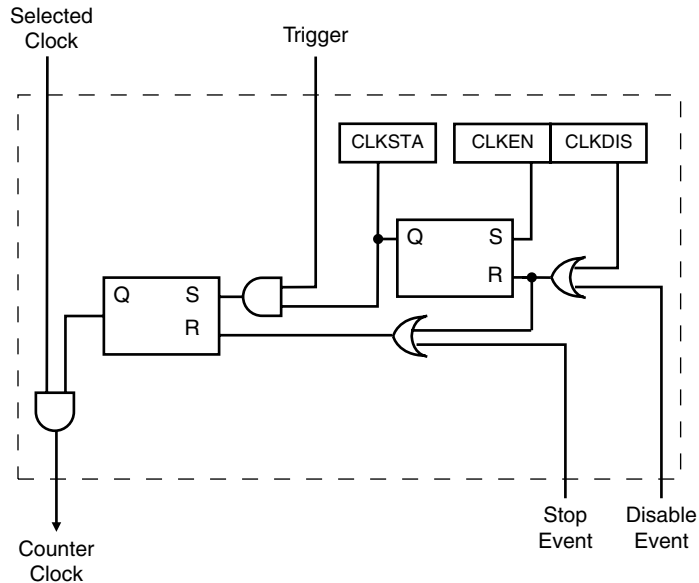


Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped.

- The clock can be **enabled** or **disabled** by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in TC_CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be **started** or **stopped**: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode (LDBSTOP = 1 in TC_CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have effect only if the clock is enabled.

Figure 57. Clock Control



Timer/Counter Operating Modes

Each Timer/Counter channel can independently operate in two different modes:

- Capture mode allows measurement on signals
- Waveform mode allows wave generation

The Timer/Counter mode is programmed with the WAVE bit in the TC Mode Register. In Capture mode, TIOA and TIOB are configured as inputs. In Waveform mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.

- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in TC_CMR.

The Timer/Counter channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRIG in TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the system clock (MCK) period in order to be detected.

Capture Operating Mode

This mode is entered by clearing the WAVE parameter in TC_CMR (Channel Mode Register). Capture Mode allows the TC Channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as input.

Figure 58 shows the configuration of the TC Channel when programmed in Capture Mode.

Capture Registers A and B (RA and RB)

Registers A and B are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The parameter LDRA in TC_CMR defines the TIOA edge for the loading of register A, and the parameter LDRB defines the TIOA edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

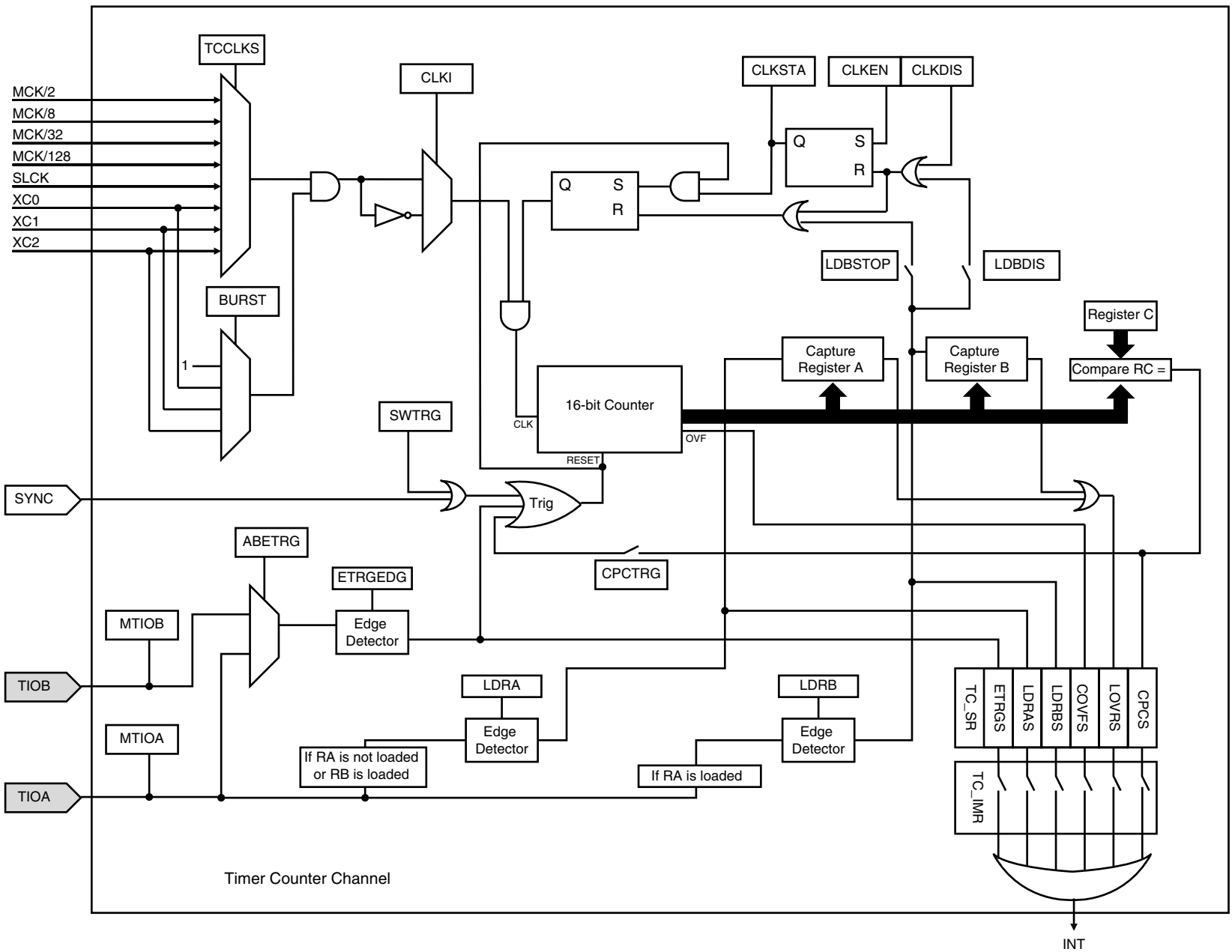
Bit ABETRIG in TC_CMR selects input signal TIOA or TIOB as an external trigger. Parameter ETRGEDG defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Status Register

The following bits in the status register are significant in Capture Operating mode.

- **CPCS: RC Compare Status**
There has been an RC Compare match at least once since the last read of the status
- **COVFS: Counter Overflow Status**
The counter has attempted to count past \$FFFF since the last read of the status
- **LOVRS: Load Overrun Status**
RA or RB has been loaded at least twice without any read of the corresponding register, since the last read of the status
- **LDRAS: Load RA Status**
RA has been loaded at least once without any read, since the last read of the status
- **LDRBS: Load RB Status**
RB has been loaded at least once without any read, since the last read of the status
- **ETRGS: External Trigger Status**
An external trigger on TIOA or TIOB has been detected since the last read of the status

Figure 58. Capture Mode



Waveform Operating Mode

This mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

Waveform Operating Mode allows the TC Channel to generate 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or to generate different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as output and TIOB is defined as output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 59 shows the configuration of the TC Channel when programmed in Waveform Operating Mode.

Compare Register A, B and C (RA, RB, and RC)

In Waveform Operating Mode, RA, RB and RC are all used as compare registers.

RA Compare is used to control the TIOA output. RB Compare is used to control the TIOB (if configured as output). RC Compare can be programmed to control TIOA and/or TIOB outputs.

RC Compare can also stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

As in Capture Mode, RC Compare can also generate a trigger if CPCTRG = 1. Trigger resets the counter so RC can control the period of PWM waveforms.

External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The parameter EEVT in TC_CMR selects the external trigger. The parameter EEVT-EDG defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as output and the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRIG in TC_CMR.

As in Capture Mode, the SYNC signal, the software trigger and the RC compare trigger are also available as triggers.

Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

The tables below show which parameter in TC_CMCR is used to define the effect of each event.

| Parameter | TIOA Event |
|-----------|------------------|
| ASWTRG | Software Trigger |
| AAEVT | External Event |
| ACPC | RC Compare |
| ACPA | RA Compare |

| Parameter | TIOB Event |
|-----------|------------------|
| BSWTRG | Software Trigger |
| BEEVT | External Event |
| BCPC | RC Compare |
| BCPB | RB Compare |

If two or more events occur at the same time, the priority level is defined as follows:

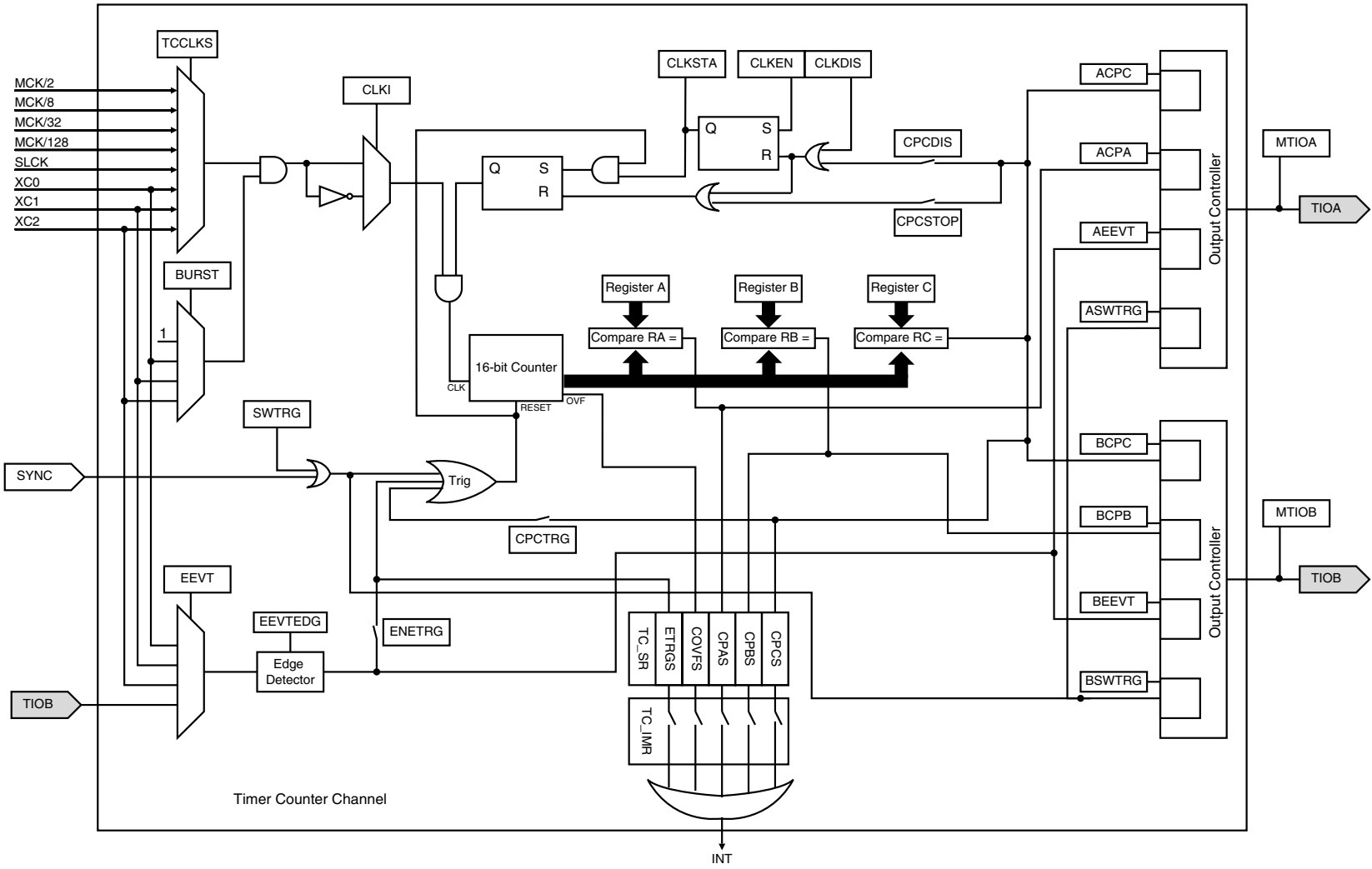
1. Software Trigger
2. External Event
3. RC Compare
4. RA or RB Compare

Status

The following bits in the status register are significant in Waveform mode:

- CPAS: RA Compare Status
There has been a RA Compare match at least once since the last read of the status
- CPBS: RB Compare Status
There has been a RB Compare match at least once since the last read of the status
- CPCS: RC Compare Status
There has been a RC Compare match at least once since the last read of the status
- COVFS: Counter Overflow
Counter has attempted to count past \$FFFF since the last read of the status
- ETRGS: External Trigger
External trigger has been detected since the last read of the status

Figure 59. Waveform Mode



TC User Interface

TC Block 0 Base Address: 0xFFFD0000 (Code Label TCB0_BASE)

TC Block 1 Base Address: 0xFFFD4000 (Code Label TCB1_BASE)

Table 18. TC Global Memory Map

| Offset | Channel/Register | Name | Access | Reset State |
|--------|---------------------------|--------|--------------|-------------|
| 0x00 | TC Channel 0 | | See Table 19 | |
| 0x40 | TC Channel 1 | | See Table 19 | |
| 0x80 | TC Channel 2 | | See Table 19 | |
| 0xC0 | TC Block Control Register | TC_BCR | Write-only | – |
| 0xC4 | TC Block Mode Register | TC_BMR | Read/Write | 0 |

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the TC block. TC Channels are controlled by the registers listed in Table 19. The offset of each of the Channel registers in Table 19 is in relation to the offset of the corresponding channel as mentioned in Table 18.

Table 19. TC Channel Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|----------------------------|--------|---------------------------|-------------|
| 0x00 | Channel Control Register | TC_CCR | Write-only | – |
| 0x04 | Channel Mode Register | TC_CMR | Read/Write | 0 |
| 0x08 | Reserved | | | – |
| 0x0C | Reserved | | | – |
| 0x10 | Counter Value | TC_CV | Read/Write | 0 |
| 0x14 | Register A | TC_RA | Read/Write ⁽¹⁾ | 0 |
| 0x18 | Register B | TC_RB | Read/Write ⁽¹⁾ | 0 |
| 0x1C | Register C | TC_RC | Read/Write | 0 |
| 0x20 | Status Register | TC_SR | Read-only | – |
| 0x24 | Interrupt Enable Register | TC_IER | Write-only | – |
| 0x28 | Interrupt Disable Register | TC_IDR | Write-only | – |
| 0x2C | Interrupt Mask Register | TC_IMR | Read-only | 0 |

Note: 1. Read-only if WAVE = 0

TC Block Control Register

Register Name: TC_BCR
Access Type: Write-only
Offset: 0xC0

| | | | | | | | |
|----|----|----|----|----|----|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | SYNC |

• **SYNC: Synchro Command (Code Label TC_SYNC)**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.



TC Block Mode Register

Register Name: TC_BMR
Access Type: Read/Write
Offset: 0xC4
Reset Value: 0x0

| | | | | | | | |
|----|----|---------|----|---------|----|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | TC2XC2S | | TC1XC1S | | TC0XC0S | |

• TC0XC0S: External Clock Signal 0 Selection

| TC0XC0S | | Signal Connected to XC0 | Code Label: TC_TC0XC0S |
|---------|---|-------------------------|------------------------|
| 0 | 0 | TCLK0 | TC_TCLK0XC0 |
| 0 | 1 | None | TC_NONEXC0 |
| 1 | 0 | TIOA1 | TC_TIOA1XC0 |
| 1 | 1 | TIOA2 | TC_TIOA2XC0 |

• TC1XC1S: External Clock Signal 1 Selection

| TC1XC1S | | Signal Connected to XC1 | Code Label: TC_TC1XC1S |
|---------|---|-------------------------|------------------------|
| 0 | 0 | TCLK1 | TC_TCLK1XC1 |
| 0 | 1 | None | TC_NONEXC1 |
| 1 | 0 | TIOA0 | TC_TIOA0XC1 |
| 1 | 1 | TIOA2 | TC_TIOA2XC1 |

• TC2XC2S: External Clock Signal 2 Selection

| TC2XC2S | | Signal Connected to XC2 | Code Label: TC_TC2XC2S |
|---------|---|-------------------------|------------------------|
| 0 | 0 | TCLK2 | TC_TCLK2XC2 |
| 0 | 1 | None | TC_NONEXC2 |
| 1 | 0 | TIOA0 | TC_TIOA0XC2 |
| 1 | 1 | TIOA1 | TC_TIOA1XC2 |

TC Channel Control Register

Register Name: TC_CCR
Access Type: Write-only
Offset: 0x00

| | | | | | | | |
|----|----|----|----|----|-------|--------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | SWTRG | CLKDIS | CLKEN |

- **CLKEN: Counter Clock Enable Command (Code Label TC_CLKEN)**
 0 = No effect.
 1 = Enables the clock if CLKDIS is not 1.
- **CLKDIS: Counter Clock Disable Command (Code Label TC_CLKDIS)**
 0 = No effect.
 1 = Disables the clock.
- **SWTRG: Software Trigger Command (Code Label TC_SWTRG)**
 0 = No effect.
 1 = A software trigger is performed: the counter is reset and clock is started.

TC Channel Mode Register: Capture Mode

Register Name: TC_CMCR
Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

| | | | | | | | |
|--------|---------|-------|----|------|--------|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | LDRB | | LDRA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE=0 | CPCTRG | – | – | – | ABETRG | ETRGEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LDBDIS | LDBSTOP | BURST | | CLKI | TCCLKS | | |

• TCCLKS: Clock Selection

| TCCLKS | | | Clock Selected | Code Label: TC_CLKS |
|--------|---|---|----------------|---------------------|
| 0 | 0 | 0 | MCK/2 | TC_CLKS_MCK2 |
| 0 | 0 | 1 | MCK/8 | TC_CLKS_MCK8 |
| 0 | 1 | 0 | MCK/32 | TC_CLKS_MCK32 |
| 0 | 1 | 1 | MCK/128 | TC_CLKS_MCK128 |
| 1 | 0 | 0 | SLCK | TC_CLKS_SLCK |
| 1 | 0 | 1 | XC0 | TC_CLKS_XC0 |
| 1 | 1 | 0 | XC1 | TC_CLKS_XC1 |
| 1 | 1 | 1 | XC2 | TC_CLKS_XC2 |

• CLKI: Clock Invert (Code Label TC_CLKI)

0 = Counter is incremented on rising edge of the clock.
 1 = Counter is incremented on falling edge of the clock.

• BURST: Burst Signal Selection

| BURST | | Selected BURST | Code Label: TC_BURST |
|-------|---|---|----------------------|
| 0 | 0 | The clock is not gated by an external signal. | TC_BURST_NONE |
| 0 | 1 | XC0 is ANDed with the selected clock. | TC_BURST_XC0 |
| 1 | 0 | XC1 is ANDed with the selected clock. | TC_BURST_XC1 |
| 1 | 1 | XC2 is ANDed with the selected clock. | TC_BURST_XC2 |

• LDBSTOP: Counter Clock Stopped with RB Loading (Code Label TC_LDBSTOP)

0 = Counter clock is not stopped when RB loading occurs.
 1 = Counter clock is stopped when RB loading occurs.

• LDBDIS: Counter Clock Disable with RB Loading (Code Label TC_LDBDIS)

0 = Counter clock is not disabled when RB loading occurs.
 1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

| ETRGEDG | | Edge | Code Label: TC_ETRGEDG |
|---------|---|--------------|-------------------------|
| 0 | 0 | None | TC_ETRGEDG_EDGE_NONE |
| 0 | 1 | Rising edge | TC_ETRGEDG_RISING_EDGE |
| 1 | 0 | Falling edge | TC_ETRGEDG_FALLING_EDGE |
| 1 | 1 | Each edge | TC_ETRGEDG_BOTH_EDGE |

- **ABETRG: TIOA or TIOB External Trigger Selection**

| ABETRG | Selected ABETRG | Code Label: TC_ABETRG |
|--------|--------------------------------------|-----------------------|
| 0 | TIOB is used as an external trigger. | TC_ABETRG_TIOB |
| 1 | TIOA is used as an external trigger. | TC_ABETRG_TIOA |

- **CPCTR: RC Compare Trigger Enable (Code Label TC_CPCTR)**

0 = RC Compare has no effect on the counter and its clock.
 1 = RC Compare resets the counter and starts the counter clock.

- **WAVE = 0 (Code Label TC_WAVE)**

0 = Capture Mode is enabled.
 1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

| LDRA | | Edge | Code Label: TC_LDRA |
|------|---|----------------------|----------------------|
| 0 | 0 | None | TC_LDRA_EDGE_NONE |
| 0 | 1 | Rising edge of TIOA | TC_LDRA_RISING_EDGE |
| 1 | 0 | Falling edge of TIOA | TC_LDRA_FALLING_EDGE |
| 1 | 1 | Each edge of TIOA | TC_LDRA_BOTH_EDGE |

- **LDRB: RB Loading Selection**

| LDRB | | Edge | Code Label: TC_LDRB |
|------|---|----------------------|----------------------|
| 0 | 0 | None | TC_LDRB_EDGE_NONE |
| 0 | 1 | Rising edge of TIOA | TC_LDRB_RISING_EDGE |
| 1 | 0 | Falling edge of TIOA | TC_LDRB_FALLING_EDGE |
| 1 | 1 | Each edge of TIOA | TC_LDRB_BOTH_EDGE |



TC Channel Mode Register: Waveform Mode

Register Name: TC_CM
Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

| | | | | | | | |
|--------|---------|-------|--------|------|--------|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| BSWTRG | | BEEVT | | BCPC | | BCPB | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ASWTRG | | AEEVT | | ACPC | | ACPA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE=1 | CPCTRG | – | ENETRG | EEVT | | EEVTEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPCDIS | CPCSTOP | BURST | | CLKI | TCCLKS | | |

• TCCLKS: Clock Selection

| TCCLKS | | | Clock Selected | Code Label: TC_CLKS |
|--------|---|---|----------------|---------------------|
| 0 | 0 | 0 | MCK/2 | TC_CLKS_MCK2 |
| 0 | 0 | 1 | MCK/8 | TC_CLKS_MCK8 |
| 0 | 1 | 0 | MCK/32 | TC_CLKS_MCK32 |
| 0 | 1 | 1 | MCK/128 | TC_CLKS_MCK128 |
| 1 | 0 | 0 | SLCK | TC_CLKS_SLCK |
| 1 | 0 | 1 | XC0 | TC_CLKS_XC0 |
| 1 | 1 | 0 | XC1 | TC_CLKS_XC1 |
| 1 | 1 | 1 | XC2 | TC_CLKS_XC2 |

• CLKI: Clock Invert (Code Label TC_CLKI)

0 = Counter is incremented on rising edge of the clock.
 1 = Counter is incremented on falling edge of the clock.

• BURST: Burst Signal Selection

| BURST | | Selected BURST | Code Label: TC_BURST |
|-------|---|---|----------------------|
| 0 | 0 | The clock is not gated by an external signal. | TC_BURST_NONE |
| 0 | 1 | XC0 is ANDed with the selected clock. | TC_BURST_XC0 |
| 1 | 0 | XC1 is ANDed with the selected clock. | TC_BURST_XC1 |
| 1 | 1 | XC2 is ANDed with the selected clock. | TC_BURST_XC2 |

• CPCSTOP: Counter Clock Stopped with RC Compare (Code Label TC_CPCSTOP)

0 = Counter clock is not stopped when counter reaches RC.
 1 = Counter clock is stopped when counter reaches RC.

• CPCDIS: Counter Clock Disable with RC Compare (Code Label TC_CPCDIS)

0 = Counter clock is not disabled when counter reaches RC.
 1 = Counter clock is disabled when counter reaches RC.

• **EEVTEDG: External Event Edge Selection**

| EEVTEDG | | Edge | Code Label: TC_EEVTEDG |
|---------|---|--------------|-------------------------|
| 0 | 0 | None | TC_EEVTEDG_EDGE_NONE |
| 0 | 1 | Rising edge | TC_EEVTEDG_RISING_EDGE |
| 1 | 0 | Falling edge | TC_EEVTEDG_FALLING_EDGE |
| 1 | 1 | Each edge | TC_EEVTEDG_BOTH_EDGE |

• **EEVT: External Event Selection**

| EEVT | | Signal Selected as External Event | TIOB Direction | Code Label: TC_EEVT |
|------|---|-----------------------------------|----------------------|---------------------|
| 0 | 0 | TIOB | Input ⁽¹⁾ | TC_EEVT_TIOB |
| 0 | 1 | XC0 | Output | TC_EEVT_XC0 |
| 1 | 0 | XC1 | Output | TC_EEVT_XC1 |
| 1 | 1 | XC2 | Output | TC_EEVT_XC2 |

Note: If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms.

• **ENETRГ: External Event Trigger Enable (Code Label TC_ENETRГ)**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

• **CPCTRГ: RC Compare Trigger Enable (Code Label TC_CPCTRГ)**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

• **WAVE = 1 (Code Label TC_WAVE)**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

• **ACPA: RA Compare Effect on TIOA**

| ACPA | | Effect | Code Label: TC_ACPA |
|------|---|--------|-----------------------|
| 0 | 0 | None | TC_ACPA_OUTPUT_NONE |
| 0 | 1 | Set | TC_ACPA_SET_OUTPUT |
| 1 | 0 | Clear | TC_ACPA_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ACPA_TOGGLE_OUTPUT |

• **ACPC: RC Compare Effect on TIOA**

| ACPC | | Effect | Code Label: TC_ACPC |
|------|---|--------|-----------------------|
| 0 | 0 | None | TC_ACPC_OUTPUT_NONE |
| 0 | 1 | Set | TC_ACPC_SET_OUTPUT |
| 1 | 0 | Clear | TC_ACPC_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ACPC_TOGGLE_OUTPUT |

- **AEEVT: External Event Effect on TIOA**

| AEEVT | | Effect | Code Label: TC_AEEVT |
|-------|---|--------|------------------------|
| 0 | 0 | None | TC_AEEVT_OUTPUT_NONE |
| 0 | 1 | Set | TC_AEEVT_SET_OUTPUT |
| 1 | 0 | Clear | TC_AEEVT_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_AEEVT_TOGGLE_OUTPUT |

- **ASWTRG: Software Trigger Effect on TIOA**

| ASWTRG | | Effect | Code Label: TC_ASWTRG |
|--------|---|--------|-------------------------|
| 0 | 0 | None | TC_ASWTRG_OUTPUT_NONE |
| 0 | 1 | Set | TC_ASWTRG_SET_OUTPUT |
| 1 | 0 | Clear | TC_ASWTRG_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ASWTRG_TOGGLE_OUTPUT |

- **BCPB: RB Compare Effect on TIOB**

| BCPB | | Effect | Code Label: TC_BCPB |
|------|---|--------|-----------------------|
| 0 | 0 | None | TC_BCPB_OUTPUT_NONE |
| 0 | 1 | Set | TC_BCPB_SET_OUTPUT |
| 1 | 0 | Clear | TC_BCPB_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BCPB_TOGGLE_OUTPUT |

- **BCPC: RC Compare Effect on TIOB**

| BCPC | | Effect | Code Label: TC_BCPC |
|------|---|--------|-----------------------|
| 0 | 0 | None | TC_BCPC_OUTPUT_NONE |
| 0 | 1 | Set | TC_BCPC_SET_OUTPUT |
| 1 | 0 | Clear | TC_BCPC_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BCPC_TOGGLE_OUTPUT |

- **BEEVT: External Event Effect on TIOB**

| BEEVT | | Effect | Code Label: TC_BEEVT |
|-------|---|--------|------------------------|
| 0 | 0 | None | TC_BEEVT_OUTPUT_NONE |
| 0 | 1 | Set | TC_BEEVT_SET_OUTPUT |
| 1 | 0 | Clear | TC_BEEVT_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BEEVT_TOGGLE_OUTPUT |

- **BSWTRG: Software Trigger Effect on TIOB**

| BSWTRG | | Effect | Code Label: TC_BSWTRG |
|--------|---|--------|-------------------------|
| 0 | 0 | None | TC_BSWTRG_OUTPUT_NONE |
| 0 | 1 | Set | TC_BSWTRG_SET_OUTPUT |
| 1 | 0 | Clear | TC_BSWTRG_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BSWTRG_TOGGLE_OUTPUT |

TC Counter Value Register

Register Name: TC_CV
Access Type: Read-only
Offset: 0x10
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CV | | | | | | | |

- **CV: Counter Value (Code Label TC_CV)**
CV contains the counter value in real time.

TC Register A

Register Name: TC_RA
Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1
Offset: 0x14
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RA | | | | | | | |

- **RA: Register A (Code Label TC_RA)**
RA contains the Register A value in real time.

TC Register B

Register Name: TC_RB
Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1
Offset: 0x18
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RB | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RB | | | | | | | |

- **RB: Register B (Code Label $\tau\text{C_RB}$)**
RB contains the Register B value in real time.

TC Register C

Register Name: TC_RC
Access Type: Read/Write
Offset: 0x1C
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RC | | | | | | | |

- **RC: Register C (Code Label $\tau\text{C_RC}$)**
RC contains the Register C value in real time.

TC Status Register

Register Name: TC_SR
Access Type: Read-only
Offset: 0x20

| | | | | | | | |
|-------|-------|-------|------|------|-------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | MTIOB | MTIOA | CLKSTA |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow Status (Code Label TC_COVFS)**
 0 = No counter overflow has occurred since the last read of the Status Register.
 1 = A counter overflow has occurred since the last read of the Status Register.
- **LOVRS: Load Overrun Status (Code Label TC_LOVRS)**
 0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.
 1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.
- **CPAS: RA Compare Status (Code Label TC_CPAS)**
 0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.
 1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.
- **CPBS: RB Compare Status (Code Label TC_CPBS)**
 0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.
 1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.
- **CPCS: RC Compare Status (Code Label TC_CPCS)**
 0 = RC Compare has not occurred since the last read of the Status Register.
 1 = RC Compare has occurred since the last read of the Status Register.
- **LDRAS: RA Loading Status (Code Label TC_LDRAS)**
 0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.
 1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.
- **LDRBS: RB Loading Status (Code Label TC_LDRBS)**
 0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.
 1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.
- **ETRGS: External Trigger Status (Code Label TC_ETRGS)**
 0 = External trigger has not occurred since the last read of the Status Register.
 1 = External trigger has occurred since the last read of the Status Register.
- **CLKSTA: Clock Enabling Status (Code Label TC_CLKSTA)**
 0 = Clock is disabled.
 1 = Clock is enabled.
- **MTIOA: TIOA Mirror (Code Label TC_MTIOA)**
 0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.
 1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.
- **MTIOB: TIOB Mirror (Code Label TC_MTIOB)**
 0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.
 1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

TC Interrupt Enable Register

Register Name: TC_IER
Access Type: Write-only
Offset: 0x24

| | | | | | | | |
|-------|-------|-------|------|------|------|-------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow (Code Label TC_COVFS)**
 0 = No effect.
 1 = Enables the Counter Overflow Interrupt.
- **LOVRS: Load Overrun (Code Label TC_LOVRS)**
 0 = No effect.
 1 = Enables the Load Overrun Interrupt.
- **CPAS: RA Compare (Code Label TC_CPAS)**
 0 = No effect.
 1 = Enables the RA Compare Interrupt.
- **CPBS: RB Compare (Code Label TC_CPBS)**
 0 = No effect.
 1 = Enables the RB Compare Interrupt.
- **CPCS: RC Compare (Code Label TC_CPCS)**
 0 = No effect.
 1 = Enables the RC Compare Interrupt.
- **LDRAS: RA Loading (Code Label TC_LDRAS)**
 0 = No effect.
 1 = Enables the RA Load Interrupt.
- **LDRBS: RB Loading (Code Label TC_LDRBS)**
 0 = No effect.
 1 = Enables the RB Load Interrupt.
- **ETRGS: External Trigger (Code Label TC_ETRGS)**
 0 = No effect.
 1 = Enables the External Trigger Interrupt.

TC Interrupt Disable Register

Register Name: TC_IDR
Access Type: Write-only
Offset: 0x28

| | | | | | | | |
|-------|-------|-------|------|------|------|-------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow (Code Label TC_COVFS)**
 0 = No effect.
 1 = Disables the Counter Overflow Interrupt.
- **LOVRS: Load Overrun (Code Label TC_LOVRS)**
 0 = No effect.
 1 = Disables the Load Overrun Interrupt (if WAVE = 0).
- **CPAS: RA Compare (Code Label TC_CPAS)**
 0 = No effect.
 1 = Disables the RA Compare Interrupt (if WAVE = 1).
- **CPBS: RB Compare (Code Label TC_CPBS)**
 0 = No effect.
 1 = Disables the RB Compare Interrupt (if WAVE = 1).
- **CPCS: RC Compare (Code Label TC_CPCS)**
 0 = No effect.
 1 = Disables the RC Compare Interrupt.
- **LDRAS: RA Loading (Code Label TC_LDRAS)**
 0 = No effect.
 1 = Disables the RA Load Interrupt (if WAVE = 0).
- **LDRBS: RB Loading (Code Label TC_LDRBS)**
 0 = No effect.
 1 = Disables the RB Load Interrupt (if WAVE = 0).
- **ETRGS: External Trigger (Code Label TC_ETRGS)**
 0 = No effect.
 1 = Disables the External Trigger Interrupt.



TC Interrupt Mask Register

Register Name: TC_IMR
Access Type: Read-only
Offset: 0x2C
Reset Value: 0x0

| | | | | | | | |
|-------|-------|-------|------|------|------|-------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow (Code Label TC_COVFS)**
 0 = The Counter Overflow Interrupt is disabled.
 1 = The Counter Overflow Interrupt is enabled.
- **LOVRS: Load Overrun (Code Label TC_LOVRS)**
 0 = The Load Overrun Interrupt is disabled.
 1 = The Load Overrun Interrupt is enabled.
- **CPAS: RA Compare (Code Label TC_CPAS)**
 0 = The RA Compare Interrupt is disabled.
 1 = The RA Compare Interrupt is enabled.
- **CPBS: RB Compare (Code Label TC_CPBS)**
 0 = The RB Compare Interrupt is disabled.
 1 = The RB Compare Interrupt is enabled.
- **CPCS: RC Compare (Code Label TC_CPCS)**
 0 = The RC Compare Interrupt is disabled.
 1 = The RC Compare Interrupt is enabled.
- **LDRAS: RA Loading (Code Label TC_LDRAS)**
 0 = The Load RA Interrupt is disabled.
 1 = The Load RA Interrupt is enabled.
- **LDRBS: RB Loading (Code Label TC_LDRBS)**
 0 = The Load RB Interrupt is disabled.
 1 = The Load RB Interrupt is enabled.
- **ETRGS: External Trigger (Code Label TC_ETRGS)**
 0 = The External Trigger Interrupt is disabled.
 1 = The External Trigger Interrupt is enabled.

SPI: Serial Peripheral Interface

The AT91M42800A includes two SPIs which provide communication with external devices in master or slave mode. They are independent, and are referred to by the letters A and B.

Pin Description

Seven pins are associated with the SPI Interface. When not needed for the SPI function, each of these pins can be configured as a PIO. Support for an external master is provided by the PIO Controller Multi-driver option. To configure an SPI pin as open-drain to support external drivers, set the corresponding bits in the PIO_MDSR register (see page 115).

An input filter can be enabled on the SPI input pins by setting the corresponding bits in the PIO_IFSR (see page 109). The NPCS0/NSS pin can function as a peripheral chip select output or slave select input. Refer to Table 20 for a description of the SPI pins.

Figure 60. SPI Block Diagram

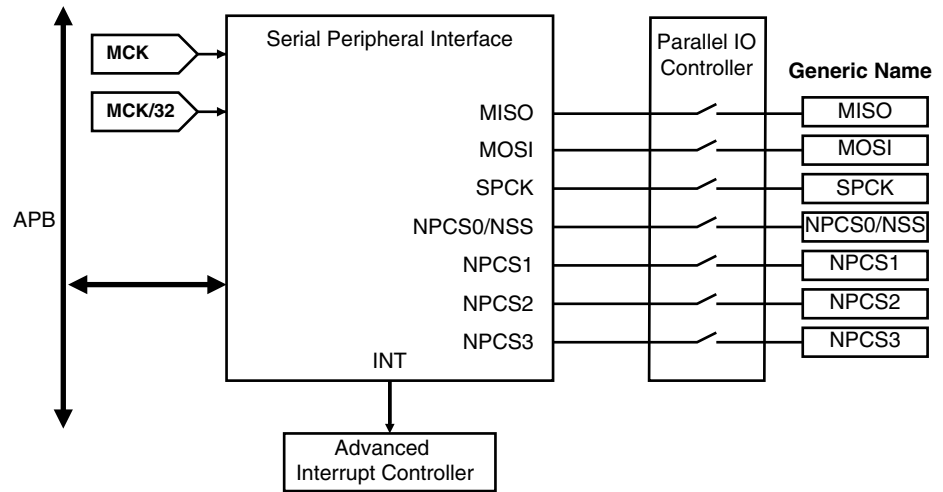


Table 20. SPI Pins

| Pin Name | Generic Mnemonic | Mode | Function |
|---|------------------|---------------------------|--|
| Master In Slave Out | MISO | Master Slave | Serial data input to SPI Serial data output from SPI |
| Master Out Slave In | MOSI | Master Slave | Serial data output from SPI Serial data input to SPI |
| Serial Clock | SPCK | Master Slave | Clock output from SPI Clock input to SPI |
| Peripheral Chip Selects | NPCS1- NPCS3 | Master | Select peripherals |
| Peripheral Chip Select/ Slave Select | NPCS0/ NSS | Master Master Slave | Output: Selects peripheral Input: low causes mode fault Input: chip select for SPI |

- Notes:
1. After a hardware reset, the SPI clock is disabled by default (see “PMC: Power Management Controller” on page 55). The user must configure the Power Management Controller before any access to the User Interface of the SPI.
 2. After a hardware reset, the SPI pins are deselected by default (see “PIO: Parallel I/O Controller” on page 98). The user must configure the PIO Controller to enable the corresponding pins for their SPI function. NPCS0/NSS must be configured as open-drain in the Parallel I/O Controller for multi-master operation.

Master Mode

In Master mode, the SPI controls data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select(s) to the slave(s) and the serial clock (SPCK). After enabling the SPI, a data transfer begins when the ARM core writes to the SP_TDR (Transmit Data Register). See Table 21.

Transmit and Receive buffers maintain the data flow at a constant rate with a reduced requirement for high priority interrupt servicing. When new data is available in the SP_TDR (Transmit Data Register) the SPI continues to transfer data. If the SP_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error (OVRES) flag is set.

The delay between the activation of the chip select and the start of the data transfer (DLYBS) as well as the delay between each data transfer (DLYBCT) can be programmed for each of the four external chip selects. All data transfer characteristics including the two timing values are programmed in registers SP_CSR0 to SP_CSR3 (Chip Select Registers). See Table 21.

In master mode the peripheral selection can be defined in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral
- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Figures 61 and 62 show the operation of the SPI in Master mode. For details concerning the flag and control bits in these diagrams, see the tables in the Programmer's Model, starting on page 175.

Fixed Peripheral Select

This mode is ideal for transferring memory blocks without the extra overhead in the transmit data register to determine the peripheral.

Fixed Peripheral Select is activated by setting bit PS to zero in SP_MR (Mode Register). The peripheral is defined by the PCS field, also in SP_MR.

This option is only available when the SPI is programmed in master mode.

Variable Peripheral Select

Variable Peripheral Select is activated by setting bit PS to one. The PCS field in SP_TDR (Transmit Data Register) is used to select the destination peripheral. The data transfer characteristics are changed when the selected peripheral changes, according to the associated chip select register.

The PCS field in the SP_MR has no effect.

This option is only available when the SPI is programmed in master mode.

Chip Selects

The Chip Select lines are driven by the SPI only if it is programmed in Master mode. These lines are used to select the destination peripheral. The PCSDEC field in SP_MR (Mode Register) selects 1 to 4 peripherals (PCSDEC = 0) or up to 15 peripherals (PCSDEC = 1).

If Variable Peripheral Select is active, the chip select signals are defined for each transfer in the PCS field in SP_TDR. Chip select signals can thus be defined independently for each transfer.

If Fixed Peripheral Select is active, Chip Select signals are defined for all transfers by the field PCS in SP_MR. If a transfer with a new peripheral is necessary, the software must wait until the current transfer is completed, then change the value of PCS in SP_MR before writing new data in SP_TDR.

The value on the NPCS pins at the end of each transfer can be read in the SP_RDR (Receive Data Register). By default, all NPCS signals are high (equal to one) before and after each transfer.

Mode Fault Detection

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCSA/NSS signal.

When a mode fault is detected, the MODF bit in the SP_SR is set until the SP_SR is read and the SPI is disabled until re-enabled by bit SPIEN in the SP_CR (Control Register).

Figure 61. Functional Flow Diagram in Master Mode

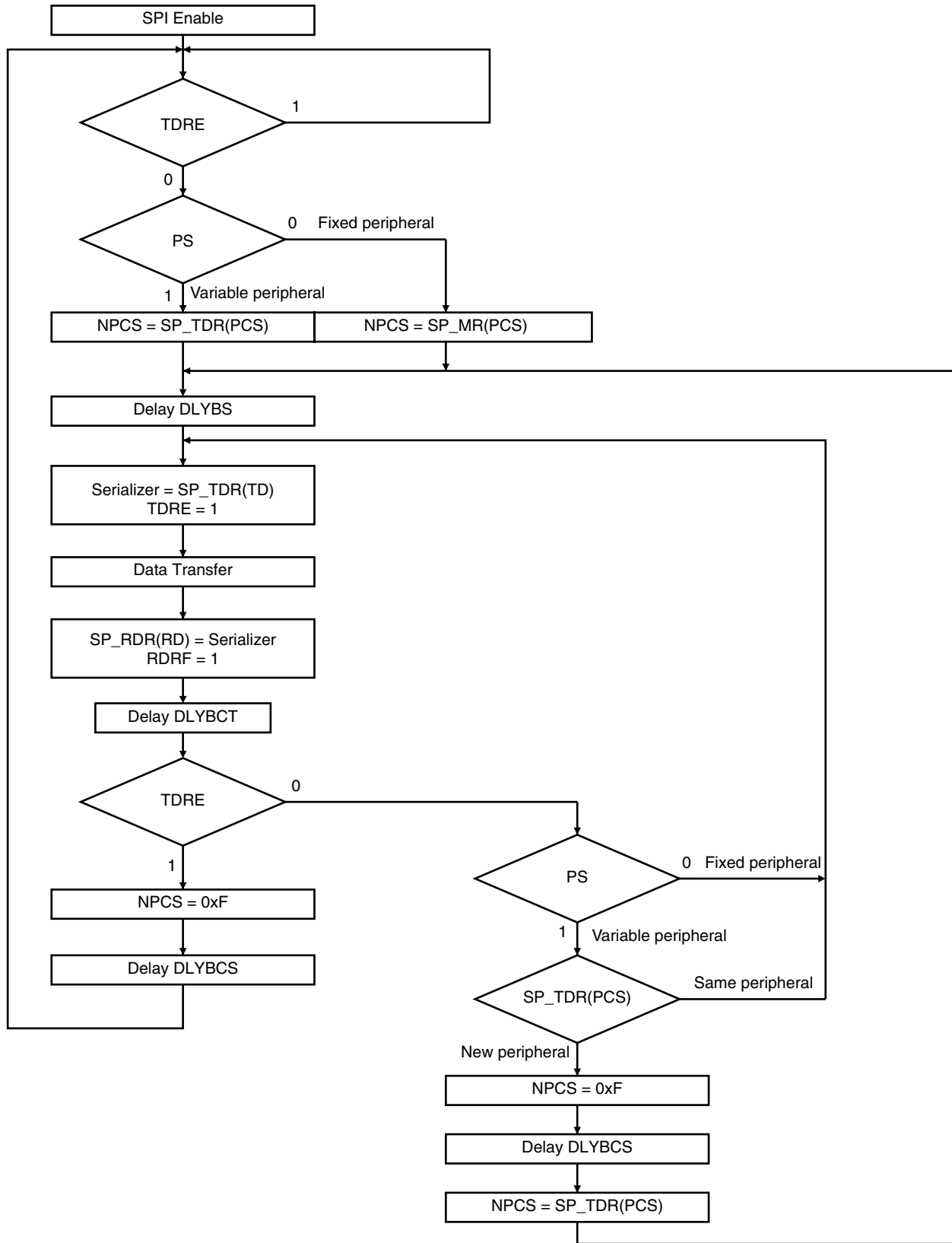
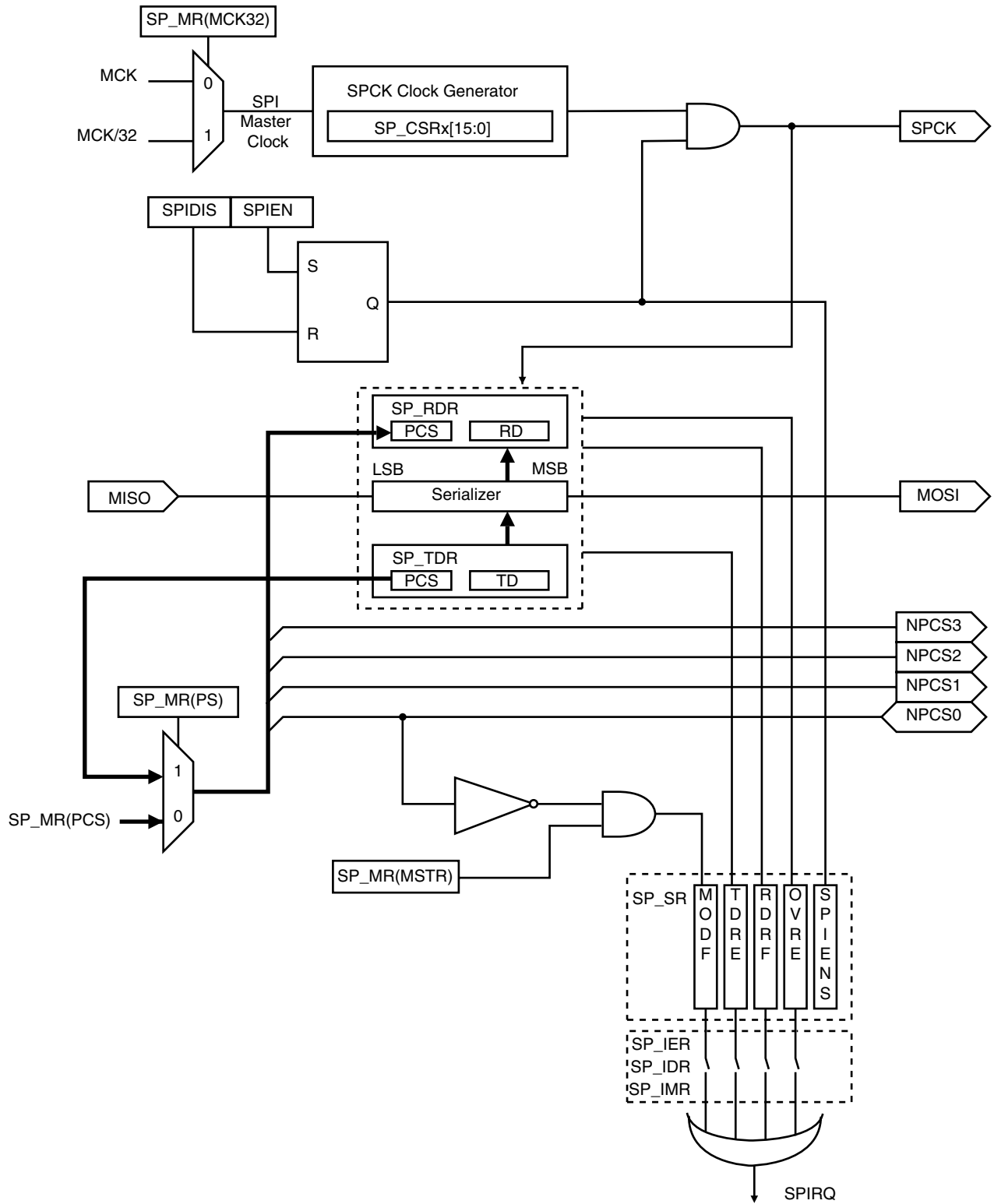


Figure 62. SPI in Master Mode

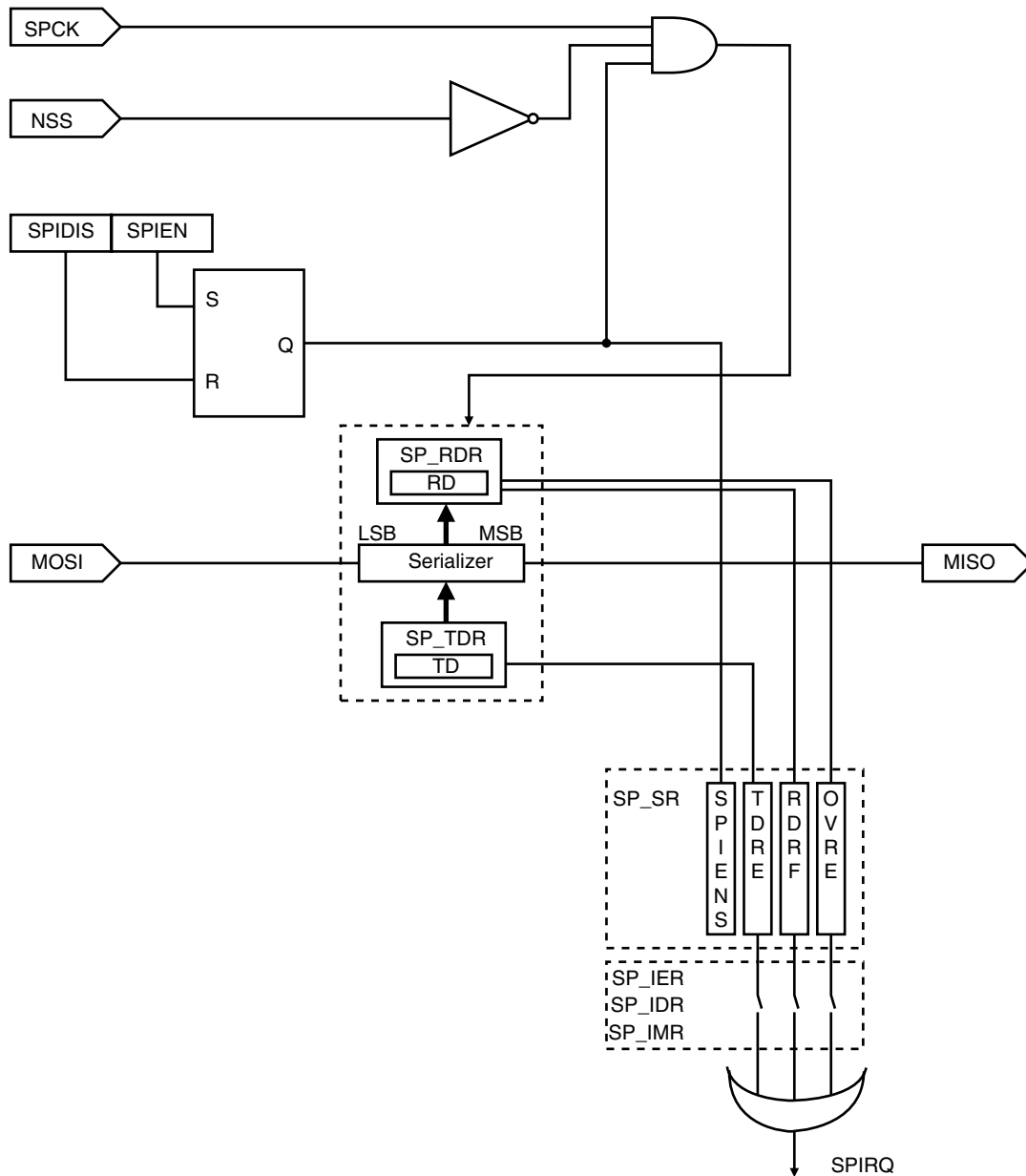


Slave Mode

In Slave Mode, the SPI waits for NSS to go active low before receiving the serial clock from an external master.

In slave mode CPOL, NCPHA and BITS fields of SP_CSR0 are used to define the transfer characteristics. The other Chip Select Registers are not used in slave mode.

Figure 63. SPI in Slave Mode



Data Transfer

The following waveforms show examples of data transfers.

Figure 64. SPI Transfer Format (NCPHA = 1, 8 Bits per Transfer)

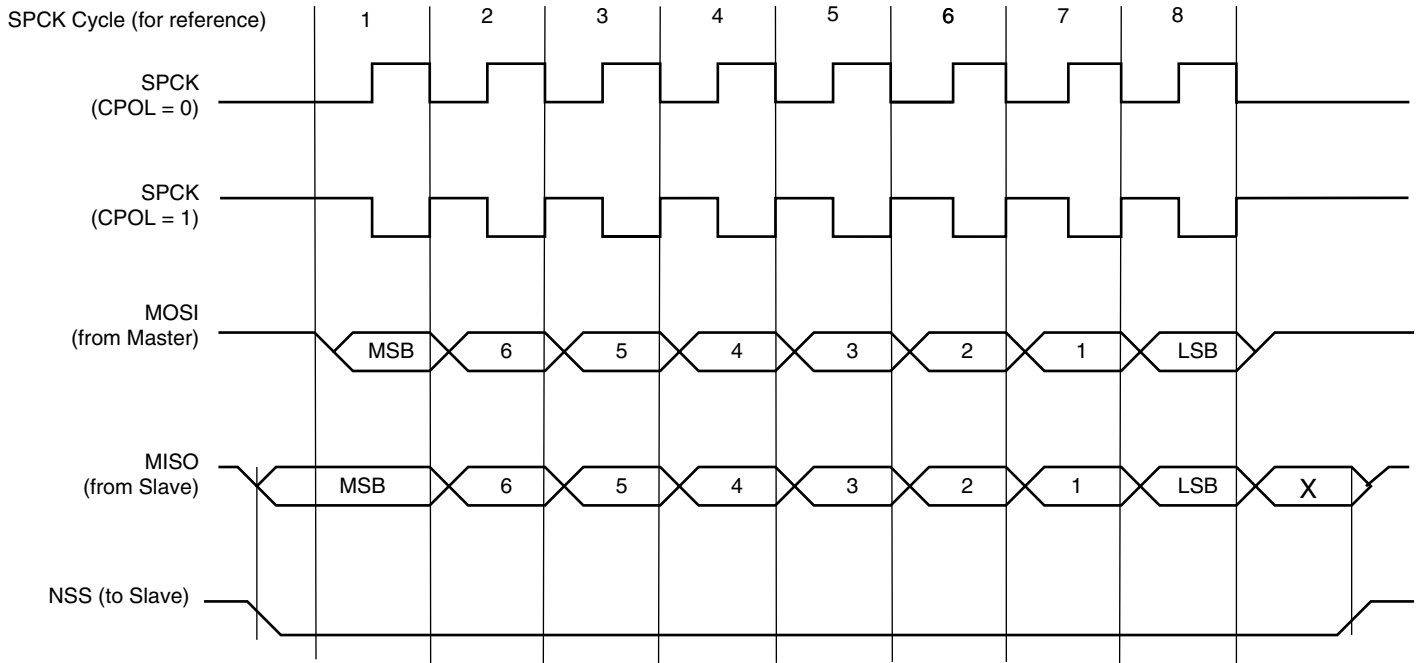


Figure 65. SPI Transfer Format (NCPHA = 0, 8 Bits per Transfer)

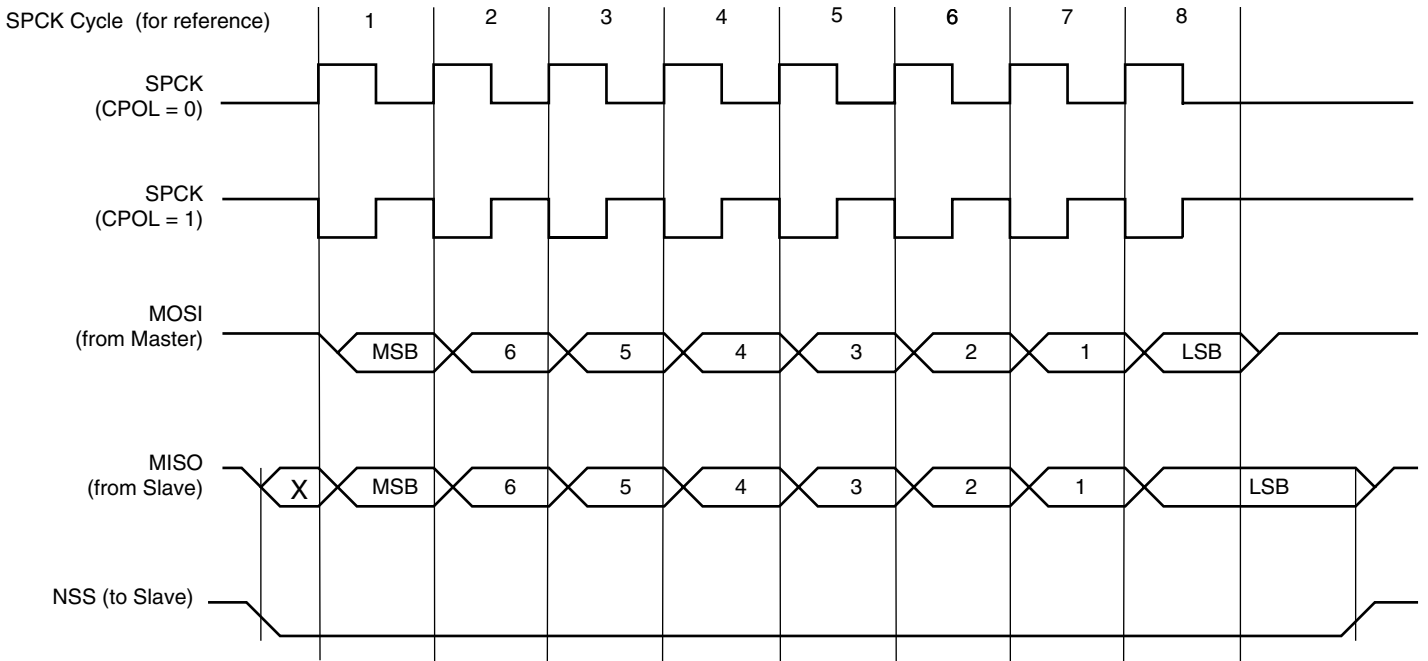
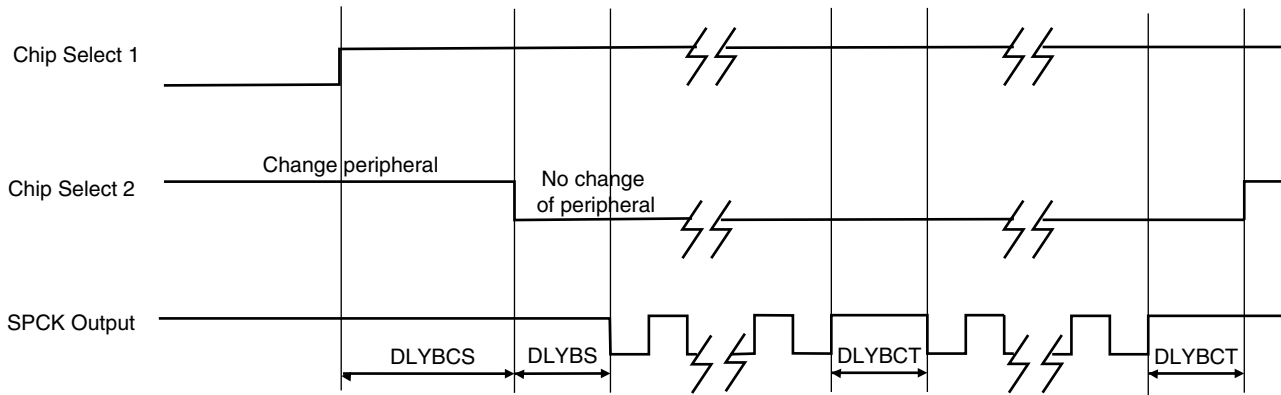


Figure 66. Programmable Delays (DLYBCS, DLYBS and DLYBCT)



Clock Generation

In Master Mode the SPI Master Clock is either MCK or MCK/32, as defined by the MCK32 field of SP_MR. The SPI baud rate clock is generated by dividing the SPI Master Clock by a value between 4 and 510. The divisor is defined in the SCBR field in each Chip Select Register. The transfer speed can thus be defined independently for each chip select signal.

CPOL and NCPHA in the Chip Select Registers define the clock/data relationship between master and slave devices. CPOL defines the inactive value of the SPCK. NCPHA defines which edge causes data to change and which edge causes data to be captured.

In Slave Mode, the input clock low and high pulse duration must strictly be longer than two system clock (MCK) periods.

Peripheral Data Controller

Each SPI is closely connected to two Peripheral Data Controller channels. One is dedicated to the receiver. The other is dedicated to the transmitter.

The PDC channel is programmed using SP_TPR (Transmit Pointer) and SP_TCR (Transmit Counter) for the transmitter and SP_RPR (Receive Pointer) and SP_RCR (Receive Counter) for the receiver. The status of the PDC is given in SP_SR by the SPENDTX bit for the transmitter and by the SPENDRX bit for the receiver.

The pointer registers (SP_TPR and SP_RPR) are used to store the address of the transmit or receive buffers. The counter registers (SP_TCR and SP_RCR) are used to store the size of these buffers.

The receiver data transfer is triggered by the RDRF bit and the transmitter data transfer is triggered by TDRE. When a transfer is performed, the counter is decremented and the pointer is incremented. When the counter reaches 0, the status bit is set (SPENDRX for the receiver, SPENDTX for the transmitter in SP_SR) and can be programmed to generate an interrupt. While the counter is at zero, the status bit is asserted and transfers are disabled.

SPI Programmer's Model

SPIA Base Address: 0xFFFC8000

SPIB Base Address: 0xFFCC000

Table 21. SPI Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|----------------------------|---------|------------|-------------|
| 0x00 | Control Register | SP_CR | Write-only | – |
| 0x04 | Mode Register | SP_MR | Read/Write | 0 |
| 0x08 | Receive Data Register | SP_RDR | Read-only | 0 |
| 0x0C | Transmit Data Register | SP_TDR | Write-only | – |
| 0x10 | Status Register | SP_SR | Read-only | 0 |
| 0x14 | Interrupt Enable Register | SP_IER | Write-only | – |
| 0x18 | Interrupt Disable Register | SP_IDR | Write-only | – |
| 0x1C | Interrupt Mask Register | SP_IMR | Read-only | 0 |
| 0x20 | Receive Pointer Register | SP_RPR | Read/Write | 0 |
| 0x24 | Receive Counter Register | SP_RCR | Read/Write | 0 |
| 0x28 | Transmit Pointer Register | SP_TPR | Read/Write | 0 |
| 0x2C | Transmit Counter Register | SP_TCR | Read/Write | 0 |
| 0x30 | Chip Select Register 0 | SP_CSR0 | Read/Write | 0 |
| 0x34 | Chip Select Register 1 | SP_CSR1 | Read/Write | 0 |
| 0x38 | Chip Select Register 2 | SP_CSR2 | Read/Write | 0 |
| 0x3C | Chip Select Register 3 | SP_CSR3 | Read/Write | 0 |

SPI Control Register

Register Name: SP_CR
Access Type: Write-only
Offset: 0x00

| | | | | | | | |
|-------|----|----|----|----|----|--------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWRST | – | – | – | – | – | SPIDIS | SPIEN |

- **SPIEN: SPI Enable (Code Label SP_SPIEN)**

0 = No effect.
 1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable (Code Label SP_SPIDIS)**

0 = No effect.
 1 = Disables the SPI.
 All pins are set in input mode and no data is received or transmitted.
 If a transfer is in progress, the transfer is finished before the SPI is disabled.
 If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software reset (Code Label SP_SWRST)**

0 = No effect.
 1 = Resets the SPI.
 A software triggered hardware reset of the SPI interface is performed.

SPI Mode Register

Register Name: SP_MR
Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

| | | | | | | | |
|--------|----|----|----|-------|--------|----|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DLYBCS | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | PCS | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLB | - | - | - | MCK32 | PCSDEC | PS | MSTR |

- **MSTR: Master/Slave Mode (Code Label SP_MSTR)**
 0 = SPI is in Slave mode.
 1 = SPI is in Master mode.
 MSTR configures the SPI Interface for either master or slave mode operation.
- **PS: Peripheral Select**

| PS | Selected PS | Code Label: SP_PS |
|----|----------------------------|-------------------|
| 0 | Fixed Peripheral Select | SP_PS_FIXED |
| 1 | Variable Peripheral Select | SP_PS_VARIABLE |

- **PCSDEC: Chip Select Decode (Code Label SP_PCSDEC)**
 0 = The chip selects are directly connected to a peripheral device.
 1 = The four chip select lines are connected to a 4- to 16-bit decoder.
 When PCSDEC equals one, up to 16 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder.
 The Chip Select Registers define the characteristics of the 16 chip selects according to the following rules:
 SP_CSR0 defines peripheral chip select signals 0 to 3.
 SP_CSR1 defines peripheral chip select signals 4 to 7.
 SP_CSR2 defines peripheral chip select signals 8 to 11.
 SP_CSR3 defines peripheral chip select signals 12 to 15 ⁽¹⁾.
- Note: 1. The 16th state corresponds to a state in which all chip selects are inactive. This allows a different clock configuration to be defined by each chip select register.

- **MCK32: Clock Selection (Code Label SP_DIV32)**
 0 = SPI Master Clock equals MCK
 1 = SPI Master Clock equals MCK/32
- **LLB: Local Loopback Enable (Code Label SP_LLB)**
 0 = Local loopback path disabled
 1 = Local loopback path enabled
 LLB controls the local loopback on the data serializer for testing in master mode only.

- **PCS: Peripheral Chip Select (Code Label SP_PCS)**
 This field is only used if Fixed Peripheral Select is active (PS = 0).
 If PCSDEC=0:
 PCS = xxx0 NPCS[3:0] = 1110
 PCS = xx01 NPCS[3:0] = 1101
 PCS = x011 NPCS[3:0] = 1011
 PCS = 0111 NPCS[3:0] = 0111
 PCS = 1111 forbidden (no peripheral is selected)
 (x = don't care)
 If PCSDEC=1: NPCS[3:0] output signals = PCS

- **DLYBCS: Delay Between Chip Selects (Code Label SP_DLYBCS)**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times. If DLYBCS is less than or equal to six, six SPI Master Clock periods will be inserted by default. Otherwise, the following equation determines the delay:

$$\text{Delay_Between_Chip_Selects} = \text{DLYBCS} \cdot \text{SPI_Master_Clock_period}$$

SPI Receive Data Register

Register Name: SP_RDR
Access Type: Read-only
Offset: 0x08
Reset Value: 0x0

| | | | | | | | |
|----|----|----|----|-----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | PCS | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RD | | | | | | | |

- **RD: Receive Data (Code Label SP_RD)**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select Status**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

SPI Transmit Data Register

Register Name: SP_TDR
Access Type: Write-only
Offset: 0x0C

| | | | | | | | |
|----|----|----|----|-----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | PCS | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TD | | | | | | | |

• **TD: Transmit Data (Code Label *SP_TD*)**

Data which is to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

• **PCS: Peripheral Chip Select**

This field is only used if Variable Peripheral Select is active (PS = 1) and if the SPI is in Master Mode.

If PCSDEC = 0:

- PCS = xxx0 NPCS[3:0] = 1110
 - PCS = xx01 NPCS[3:0] = 1101
 - PCS = x011 NPCS[3:0] = 1011
 - PCS = 0111 NPCS[3:0] = 0111
 - PCS = 1111 forbidden (no peripheral is selected)
- (x = don't care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS



SPI Status Register

Register Name: SP_SR
Access Type: Read-only
Offset: 0x10
Reset Value: 0x0

| | | | | | | | |
|----|----|---------|---------|-------|------|------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | SPIENS |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | SPENDTX | SPENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full (Code Label SP_RDRF)**
 0 = No data has been received since the last read of SP_RDR
 1 = Data has been received and the received data has been transferred from the serializer to SP_RDR since the last read of SP_RDR.
- **TDRE: Transmit Data Register Empty (Code Label SP_TDRE)**
 0 = Data has been written to SP_TDR and not yet transferred to the serializer.
 1 = The last data written in the Transmit Data Register has been transferred in the serializer.
 TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.
- **MODF: Mode Fault Error (Code Label SP_MODF)**
 0 = No Mode Fault has been detected since the last read of SP_SR.
 1 = A Mode Fault occurred since the last read of the SP_SR.
- **OVRES: Overrun Error Status (Code Label SP_OVRES)**
 0 = No overrun has been detected since the last read of SP_SR.
 1 = An overrun has occurred since the last read of SP_SR.
 An overrun occurs when SP_RDR is loaded at least twice from the serializer since the last read of the SP_RDR.
- **SPENDRX: End of Receiver Transfer (Code Label SP_SPENDRX)**
 0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is inactive.
 1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is active.
- **SPENDTX: End of Transmitter Transfer (Code Label SP_SPENDTX)**
 0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is inactive.
 1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is active.
- **SPIENS: SPI Enable Status (Code Label SP_SPIENS)**
 0 = SPI is disabled.
 1 = SPI is enabled.

SPI Interrupt Enable Register

Register Name: SP_IER
Access Type: Write-only
Offset: 0x14

| | | | | | | | |
|----|----|---------|---------|-------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | SPENDTX | SPENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Enable (Code Label SP_RDRF)**
 0 = No effect.
 1 = Enables the Receiver Data Register Full Interrupt.
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable (Code Label SP_TDRE)**
 0 = No effect.
 1 = Enables the Transmit Data Register Empty Interrupt.
- **MODF: Mode Fault Error Interrupt Enable (Code Label SP_MODF)**
 0 = No effect.
 1 = Enables the Mode Fault Interrupt.
- **OVRES: Overrun Error Interrupt Enable (Code Label SP_OVRES)**
 0 = No effect.
 1 = Enables the Overrun Error Interrupt.
- **SPENDRX: End of Receiver Transfer Interrupt Enable (Code Label SP_SPENDRX)**
 0 = No effect.
 1 = Enables the End of Receiver Transfer Interrupt.
- **SPENDTX: End of Transmitter Transfer Interrupt Enable (Code Label SP_SPENDTX)**
 0 = No effect.
 1 = Enables the End of Transmitter Transfer Interrupt.

SPI Interrupt Disable Register

Register Name: SP_IDR
Access Type: Write-only
Offset: 0x18

| | | | | | | | |
|----|----|---------|---------|-------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | SPENDTX | SPENDRX | OVRES | MODF | TDRE | RDRF |

- RDRF: Receive Data Register Full Interrupt Disable (Code Label SP_RDRF)**
 0 = No effect.
 1 = Disables the Receiver Data Register Full Interrupt.
- TDRE: Transmit Data Register Empty Interrupt Disable (Code Label SP_TDRE)**
 0 = No effect.
 1 = Disables the Transmit Data Register Empty Interrupt.
- MODF: Mode Fault Interrupt Disable (Code Label SP_MODF)**
 0 = No effect.
 1 = Disables the Mode Fault Interrupt.
- OVRES: Overrun Error Interrupt Disable (Code Label SP_OVRES)**
 0 = No effect.
 1 = Disables the Overrun Error Interrupt.
- SPENDRX: End of Receiver Transfer Interrupt Disable (Code Label SP_SPENDRX)**
 0 = No effect.
 1 = Disables the End of Receiver Transfer Interrupt.
- SPENDTX: End of Transmitter Transfer Interrupt Disable (Code Label SP_SPENDTX)**
 0 = No effect.
 1 = Disables the End of Transmitter Transfer Interrupt.

SPI Interrupt Mask Register

Register Name: SP_IMR
Access Type: Read-only
Offset: 0x1C
Reset Value: 0x0

| | | | | | | | |
|----|----|---------|---------|-------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | SPENDTX | SPENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Mask (Code Label SP_RDRF)**
 0 = Receive Data Register Full Interrupt is disabled.
 1 = Receive Data Register Full Interrupt is enabled.
- **TDRE: Transmit Data Register Empty Interrupt Mask (Code Label SP_TDRE)**
 0 = Transmit Data Register Empty Interrupt is disabled.
 1 = Transmit Data Register Empty Interrupt is enabled.
- **MODF: Mode Fault Interrupt Mask (Code Label SP_MODF)**
 0 = Mode Fault Interrupt is disabled.
 1 = Mode Fault Interrupt is enabled.
- **OVRES: Overrun Error Interrupt Mask (Code Label SP_OVRES)**
 0 = Overrun Error Interrupt is disabled.
 1 = Overrun Error Interrupt is enabled.
- **SPENDRX: End of Receiver Transfer Interrupt Mask (Code Label SP_SPENDRX)**
 0 = End of Receiver Transfer Interrupt is disabled.
 1 = End of Receiver Transfer Interrupt is enabled.
- **SPENDTX: End of Transmitter Transfer Interrupt Mask (Code Label SP_SPENDTX)**
 0 = End of Transmitter Transfer Interrupt is disabled.
 1 = End of Transmitter Transfer Interrupt is enabled.

SPI Receive Pointer Register

Name: SP_RPR
Access Type: Read/Write
Offset: 0x20
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| RXPTR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXPTR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXPTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXPTR | | | | | | | |

- **RXPTR: Receive Pointer**

RXPTR must be loaded with the address of the receive buffer.

SPI Receive Counter Register

Name: SP_RCR
Access Type: Read/Write
Offset: 0x24
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXCTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXCTR | | | | | | | |

- **RXCTR: Receive Counter**

RXCTR must be loaded with the size of the receive buffer.
 0: Stop Peripheral Data Transfer dedicated to the receiver.
 1-65535: Start Peripheral Data transfer if RDRF is active.

SPI Transmit Pointer Register

Name: SP_TPR
Access Type: Read/Write
Offset: 0x28
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| TXPTR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXPTR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXPTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXPTR | | | | | | | |

- **TXPTR: Transmit Pointer**
 TXPTR must be loaded with the address of the transmit buffer.

SPI Transmit Counter Register

Name: SP_TCR
Access Type: Read/Write
Offset: 0x2C
Reset Value: 0x0

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXCTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXCTR | | | | | | | |

- **TXCTR: Transmit Counter**
 TXCTR must be loaded with the size of the transmit buffer.
 0: Stop Peripheral Data Transfer dedicated to the transmitter.
 1-65535: Start Peripheral Data transfer if TDRE is active.



SPI Chip Select Register

Register Name: SP_CSR0..SP_CSR3
Access Type: Read/Write
Reset Value: 0x0

| | | | | | | | |
|--------|----|----|----|----|----|-------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DLYBCT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DLYBS | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SCBR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BITS | | | | - | - | NCPHA | CPOL |

- **CPOL: Clock Polarity (Code Label SP_CPOL)**

0 = The inactive state value of SPCK is logic level zero.
 1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce a desired clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase (Code Label SP_NCPHA)**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.
 1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used.

| BITS[3:0] | Bits Per Transfer | Code Label: SP_BITS |
|-----------|-------------------|---------------------|
| 0000 | 8 | SP_BITS_8 |
| 0001 | 9 | SP_BITS_9 |
| 0010 | 10 | SP_BITS_10 |
| 0011 | 11 | SP_BITS_11 |
| 0100 | 12 | SP_BITS_12 |
| 0101 | 13 | SP_BITS_13 |
| 0110 | 14 | SP_BITS_14 |
| 0111 | 15 | SP_BITS_15 |
| 1000 | 16 | SP_BITS_16 |
| 1001 | Reserved | - |
| 1010 | Reserved | - |
| 1011 | Reserved | - |
| 1100 | Reserved | - |
| 1101 | Reserved | - |
| 1110 | Reserved | - |
| 1111 | Reserved | - |

- **SCBR: Serial Clock Baud Rate (Code Label SP_SCBR)**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the SPI Master Clock (selected between MCK and MCK/32). The baud rate is selected by writing a value from 2 to 255 in the field SCBR. The following equation determines the SPCK baud rate:

$$\text{SPCK_Baud_Rate} = \frac{\text{SPI_Master_Clock_frequency}}{2 \times \text{SCBR}}$$

Giving SCBR a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers may occur. At reset, baud rate is disabled.

- **DLYBS: Delay Before SPCK (Code Label SP_DLYBS)**

This field defines the delay from NPCS valid to the first valid SPCK transition. When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period. Otherwise, the following equation determines the delay:

$$\text{NPCS_to_SPCK_Delay} = \text{DLYBS} \cdot \text{SPI_Master_Clock_period}$$

- **DLYBCT: Delay Between Consecutive Transfers (Code Label SP_DLYBCT)**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed. When DLYBCT equals zero, a delay of four SPI Master Clock periods are inserted. Otherwise, the following equation determines the delay:

$$\text{Delay_After_Transfer} = 32 \cdot \text{DLYBCT} \cdot \text{SPI_Master_Clock_period}$$

JTAG Boundary-scan Register

The Boundary-scan Register (BSR) contains 237 bits which correspond to active pins and associated control signals.

Each AT91M42800A input pin has a corresponding bit in the Boundary-scan Register for observability.

Each AT91M42800A output pin has a corresponding 2-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The CTRL bit can put the pad into high impedance.

Each AT91M42800A in/out pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit is for the observability of data applied to the pad. The CTRL bit selects the direction of the pad.

Table 22. Boundary-scan Register

| Bit Number | Pin Name | Pin Type | Associated BSR Cells | Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|----------------------|----------|----------------------|------------|----------------------|----------|----------------------|
| 237 | PA25/MCKO | IN/OUT | OUTPUT | 210 | PA16/NPCSA2 | IN/OUT | OUTPUT |
| 236 | | | INPUT | 209 | | | INPUT |
| 235 | | | CTRL | 208 | | | CTRL |
| 234 | PA24/NPCSB3 | IN/OUT | OUTPUT | 207 | PA15/NPCSA1 | IN/OUT | OUTPUT |
| 233 | | | INPUT | 206 | | | INPUT |
| 232 | | | CTRL | 205 | | | CTRL |
| 231 | PA23/NPCSB2 | IN/OUT | OUTPUT | 204 | PA14/NPCSA0/ NSSA | IN/OUT | OUTPUT |
| 230 | | | INPUT | 203 | | | INPUT |
| 229 | | | CTRL | 202 | | | CTRL |
| 228 | PA22/NPCSB1 | IN/OUT | OUTPUT | 201 | PA13/MOSIA | IN/OUT | OUTPUT |
| 227 | | | INPUT | 200 | | | INPUT |
| 226 | | | CTRL | 199 | | | CTRL |
| 225 | PA21/NPCSB0/ NSSB | IN/OUT | OUTPUT | 198 | PA12/MISOA | IN/OUT | OUTPUT |
| 224 | | | INPUT | 197 | | | INPUT |
| 223 | | | CTRL | 196 | | | CTRL |
| 222 | PA20/MOSIB | IN/OUT | OUTPUT | 195 | PA11/SPCKA | IN/OUT | OUTPUT |
| 221 | | | INPUT | 194 | | | INPUT |
| 220 | | | CTRL | 193 | | | CTRL |
| 219 | PA19/MISOB | IN/OUT | OUTPUT | 192 | PA10/RXD1 | IN/OUT | OUTPUT |
| 218 | | | INPUT | 191 | | | INPUT |
| 217 | | | CTRL | 190 | | | CTRL |
| 216 | PA18/SPCKB | IN/OUT | OUTPUT | 189 | PA9/TXD1/NTRI | IN/OUT | OUTPUT |
| 215 | | | INPUT | 188 | | | INPUT |
| 214 | | | CTRL | 187 | | | CTRL |

Table 22. Boundary-scan Register (Continued)

| Bit Number | Pin Name | Pin Type | Associated BSR Cells | Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|-------------|----------|----------------------|------------|------------|----------|----------------------|
| 213 | PA17/NPCSA3 | IN/OUT | OUTPUT | 186 | PA8/SCK1 | IN/OUT | OUTPUT |
| 212 | | | INPUT | 185 | | | INPUT |
| 211 | | | CTRL | 184 | | | CTRL |
| 183 | PA7/RXD0 | IN/OUT | OUTPUT | 150 | PB20/TIOB4 | IN/OUT | OUTPUT |
| 182 | | | INPUT | 149 | | | INPUT |
| 181 | | | CTRL | 148 | | | CTRL |
| 180 | PA6/TXD0 | IN/OUT | OUTPUT | 147 | PB19/TIOA4 | IN/OUT | OUTPUT |
| 179 | | | INPUT | 146 | | | INPUT |
| 178 | | | CTRL | 145 | | | CTRL |
| 177 | PA5/SCK0 | IN/OUT | OUTPUT | 144 | PB18/TCLK4 | IN/OUT | OUTPUT |
| 176 | | | INPUT | 143 | | | INPUT |
| 175 | | | CTRL | 142 | | | CTRL |
| 174 | PA4/FIQ | IN/OUT | OUTPUT | 141 | PB17/TIOB3 | IN/OUT | OUTPUT |
| 173 | | | INPUT | 140 | | | INPUT |
| 172 | | | CTRL | 139 | | | CTRL |
| 171 | PA3/IRQ3 | IN/OUT | OUTPUT | 138 | PB16/TIOA3 | IN/OUT | OUTPUT |
| 170 | | | INPUT | 137 | | | INPUT |
| 169 | | | CTRL | 136 | | | CTRL |
| 168 | PA2/IRQ2 | IN/OUT | OUTPUT | 135 | PB15/TCLK3 | IN/OUT | OUTPUT |
| 167 | | | INPUT | 134 | | | INPUT |
| 166 | | | CTRL | 133 | | | CTRL |
| 165 | PA1/IRQ1 | IN/OUT | OUTPUT | 132 | PB14/TIOB2 | IN/OUT | OUTPUT |
| 164 | | | INPUT | 131 | | | INPUT |
| 163 | | | CTRL | 130 | | | CTRL |
| 162 | PA0/IRQ0 | IN/OUT | OUTPUT | 129 | PB13/TIOA2 | IN/OUT | OUTPUT |
| 161 | | | INPUT | 128 | | | INPUT |
| 160 | | | CTRL | 127 | | | CTRL |
| 159 | PB23/TIOB5 | IN/OUT | OUTPUT | 126 | PB12/TCLK2 | IN/OUT | OUTPUT |
| 158 | | | INPUT | 125 | | | INPUT |
| 157 | | | CTRL | 124 | | | CTRL |
| 156 | PB22/TIOA5 | IN/OUT | OUTPUT | 123 | PB11/TIOB1 | IN/OUT | OUTPUT |
| 155 | | | INPUT | 122 | | | INPUT |
| 154 | | | CTRL | 121 | | | CTRL |

Table 22. Boundary-scan Register (Continued)

| Bit Number | Pin Name | Pin Type | Associated BSR Cells | Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|------------|----------|----------------------|------------|-------------|----------|----------------------|
| 153 | PB21/TCLK5 | IN/OUT | OUTPUT | 120 | PB10/TIOA1 | IN/OUT | OUTPUT |
| 152 | | | INPUT | 119 | | | INPUT |
| 151 | | | CTRL | 118 | | | CTRL |
| 117 | PB9/TCLK1 | IN/OUT | OUTPUT | 82 | D5 | IN/OUT | INPUT |
| 116 | | | INPUT | 81 | D4 | IN/OUT | OUTPUT |
| 115 | | | CTRL | 80 | | INPUT | |
| 114 | PB8/TIOB0 | IN/OUT | OUTPUT | 79 | D[7:4] | IN/OUT | CTRL |
| 113 | | | INPUT | 78 | D3 | IN/OUT | OUTPUT |
| 112 | | | CTRL | 77 | | INPUT | |
| 111 | PB7/TIOA0 | IN/OUT | OUTPUT | 76 | D2 | IN/OUT | OUTPUT |
| 110 | | | INPUT | 75 | | | INPUT |
| 109 | | | CTRL | 74 | D1 | IN/OUT | OUTPUT |
| 108 | IN/OUT | OUTPUT | 73 | INPUT | | | |
| 107 | PB6/TCLK0 | | INPUT | 72 | D0 | IN/OUT | OUTPUT |
| 106 | | | CTRL | 71 | | | INPUT |
| 105 | D15 | IN/OUT | OUTPUT | 70 | D[3:0] | IN/OUT | CTRL |
| 104 | | | INPUT | 69 | PB5/A23/CS4 | IN/OUT | OUTPUT |
| 103 | OUTPUT | 68 | INPUT | | | | |
| 102 | INPUT | 67 | CTRL | | | | |
| 101 | D13 | IN/OUT | OUTPUT | 66 | PB4/A22/CS5 | IN/OUT | OUTPUT |
| 100 | | | INPUT | 65 | | | INPUT |
| 99 | D12 | IN/OUT | OUTPUT | 64 | | | CTRL |
| 98 | | | INPUT | 63 | PB3/A21/CS6 | IN/OUT | OUTPUT |
| 97 | D[15:12] | IN/OUT | 62 | INPUT | | | |
| 96 | D11 | IN/OUT | OUTPUT | 61 | | | CTRL |
| 95 | | | INPUT | 60 | PB2/A20/CS7 | IN/OUT | OUTPUT |
| 94 | D10 | IN/OUT | OUTPUT | 59 | | | INPUT |
| 93 | | | INPUT | 58 | | | CTRL |
| 92 | D9 | IN/OUT | OUTPUT | 57 | A19 | OUTPUT | OUTPUT |
| 91 | | | INPUT | 56 | A18 | OUTPUT | OUTPUT |
| 90 | D8 | IN/OUT | OUTPUT | 55 | A17 | OUTPUT | OUTPUT |
| 89 | | | INPUT | 54 | A16 | OUTPUT | OUTPUT |
| 88 | D[11:8] | IN/OUT | CTRL | 53 | A[19:16] | OUTPUT | CTRL |
| 87 | D7 | IN/OUT | OUTPUT | 52 | A15 | OUTPUT | OUTPUT |
| 86 | | | INPUT | 51 | A14 | OUTPUT | OUTPUT |

Table 22. Boundary-scan Register (Continued)

| Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|----------|----------|----------------------|
| 85 | D6 | IN/OUT | OUTPUT |
| 84 | | | INPUT |
| 83 | D5 | IN/OUT | OUTPUT |
| 47 | A11 | OUTPUT | OUTPUT |
| 46 | A10 | OUTPUT | OUTPUT |
| 45 | A9 | OUTPUT | OUTPUT |
| 44 | A8 | OUTPUT | OUTPUT |
| 43 | A[11:8] | OUTPUT | CTRL |
| 42 | A7 | OUTPUT | OUTPUT |
| 41 | A6 | OUTPUT | OUTPUT |
| 40 | A5 | OUTPUT | OUTPUT |
| 39 | A4 | OUTPUT | OUTPUT |
| 38 | A[7:4] | OUTPUT | CTRL |
| 37 | A3 | OUTPUT | OUTPUT |
| 36 | A2 | OUTPUT | OUTPUT |
| 35 | A1 | OUTPUT | OUTPUT |
| 34 | NLB/A0 | OUTPUT | OUTPUT |
| 33 | A[3:0] | OUTPUT | CTRL |
| 32 | PB1/NCS3 | IN/OUT | OUTPUT |
| 31 | | | INPUT |
| 30 | | | CTRL |
| 29 | PB0/NCS2 | IN/OUT | OUTPUT |
| 28 | | | INPUT |
| 27 | | | CTRL |
| 26 | NCS1 | OUTPUT | OUTPUT |
| 25 | NCS0 | IN/OUT | OUTPUT |
| 24 | | | CTRL |
| 23 | NUB/NWR1 | IN/OUT | OUTPUT |
| 22 | | | INPUT |

| Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|---|----------|----------------------|
| 50 | A13 | OUTPUT | OUTPUT |
| 49 | A12 | OUTPUT | OUTPUT |
| 48 | A[15:12] | OUTPUT | CTRL |
| 21 | NWE/NWR0 | IN/OUT | OUTPUT |
| 20 | | | INPUT |
| 19 | NOE/NRD | IN/OUT | OUTPUT |
| 18 | | | INPUT |
| 17 | NOE/NRD NEW/NWR0 NUB/NWR1 NCS1 | IN/OUT | CTRL |
| 16 | NWAIT | INPUT | INPUT |
| 15 | PA29/PME | IN/OUT | OUTPUT |
| 14 | | | INPUT |
| 13 | | | CTRL |
| 12 | PA28 | IN/OUT | OUTPUT |
| 11 | | | INPUT |
| 10 | | | CTRL |
| 9 | NRST | INPUT | INPUT |
| 8 | PA27/BMS | IN/OUT | OUTPUT |
| 7 | | | INPUT |
| 6 | | | CTRL |
| 5 | NWDOVF | OUTPUT | OUTPUT |
| 5 | NWDOVF | OUTPUT | OUTPUT |
| 4 | | | CTRL |
| 3 | PA26 | IN/OUT | OUTPUT |
| 2 | | | INPUT |
| 1 | | | CTRL |

Document Details

Title AT91M42800A Datasheet

Literature Number 1779

Revision History

Version A **Publication Date:** Oct-01

Version B **Publication Date:** 22-Mar-02

Revisions Since Previous Version

Page: 4 Change in Table 2

Page: 5 Change in Table 3

Page: 10 Change in Table 4

Page: 11 Change in section Power Supply
Change in section Clock Generator

Page: 13 Added section Protection Mode
Change in section Internal Memories
Deleted section Protect Mode

Table of Contents

- Features..... 1***
- Description..... 1***
- Pin Configuration..... 2***
- Pin Description 5***
- Block Diagram..... 7***
- Architectural Overview..... 8***
 - Memories 8
 - Peripherals..... 8
- Associated Documentation 10***
- Product Overview 11***
 - Power Supply..... 11
 - Input/Output Considerations..... 11
 - Operating Modes 11
 - Clock Generator..... 11
 - Reset 12
 - Emulation Functions 12
 - Memory Controller 13
 - External Bus Interface 14
- Peripherals 15***
 - System Peripherals..... 16
 - User Peripherals 17
- Memory Map..... 18***
- Peripheral Memory Map 20***
- EBI: External Bus Interface..... 21***
 - External Memory Mapping..... 21
 - Abort Status 22
 - EBI Behavior During Internal Accesses..... 22
 - Pin Description..... 23
 - Chip Select Lines..... 24
 - Data Bus Width..... 25
 - Byte Write or Byte Select Access 26
 - Boot on NCS0..... 28
 - Read Protocols 29
 - Write Data Hold Time 31
 - Wait States 32

| | |
|--|-----------|
| Memory Access Waveforms | 36 |
| EBI User Interface | 48 |
| EBI Chip Select Register | 49 |
| EBI Remap Control Register | 51 |
| EBI Memory Control Register | 51 |
| Abort Status Register..... | 53 |
| Abort Address Status Register | 54 |
| | |
| PMC: Power Management Controller..... | 55 |
| Oscillator and Slow Clock | 56 |
| Master Clock..... | 56 |
| Master Clock Output Controller | 59 |
| ARM Processor Clock Controller | 59 |
| Peripheral Clock Controller..... | 60 |
| PMC User Interface | 61 |
| PMC System Clock Enable Register | 62 |
| PMC System Clock Disable Register | 62 |
| PMC System Clock Status Register | 63 |
| PMC Peripheral Clock Enable Register | 63 |
| PMC Peripheral Clock Disable Register | 64 |
| PMC Peripheral Clock Status Register..... | 64 |
| PMC Clock Generator Mode Register | 65 |
| PMC Status Register | 66 |
| PMC Interrupt Enable Register..... | 67 |
| PMC Interrupt Disable Register | 67 |
| PMC Interrupt Mask Register | 68 |
| | |
| ST: System Timer | 69 |
| PIT: Period Interval Timer..... | 69 |
| WDT: Watchdog Timer | 69 |
| RTT: Real-time Timer | 70 |
| System Timer User Interface | 72 |
| System Timer Control Register..... | 73 |
| System Timer Period Interval Mode Register | 74 |
| System Timer Watchdog Mode Register | 74 |
| System Timer Real-time Mode Register..... | 75 |
| System Timer Status Register | 75 |
| System Timer Interrupt Enable Register..... | 76 |
| System Timer Interrupt Disable Register | 77 |
| System Timer Interrupt Mask Register | 78 |
| System Timer Real-time Alarm Register | 78 |
| System Timer Current Real-time Register..... | 79 |
| | |
| AIC: Advanced Interrupt Controller | 80 |
| Hardware Interrupt Vectoring..... | 82 |
| Priority Controller | 82 |

Interrupt Handling 82
 Interrupt Masking 82
 Interrupt Clearing and Setting 83
 Fast Interrupt Request 83
 Software Interrupt 83
 Spurious Interrupt 83
 Protect Mode 84
 AIC User Interface 85
 AIC Source Mode Register 86
 AIC Source Vector Register 87
 AIC Interrupt Vector Register 88
 AIC FIQ Vector Register 88
 AIC Interrupt Status Register 89
 AIC Interrupt Pending Register 90
 AIC Interrupt Mask Register 90
 AIC Core Interrupt Status Register 91
 AIC Interrupt Enable Command Register 91
 AIC Interrupt Disable Command Register 92
 AIC Interrupt Clear Command Register 93
 AIC Interrupt Set Command Register 93
 AIC End of Interrupt Command Register 94
 AIC Spurious Vector Register 94
 Standard Interrupt Sequence 95
 Fast Interrupt Sequence 97

PIO: Parallel I/O Controller..... 98

PIO Connection Tables 101
 PIO Enable Register 104
 PIO Disable Register 104
 PIO Status Register 105
 PIO Output Enable Register 106
 PIO Output Disable Register 106
 PIO Output Status Register 107
 PIO Input Filter Enable Register 108
 PIO Input Filter Disable Register 108
 PIO Input Filter Status Register 109
 PIO Set Output Data Register 110
 PIO Clear Output Data Register 110
 PIO Output Data Status Register 111
 PIO Pin Data Status Register 111
 PIO Interrupt Enable Register 112
 PIO Interrupt Disable Register 112
 PIO Interrupt Mask Register 113
 PIO Interrupt Status Register 113
 PIO Multi-drive Enable Register 114
 PIO Multi-drive Disable Register 114





| | |
|---|------------|
| PIO Multi-drive Status Register | 115 |
| SF: Special Function Registers..... | 116 |
| Chip Identification | 116 |
| Chip ID Register | 117 |
| Chip ID Extension Register..... | 118 |
| Reset Status Register..... | 119 |
| SF Protect Mode Register | 119 |
| | |
| USART: Universal Synchronous/ Asynchronous Receiver/Transmitter..... | 120 |
| Pin Description..... | 121 |
| Baud Rate Generator..... | 122 |
| Receiver..... | 123 |
| Transmitter..... | 125 |
| Multi-drop Mode..... | 125 |
| Break | 125 |
| Peripheral Data Controller | 127 |
| Interrupt Generation..... | 127 |
| Channel Modes..... | 127 |
| USART User Interface | 129 |
| USART Control Register..... | 130 |
| USART Mode Register | 131 |
| USART Interrupt Enable Register..... | 133 |
| USART Interrupt Disable Register..... | 134 |
| USART Interrupt Mask Register | 135 |
| USART Channel Status Register..... | 136 |
| USART Receiver Holding Register..... | 137 |
| USART Transmitter Holding Register..... | 138 |
| USART Baud Rate Generator Register | 138 |
| USART Receiver Time-out Register..... | 139 |
| USART Transmitter Time-guard Register..... | 139 |
| USART Receive Pointer Register..... | 140 |
| USART Receive Counter Register | 140 |
| USART Transmit Pointer Register..... | 141 |
| USART Transmit Counter Register | 141 |
| | |
| TC: Timer/Counter | 142 |
| Signal Name Description(1, 2) | 143 |
| Timer/Counter Description..... | 144 |
| Capture Operating Mode | 146 |
| Waveform Operating Mode..... | 149 |
| TC User Interface | 152 |
| TC Block Control Register | 153 |
| TC Block Mode Register..... | 154 |
| TC Channel Control Register..... | 155 |

| | |
|--|-------------------|
| TC Channel Mode Register: Capture Mode | 156 |
| TC Channel Mode Register: Waveform Mode | 158 |
| TC Counter Value Register..... | 161 |
| TC Register A | 161 |
| TC Register B | 162 |
| TC Register C | 162 |
| TC Status Register | 163 |
| TC Interrupt Enable Register | 164 |
| TC Interrupt Disable Register | 165 |
| TC Interrupt Mask Register..... | 166 |
| <i>SPI: Serial Peripheral Interface</i> | <i>167</i> |
| Pin Description..... | 167 |
| Master Mode..... | 168 |
| Slave Mode..... | 172 |
| Data Transfer..... | 173 |
| Clock Generation | 174 |
| Peripheral Data Controller | 174 |
| SPI Programmer's Model..... | 175 |
| SPI Control Register | 176 |
| SPI Mode Register..... | 177 |
| SPI Receive Data Register | 178 |
| SPI Transmit Data Register | 179 |
| SPI Status Register | 180 |
| SPI Interrupt Enable Register | 181 |
| SPI Interrupt Disable Register | 182 |
| SPI Interrupt Mask Register..... | 183 |
| SPI Receive Pointer Register | 184 |
| SPI Receive Counter Register | 184 |
| SPI Transmit Pointer Register | 185 |
| SPI Transmit Counter Register | 185 |
| SPI Chip Select Register | 186 |
| <i>JTAG Boundary-scan Register.....</i> | <i>188</i> |
| <i>Document Details</i> | <i>192</i> |
| Revision History..... | 192 |
| <i>Table of Contents</i> | <i>i</i> |



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe

Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia

Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan

Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory

Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Microcontrollers

Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Atmel Nantes

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Atmel Smart Card ICs
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive

Atmel Heilbronn
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80



e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

© Atmel Corporation 2002.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL® is the registered trademark of Atmel.

ARM®, Thumb® and ARM Powered® are the registered trademarks of ARM Ltd. ARM7TDMI™ is the trademark of ARM Ltd. Other terms and product names may be the trademarks of others.



Printed on recycled paper.